Notes on Metamathematics

Warren Goldfarb

W.B. Pearson Professor of Modern Mathematics and Mathematical Logic

Department of Philosophy
Harvard University

DRAFT: January 1, 2018

In Memory of Burton Dreben (1927–1999), whose spirited teaching on Gödelian topics provided the original inspiration for these *Notes*.

Contents

Chapter 1

Axiomatics

1.1 Formal languages

The subject of our study is formal systems, that is, precisely and rigorously specified axiom systems. A formal system has two parts: a formal language, which provides a precisely demarcated class of expressions called *formulas*; and derivation rules (axioms and rules of inference), which determine a privileged class of formulas that are said to be *derivable* in the formal system. We start by looking at formal languages.

A formal language is specified by giving an *alphabet* and *formation rules*. The alphabet is the stock of primitive signs; it may be finite or infinite. The formation rules serve to specify those strings of primitive signs that are the formulas of the formal language. (A string is a finite sequence of signs, written as a concatenation of the signs without separation.) In some books, formulas are called "well-formed formulas", or "wffs", but this is redundant: to call a string a formula is to say it is well-formed. A formal language must be effectively decidable; that is, there must be a purely mechanical procedure, an algorithm, for determining whether or not any given sign is in the alphabet, and whether or not any given string is a formula. (In Chapter 5 we give a rigorous account of what is meant by "purely mechanical procedure". For now, we rely on a loose intuitive understanding of the notion.)

Here is a simpleminded example. The alphabet consists of the signs 'Δ' and '$\star$', and the formation rules are:

(1) '$\Delta\star$' is a formula;

(2) if F is a formula then so is F followed by '$\star\star$';

(3) no string is a formula unless its being so results from clauses (1) and (2).

In this formal language, for example, '$\Delta\star\star\star$' is a formula, but '$\Delta\star\star$' is not. Although this simpleminded formal language lacks the complexity that makes formal languages of interest, it does illustrate some general points.

First, the formation rules are inductive: to specify the class of formulas we stipulate first that certain particular strings are formulas, and then that certain operations, when applied to formulas, yield new formulas. Finally, we specify that only strings obtained in this way are formulas. Thus, the class of formulas is the smallest class containing the string mentioned in clause (1) and closed under the operation given in clause (2). Henceforth, in our inductive definitions we shall tacitly assume the final "nothing else" clause.

Moreover, these inductive rules have a special feature. The operations for constructing new formulas out of old ones increase the length of the string. This feature makes it easy to check whether a string is a formula. Hence it yields the effective decidability of the class of formulas.

In specifying the alphabet above, we wrote down not the signs themselves but rather the names of those signs, just as in specifying a bunch of people we would write down names of those people. Much of what we shall be doing in this book is talking about formal languages. The language in which we talk about formal languages is the *metalanguage*, which is English amplified by some technical apparatus. The language being talked about is often called the *object language*.

In clause (2) of the formation rules we used the *syntactic variable* 'F'. Uppercase eff is not part of the formal language; it is part of the metalanguage. In the metalanguage it functions as a variable, ranging over strings of signs.

A short discussion of use and mention is now in order. We use words to talk about things. In the sentence

(a) Frege devised the notion of formal system.

the first word is *used* to refer to a German logician. The sentence *mentions* this logician, and *uses* a name to do so. Similarly, in the sentence

(b) The author of *Begriffsschrift* devised the notion of formal system.

the same logician is mentioned, and the expression consisting of the first four words is used to mention him. We shall speak of complex expressions like those four words also as names, so in (b) we have a complex name of Frege. In general, to speak of an object we use an expression that is a name of that object or, in other words, an expression that refers to that object. Clearly the object mentioned is not part of the sentence; its name is.

Confusion may arise when we speak about linguistic entities. If I wish to mention (talk about) an expression, I do not use that expression—for if I did I would be mentioning the object that the expression refers to, if any. Instead, I should use a name of that expression. Thus I might say

The first word of sentence (a) refers to a German logician.

Here, the first six words comprise a name of a linguistic expression. One standard manner of forming names of expressions is to surround the expression with single quotation marks. Thus we may say

'Frege' refers to a German logician.

Similarly,

'The author of *Begriffsschrift*' refers to a German logician.

Thus, to obtain a true sentence from '___ is a primitive sign of the simpleminded formal language' we must fill the blank with a name of a primitive sign, not with the primitive sign, for example:

'$\star$' is a primitive sign of the simple-minded formal language.

We cannot say: $\star$ is a primitive sign. That is nonsense, since a star is not a sign of English.

Now let us consider the use of syntactic variables. Let F be a formula of the simple-minded formal language. Here 'F' is used as a syntactic variable. Our formation rules tell us that F followed by '$\star\star$' is also a formula. We also say that $F^\frown$'$\star\star$' is a formula, where '$^\frown$' is a metalinguistic sign for the concatenation operator. We may even say that $F^\frown$'$\star$'$^\frown$'$\star$' is a formula (and read this as follows: F concatenated with star concatenated with star), for, after all, '$\star$'$^\frown$'$\star$' is identical with '$\star\star$'. We may not speak this way: $F\star\star$ is a formula; nor this way: '$F\star\star$' is a formula. The former doesn't work, since '$F\star\star$' is not a referring expression, and hence cannot be used to mention anything. The latter doesn't work, since '$F\star\star$' is the string consisting of an upper-case eff and two stars, and this string is not even a string of signs of the formal language, much less a formula.

Similarly, we can say: let G be a string consisting of an odd number of stars; then 'Δ'$^\frown G$ is a formula. Also, let F be a formula and let G be a string consisting of an even number of stars; then $F^\frown G$ is a formula.

The distinctions between name and thing named, and between syntactic variable and formal sign, should be thoroughly understood, particularly because—for the sake of brevity—we shall soon abandon the pedantic mode of speech in which the distinctions are strictly observed.

Now let us given an example of a formal language more typical of those considered in logical studies, which we shall call $L_=$, the language of identity. The alphabet consists of the six signs '$=$', '$\sim$', '$\supset$', '$\forall$', '(' and ')' along with the *formal variables* 'x', 'y','z', 'x'', 'y'', 'z'', 'x''', … . The first four signs are called the identity sign, the negation sign, the conditional sign, and the universal quantifier. The formation rules are as follows:

(1) if u and v are formal variables, then $u{}^\frown{}$'$=$'${}^\frown v$ is a formula;

(2) if F and G are formulas, then '$\sim$'${}^\frown F$ and '('${}^\frown F{}^\frown$'$\supset$'${}^\frown G{}^\frown$')' are formulas;

(3) if F is a formula and u is a formal variable, then '$\forall$'${}^\frown u{}^\frown$'('${}^\frown F{}^\frown$')' is a formula.

Thus the following are examples of formulas of $L_=$:

$$\sim\forall x'(x' = y')$$

$$\forall x(\sim x = 0 \supset \sim\forall y(\sim(x = y)))$$

The formation rules just set out are hard to read, due to the care with which we have abided by the use-mention distinction, and in particular have differentiated syntactic variables from signs of the formal language. For brevity, we now introduce some metalinguistic conventions that enable us to gloss over these niceties. More precisely, we indulge in a bit of ambiguous notation. We let each sign of the formal language serve in the metalanguage as a name of itself, thereby eliminating the need for quote-names of those signs (although we shall ordinarily continue to use quote-names when mentioning a sign individually). Moreover we use concatenation in the metalanguage to represent concatenation of strings of signs. With these conventions we may rewrite the rules as follows:

(1) if u and v are formal variables, then $u = v$ is a formula;

(2) if F and G are formulas, then $\sim F$ and $(F \supset G)$ are formulas;

(3) if F is a formula and u is a formal variable then $\forall u(F)$ is a formula.

Second, now that we have set up the precise formation rules, we shall often omit parentheses when we talk of formulas. For example, we will usually drop the the outermost parentheses, and use $x = y \supset y = x$ for the formula $(x = y \supset y = x)$. We will also drop them from consecutive quantifiers, using, say $\forall x \forall y(x = y)$ to mean the formula $\forall x(\forall y(x = y))$; similarly we will use $\forall x(x = x \supset x = x)$ for the formula $\forall x((x = x \supset x = x))$. Finally, we will drop them from around a syntactic variable in a quantification, writing $\forall xF$ instead of $\forall x(F)$. Of course, we omit parentheses only when it is clear and unambiguous how to reinstate them so as to yield a formula.

We end this section, first, by recalling an important syntactic definition from elementary logic. If u is a formal variable, then the quantifier $\forall u$ is said to *bind* u. An occurrence of u in a formula is *bound* if it is within the scope of a quantifier binding u; otherwise the occurrence is *free*. A formula with free variables is said to be *open* and without them *closed*. A closed formula is also called a *sentence*. Second we define the *instances* of a universal quantification $\forall uF$ as those formulas obtained from F by substituting a variable v for the free occurrences of the variable u, provided those newly introduced occurrences of v all remain free. Another way of putting the definition is this: call a variable v *free for* u in F iff no free occurrence of u in F lies in the scope of a quantifier binding v. Then an instance of $\forall uF$ is any formula obtained from F by replacing free occurrences of u with occurrences of a variable that is free for u in F. (Note that u is free for u in F, so F itself counts as an instance of $\forall uF$.

1.2 Axioms and rules of inference

A formal system consists of a formal language together with derivation rules, which split into two parts. The first part stipulates that certain formulas are *axioms*; the second part provides *rules of inference* for obtaining formulas from other formulas. Given axioms and inference rules, we define the notions of derivation and derivability as follows: A *derivation* is a finite sequence $F_1, \ldots, F_n$ of formulas such that each formula in the sequence either is an axiom or else results by a rule of inference from formulas that precede it in the sequence. A derivation $F_1, \ldots, F_n$ is said to be a derivation *of* its last formula F_n. A formula is *derivable* iff there is a derivation of it. If Σ is a formal system and F is a formula of the formal language, we write '$\vdash_\Sigma F$' for 'F is derivable in Σ', omitting the subscript 'Σ' when the context fixes which formal system is meant.

The derivation rules of a formal system must be effective: there must be a purely mechanical procedure for determining, given any formula F, whether or not F is an axiom; and a purely mechanical procedure for determining, given a formula F and some other formulas, whether or not F results from the other formulas by a rule of inference. It follows that there is a purely mechanical procedure for determining whether or not any given sequence of formulas is a derivation. The underlying idea is that whether or not a sequence of formulas is a derivation is to be determined by looking at the syntactic structure of the formulas.

We now give the derivation rules for a formal system, $\Sigma_=$, whose formal language is $L_=$. To help in discussing the rules, we split the axioms into three groups and give them labels.

Truth-functional axioms. If F, G, and H are formulas then the following are axioms:

(T1) $F \supset (G \supset F)$

(T2) $(F \supset (G \supset H)) \supset ((F \supset G) \supset (F \supset H))$

(T3) $(F \supset G) \supset ({\sim}G \supset {\sim}F)$

(T4) $F \supset {\sim}{\sim}F$

(T5) ${\sim}{\sim}F \supset F$

Quantificational axioms. If F is a formula, F' an instance of F, u a formal variable, and G a formula in which u is not free, then the following are axioms:

(Q1) $\forall u(F) \supset F'$

(Q2) $\forall u(G \supset F) \supset (G \supset \forall uF)$

Axioms of identity.

(I1) $x = x$

(I2) $x = y \supset (F \supset G)$, where F and G are any formulas that differ only in that G has free y at some or all places where F has free x.

These axioms express the traditional logical understandings of the signs of our formal language: '$\sim$' and '$\supset$' are to be read as negation and material conditional (*not* and *if-then*), '$\forall$' as the universal quantifier (*for all*) and '$=$' as identity.

Rules of inference. Let F and G be formulas and u be a variable,

Modus ponens: G may be inferred from F and $F \supset G$

Universal generalization: $\forall uF$ may be inferred from F.

Note that in specifying the truth-functional axioms we gave axiom *schemata*, each of which gives rise to an infinite number of axioms by replacement of the syntactic variables with formulas. Similarly for the quantificational axioms and the second axiom of identity. In contrast, axiom (I1) is a particular formula. The axioms generated by (Q1) are axioms of universal instantiation; by dint of them and modus ponens, if a formula is derivable then so are its instances.

Let us give several examples of derivations in this formal system. The first is fairly straightforward.

$x = x$
$\forall x(x = x)$
$\forall x(x = x) \supset y = y$
$y = y$

Here, the first formula is axiom (I1), the second results from it by application of the rule of universal generalization, the third is an axiom (Q1) of universal instantiation, and the fourth results from the second and third by modus ponens. This illustrates a general feature of the system. If a formula F containing a free variable u is derivable, then so will be the formula obtained from F by relettering the variable, as long as it remains free.

Our next derivation is a little more complex.

$x = y \supset (x = x \supset y = x)$
$(x = y \supset (x = x \supset y = x)) \supset ((x = y \supset x = x) \supset (x = y \supset y = x))$
$(x = y \supset x = x) \supset (x = y \supset y = x)$
$x = x \supset (x = y \supset x = x))$
$x = x$
$x = y \supset x = x$
$x = y \supset y = x$

The first, second, fourth, and fifth formulas are axioms ((I2), (T2), (T1), (I1), respectively). The third results from the first two by modus ponens; the sixth results from the fourth and fifth by modus ponens; and the last results from the third and sixth by modus ponens. Thus the symmetry of '=' is derivable, using just the truth-functional and identity axioms.

Finally, we want to show that the transitivity of '=' is derivable. Here we shall use the observation above about relettering of free variables, and not go through the steps of using universal generalization and instantiation. First we note that $\vdash x = y \supset (x' = x \supset x' = y)$, since it is an axiom (I2). (Recall that '$\vdash$' means "is derivable".) Relettering y as z, we obtain $\vdash x = z \supset (x' = x \supset x' = z)$, then, relettering x as y, $\vdash y = z \supset (x' = y \supset x' = z)$, and finally relettering x' as x, $\vdash y = z \supset (x = y \supset x = z)$. This last formula is a way of expressing transitivity.

We have just shown by a metamathematical argument that $y = z \supset (x = y \supset x = z)$ is derivable, that is, that there is a sequence of formulas obeying certain syntactic restrictions and ending with $\vdash y = z \supset (x = y \supset x = z)$. In this argument we did not actually exhibit the derivation. Of course, we can always show a formula derivable in PA by giving a derivation of it, by writing down the sequence of formulas. But since formal derivations quickly become very long and tedious, we eschew these direct verifications of derivability. Instead, we show general principles about derivability and use them to show that a derivation exists. It is essential to bear in mind that the metamathematical arguments are not the derivations: they establish the existence of derivations without actually exhibiting them.

An especially useful general principle for establishing derivabilities is this: axioms (T1)–(T5) together with modus ponens yield the derivability of all truth-functionally valid formulas. (A formula is truth-functionally valid if it is built up from some parts by use of '$\sim$' and '$\supset$', and every assignment of truth-values to those parts makes the whole formula come out true.) In a phrase, the system is *truth-functionally complete*. (T1)–(T5) were axioms of the first fully laid-out axiomatic system for truth-functional logic, namely that of Frege in *Begriffsschrift* (1879). It is impressive that he formulated a system that turned out to be truth-functionally complete, even though the concept of truth-functional validity was not articulated until nearly forty years later. (Frege's system had an additional axiom, which turned out to be redundant.) The first published proof of the truth-functional completeness of an axiomatic system was due to the American logician Emil Post (1921). We won't pause now to prove the property for our system; a proof is outlined in Appendix §1.

Here is a typical application of truth-functional completeness. A more natural expression of transitivity than the formula used above is $x = y \supset (y = z \supset x = z)$. To show it derivable, note that the formula

$$(y = z \supset (x = y \supset x = z)) \supset (x = y \supset (y = z \supset x = z))$$

is truth-functionally valid (it has the form $(F \supset (G \supset H) \supset (G \supset (F \supset H)))$. Hence it is derivable. Since $\vdash y = z \supset (x = y \supset x = z)$, we obtain $\vdash x = y \supset (y = z \supset x = z)$ by modus ponens. In fact, we can make the argument more concise yet. The fact that the displayed formula above is truth-functionally valid is just the fact that $(y = z \supset (x = y \supset x = z))$ truth-functionally implies $(x = y \supset (y = z \supset x = z))$. Truth-functional completeness (and modus ponens) tell us that if one derivable formula truth-functionally implies another, then the other is also derivable.

An even more natural formulation of transitivity would be $(x = y \,\textbf{.}\, y = z) \supset x = z$, where '$\textbf{.}$' is a sign for conjunction (*and*). However, $L_=$ has no sign for conjunction. Yet, of course, conjunction and all the other truth-functions are definable in terms of negation and conditional. Hence we proceed as follows: we introduce several metalinguistic signs as abbreviations, namely the signs $\textbf{.}$, $\vee$ and $\equiv$. If F and G are formulas we write

$$\begin{array}{lll} F \,\textbf{.}\, G & \text{for} & {\sim}(F \supset {\sim}G) \\ F \vee G & \text{for} & {\sim}F \supset G \\ F \equiv G & \text{for} & (F \supset G) \,\textbf{.}\, (G \supset F) \end{array}$$

Note that these are not signs of the formal language, nor are they "defined signs of the formal language" (whatever that would mean), nor are we proposing a new formal language that incorporates them (although that, of course, could be done). They are signs of the metalanguage, used to provide short names of long formulas. For example, we can write $x = y \,\textbf{.}\, y = z \supset x = z$, and mean thereby the formula ${\sim}(x = y \supset {\sim}y = z) \supset x = z$. Since $x = y \supset (y = z \supset x = z)$ truth-functionally implies $x = y \,\textbf{.}\, y = z \supset x = z$, it follows that the latter is also derivable.

We also introduce '$\exists$' as a metalinguistic abbreviation, writing $\exists uF$ for ${\sim}\forall u({\sim}F)$. Now if $\forall u({\sim}F) \supset {\sim}F'$ is an axiom (Q1), we may note that since it truth-functionally implies $F' \supset {\sim}\forall u({\sim}F)$, we obtain the derivability of $F' \supset \exists uF$, which is a form of existential generalization.

1.3 Natural numbers: the successor function

The focus of our metamathematical attention in this book are formal systems intended to capture laws of arithmetic, that is, properties of the natural numbers (nonnegative integers) $0, 1, 2, 3, \ldots$. The *successor function* is the function that takes each natural number to the next one; the natural numbers are generated from 0 by repeated application of this function. Let us start by giving a system that formalizes properties of this function. The formal language L_S amplifies the alphabet of language $L_=$ by adding two primitive signs, '0' and 'S'. In order to specify the formulas, we first specify a class of strings called *terms*.

1. '0' is a term; any formal variable is a term;
2. if t is a term then so is St.

Thus a term is '0' or a formal variable by itself, or a string of successive occurrences of 'S' followed by either '0' or a formal variable. Now, as for formulas:

1. if s and t are terms, then $s = t$ is a formula;
2. if F and G are formulas,then $\sim F$ and $(F \supset G)$ are formulas;
3. if F is a formula and v is a formal variable then $\forall v(F)$ is a formula.

Clearly (2) and (3) are the same rules for constructing complex formulas from simple ones as in $L_=$. The only difference in the formation rules lies in in the specification of the *atomic formulas*, those licensed by clause (1), which do not contain '$\sim$' '$\supset$', or '$\forall$': the atomic formulas now are equations between terms of L_S, rather than just equations between formal variables.

We now specify axioms for a system Σ_S. The logical truth-functional, quantificational, and identity axioms are all framed as in §2. Of course, the formulas referred to are formulas of L_S, so in saying, for example, that for all formulas F and G, $F \supset (G \supset F)$ is an axiom, we mean for all formulas F and G of L_S. Moreover, the notion of instance is expanded: instances of $\forall uF$ are formulas that are obtained from F by replacing u with a term, provided any variable in the term is free for u.

The axioms of successor are as follows:

(S1) $\sim Sx = 0$

(S2) $Sx = Sy \supset x = y$

(S3) $\sim x = 0 \supset \exists y(x = Sy)$

(S4) $\sim S \ldots Sx = x$, where '$S \ldots S$' represents any nonempty string of successive occurrences of 'S'.

Axioms (S1) - (S3) are each individual formulas of L_S, while (S4) is an infinite list of formulas. As before, the rules of inference are modus ponens and universal generalization.

Because the formal system contains universal instantiation as axioms and universal generalization as a rule of inference, it doesn't matter whether we formulate axioms like (S1)-(S4) as open formulas, as above, or as universally quantified closed formulas, for example, $\forall x \forall y(Sx = Sy \supset x = y)$. Each of these forms is derivable from the other using (Q1), modus ponens, and universal generalization. For some purposes, not at issue in this volume, it is important that all nonlogical axioms—those aside from the truth-functional axioms, quantificational axioms, and axioms of identity—be closed; hence some authors use only the universally quantified forms. (See the Exercises for §2.5 for an example of where this is needed.) The interderivability of the closed and open forms of the axioms motivate an extension of our terminology: we shall call any formula obtainable from an axiom by generalization and instantiation an instance of that axiom.

As an example, let us show that $x = S0 \supset \sim x = SSy$ is derivable in Σ_S. We have $\vdash \sim 0 = Sy$, from an instance of (S1) and the symmetry of identity. We also have $\vdash S0 = SSy \supset 0 = Sy$, an instance of (S2). By truth-functional implication, $\vdash \sim S0 = SSy$. Now, since by axiom (I2) $\vdash x = S0 \supset (x = SSy \supset S0 = SSy)$, by another truth-functional implication we can infer $\vdash x = S0 \supset \sim x = SSy$.

In fact Σ_S can derive every formula that is true about the integers and successor. To make this claim precise, we must talk about interpretations of the language L_S. Intuitively, an interpretation of a formal language is specified by providing meanings to the signs. The interpretation of the logical signs is fixed: '$=$' is interpreted as identity, '$\sim$' as negation, '$\supset$' as material conditional, and '$\forall$' as universal quantification. Thus, all an interpretation of $L_=$ would need to fix is the universe over which the quantifiers range. L_S, on the other hand, also contains the signs '0' and 'S'. Syntactically, '0' functions as a constant, that is, a name of an object, and 'S' as a one-place function sign. Hence an interpretation of L_S consists of first a (nonempty) universe, second an interpretation of '0' as an element of the universe, and third an interpretation of 'S' as a function on the universe whose values lie in the universe. Here is an interpretation: the universe is the natural numbers; '0' is interpreted as zero and 'S' as the successor function. Thus, under this interpretation, '$SSS0$' refers to three; and $\forall x(\sim Sx = 0)$ asserts that zero is the successor of no natural number.

The interpretation we have just given is the *intended interpretation*, the one we had in mind when we formulated L_S. Other interpretations may easily be devised; for example, we could take the universe to be all integers, negative, zero, and positive, with 'S' as the successor function on them, and '0' as zero; or take the universe to be the natural numbers together with two other objects, which we'll call a and

b, and interpret 'S' as the successor function on the natural numbers and takes a to b and b to a.

Suppose we have an interpretation of L_S. Let F be a sentence of L_S, that is, a formula without free variables. Then either F is true under the interpretation or else F is not true, that is, it is false under the interpretation. For example, $\forall x(\sim Sx = 0)$ and $\forall x(\sim SSx = x)$ are true under the intended interpretation, because zero is not the successor of any number, and no number is the double successor of itself, whereas $\forall x(\exists y(x = Sy)) \supset (\exists z(x = SSz))$ is not, because not all numbers that are successors of some number are double successors of some number (the number one is the sole counterexample). In the first variant interpretation, $\forall x(\sim Sx = 0)$ is false, since zero is the successor of minus one, while $\forall x(\sim SSx = x)$ is true, as is also $\forall x(\exists y(x = Sy) \supset \exists z(x = SSz))$, since in fact every member of the universe is the double successor of something. In the second variant interpretation, $\forall x(\sim SSx = x)$ is false, because the function that interprets 'S' when applied twice takes a to itself, and also takes b to itself.

What about open formulas, that is, formulas with free variables? Consider, for example, $\forall x(\sim y = SSx)$, in which y is free. Under interpretation, this formula is true or false once the free variable y is assigned a value in the universe of discourse. Indeed, under the intended interpretation this formula is true for values zero, one, and two of y, and false for all other values. (In the first variant interpretation, it is true for no values of y.) In general, under an interpretation a formula is true or false for given assignments of values to the free variables. We also call an open formula true under an interpretation without qualification (that is, without reference to an assignment of values to the free variables) iff it is true for *all* assignments of values in the universe of discourse to the free variables. Thus we would say that $\sim Sx = 0$ is true under the intended interpretation.

Semantics is the study of interpretations of formal languages, and of properties of signs that are defined with reference to interpretations. (In purely mathematical contexts this is also called *model theory*, since interpretations are also called models, and the intended interpretation of a system like Σ_S is called the *standard model.*) Syntax is the study of purely formal properties of signs and formal systems, with no mention of interpretation. The central notion of semantics is truth under an interpretation; the central notion of syntax is derivability. Of course there can be connections between these notions, as we shall shortly see.

1.4 General notions

Let Σ be a formal system whose language contains the usual logical signs. We define several important notions about derivability in Σ. In these definitions, we use 'formula' to mean formula of the formal language of Σ.

- Σ is *consistent* iff for no formula F are both F derivable and $\sim F$ derivable.

Let us call Σ consistent* iff there is a formula F that is not derivable. Clearly we want our formal systems to be consistent*; else all formulas are derivable, and so the system loses all interest. Our interest in having our formal systems be consistent comes from the fact the sign '$\sim$' is the sign for negation. As a result, for most formal systems of interest, consistency and consistency* are equivalent. (In fact they are equivalent as long as the system is truth-functionally complete and contains modus ponens.)

- Σ is *syntactically complete* iff for every sentence F either F is derivable or else $\sim F$ is derivable.

Recall that a sentence is a formula without free variables. Call a sentence *refutable* when its negation is derivable. Then the definition may be rephrased thus: Σ is syntactically complete iff every sentence is either derivable or refutable. The restriction to sentences, rather than arbitrary formulas, is important, since ordinarily formulas with free variables will be neither provable nor refutable. For example, in system Σ_S the formula $x = 0$ is neither derivable nor refutable. Nor would we want it to be, for if it were then by universal generalization either $\forall x(x = 0)$ or $\forall x(\sim x = 0)$ would be derivable, and neither is a happy result, since either would make the system inconsistent.

The property of syntactic completeness is sometimes called 'formal completeness' or 'negation completeness'. Note that syntactic completeness, like consistency, is a purely syntactic notion. It is important to distinguish syntactic completeness from other completeness notions, one of which we've seen already (truth-functional completeness), one of which is defined below, and one of which we'll encounter in the next section. The use of the word 'complete' for many different notions is a pun, historically engendered by the vague intuition that a formal system should be called complete when it does everything we want it to do. At different times and for different systems, what 'we want it to do' led to different notions.

- Σ is *decidable* iff there is a purely mechanical procedure for determining whether any given formula is derivable in Σ.

We require of all formal systems only that the notion of derivation be effective, not the notion of derivability. In particular, since the rules of inference may allow a shorter formula to be inferred from longer formulas, there may be no obvious way of telling from a formula how long a derivation of it might have to be. Whether or not a system is decidable is thus a real question, and often takes considerable work to settle.

Let us now leave syntax and define two semantic notions.

- Σ is *sound with respect to an interpretation* iff every formula derivable in Σ is true under that interpretation.

- Σ is *complete with respect to an interpretation* iff every formula true under that interpretation is derivable in Σ.

The power of semantic talk is illustrated by these cheerful facts: if there is at least one interpretation with respect to which Σ is sound, then Σ is consistent; if there is at least one interpretation with respect to which Σ is complete, then Σ is syntactically complete. This follows from the fact that — since '$\sim$ ' is always interpreted as negation — for any sentence F, either F is true or $\sim F$ is true, but not both.

The definitions given in this section are completely general. Let us now restrict attention to formal systems that use the same logical axioms and rules of inferences as those given in §1.2. Those axioms are true under all interpretations, and the rules of inference preserve truth: if the premises of an application of either of these rules is true under an interpreation, so is the conclusion. Consequently, if the nonlogical axioms of a system are true under an interpretion, then in any derivation $F_1, \ldots, F_n$, every formula, either being an axiom or resulting by a rule of inference from previous formulas, must be true in the interpretation, and so all derivable formulas will be true under the interpretation and the system will be sound for the interpretation. A corollary of this is: if the nonlogical axioms are true under a given interpretation and a formula F is not true under that interpretation, then F cannot be derivable.

How do the systems we have formulated in this chapter fare under these definitions? The axioms of $\Sigma_=$, being purely logical, are true under every interpretation. Obviously, then, they are consistent. The system is not syntactically complete, nor would we want it to be, since it is intended to express the general logical laws of identity, not the facts about a particular mathematical domain. In particular, for example, $\forall x \forall y(x = y)$ is neither derivable nor refutable: it is true in the interpretation with a one-element domain, and false in all other interpretations. $\Sigma_=$ is also decidable, which is not difficult, but also not trivial, to prove. As it turns out, a sentence F of $L_=$ is derivable in $\Sigma_=$ iff it holds in all universes of size at most m, where m is the number of quantifiers in F; and it is easy to compute when this property holds. (See Appendix §3.)

For system Σ_S, we have the following. All of (S1) - (S4) are true in the intended interpretation, so Σ_S is sound for that interpretation, and hence consistent. Moreover, Σ_S is complete for that interpretation, and hence syntactically complete. This was first shown by the French logician Jacques Herbrand, in his doctoral dissertation (1930). His proof also yields the decidability of the system. In fact, it yields more. For Herbrand showed how to construct, for any formula F, a formula G with no quantifiers and the same free variables as F such that $\vdash F \equiv G$. It follows that every formula F with the one free variable x will hold (in the intended interpretation) either for finitely many values of x or for all but finitely many values of x. (See the Exercises). Thus even very simple arithmetical notions like "x is odd" cannot be expressed. To formalize more serious arithmetical facts, a larger vocabulary than just '0' and 'S' is needed.

1.5 Peano Arithmetic.

The standard formal system for the formalization of arithmetic is usually called Peano Arithmetic, although this name is something of a historical misnomer (see §5.4). It is more accurately called *first-order arithmetic.* We expand the vocabulary of L_S to include function signs for addition, and multiplication. Thus we take L_{PA} to have, as its alphabet: the logical signs '$\sim$ ', '$\supset$', '$\forall$ ', '$=$', '(', ')', the arithmetical signs '0', 'S', '$+$', '$\times$', and the formal variables 'x', 'y', 'z', 'x'', 'y'', … . The formation rules first specify the terms:

1. 0 is a term; each formal variable is a term.

2. If s and t are terms then so are St, $(s + t)$, and $(s \times t)$.

So some examples of terms are $SSS0$, $((SS0+SSS0)\times SS0)$, and $((Sx\times Sy)+SSS0)$. Now, as for formulas:

1. If s and t are terms then $s = t$ is a formula (an atomic formula).

2. If F and G are formulas then so are $\sim F$ and $(F \supset G)$.

3. If F is a formula and u is a formal variable then $\forall u(F)$ is a formula.

If t is a term, u a formal variable, and F a formula, we say t is *free for* u in F iff every variable occurring in t is free for u in F. The *instances* of a formula $\forall uF$ are all the formulas that can be obtained from F by replace the free occurrences of u by occurrences of a term t that is free for u in F. We also introduce a notation for substitution in formulas. We will use a syntactic variable $F(u)$ for a formula ordinarily containing free occurrences of the variable u, and then, if t is a term of the language that is free for u in $F(u)$, $F(t)$ will stand for the result of substituting t for all free occurrences of u in $F(u)$. The same convention will govern the use of syntactic variables $F(u, v)$, and so on, with more variables indicated. We also allow $F(u)$ to stand for a formula without free occurrences of u, but in that case $F(t)$ will be the same formula as $F(u)$.

Now let us specify the axioms of the formal system PA. The logical axioms are just as in 1.2, with the understanding, of course, that the syntactic variables F, G, H range over formulas of L_{PA}. The rules of inference are, also as in previous sections, modus ponens and universal generalization. What is new are the number-theoretical axioms.

(N1) $\sim Sx = 0$

(N2) $Sx = Sy \supset x = y$

(N3) $x + 0 = x$

(N4) $x + Sy = S(x + y)$

(N5) $x \times 0 = 0$

(N6) $x \times Sy = (x \times y) + x$

(N7) $F(0) \mathbin{.} \forall x(F(x) \supset F(Sx)) \supset \forall xF(x)$

Note that (N1)–(N6) are particular formulas, but (N7) is an axiom schema: replacing $F(x)$ by any formula of L_{PA} yields an axiom. This schema provides the mathematical induction axioms. Intuitively, mathematical induction is the principle that if 0 possesses a property and if whenever a number possesses the property then so does its successor, then all numbers possess the property. The power of PA to derive formalizations of interesting arithmetical claims — including all the classical theorems of number theory — stems from the inclusion of the mathematical induction axioms.

Now the intended interpretation of L_{PA} is, not surprisingly, that the universe is the natural numbers, '0' denotes zero, 'S', '$+$' and '$\times$' denote the successor function, addition, and multiplication. It can look fairly obvious that PA is sound for this interpretation, but this obscures a difficulty. As we shall see in Chapter 5, framing the notion of truth in this interpretation requires a metalanguage that is expressively richer than what can be formalized in L_{PA}, and, despite its superficial obviousness, demonstrating that PA is sound for this interpretation requires a metalanguage that is mathematically stronger than what is formalized by PA. (The problem, as it turns out, lies in the unbounded logical complexity of the axioms of mathematical induction. In particular the formula put in for $F(x)$ in (N7) can have arbitrarily many quantifiers.) For this reason we avoid semantical reasoning in proving the results of the next three chapters. Semantic considerations will appear only as heuristic or suggestive. In my view, in the study of foundations of mathematics, we should avoid strong assumptions in the metalanguage, assumptions which are in as much need of a foundation as is the mathematics that we are trying to ground by formulating formal systems and investigating them.

Let us, then, return to the syntactic investigation of PA. Although the axioms of PA includes those we called (S1) and (S2) for the system Σ_S, here renamed (N1) and (N2), it does not include (S3) and (S4), because they are derivable in PA using mathematical induction. We shall show this at the beginning of the next section. Hence every axiom of system Σ_S is derivable in PA, so that every formula derivable in Σ_S is derivable in PA. From Herbrand's result cited at the end of §1.4, it follows that every sentence of L_{PA} that does not contain '$+$' or '$\times$' is either derivable or refutable.

Of course, the difference between L_{PA} and L_S is that L_{PA} has terms and formulas that do contain '$+$' and '$\times$'. Let us note first that the intersubstitutivity of identicals ("equals for equals yields equals") is derivable. That is,

$$\vdash x = y \supset t(x) = t(y),$$

where $t(x)$ is any term containing x and $t(y)$ comes from $t(y)$ by replacing x with y. An instance of this yields the following: suppose s, s', t and t' are terms such that t' can be obtained from t by replacing a subterm s by s'; then $\vdash s = s' \supset t = t'$.

Now let us see how (N3)–(N6) yield the derivability of equations involving addition and multiplication. (As with formulas, in speaking of terms we shall often drop the outermost pair of parentheses.) As an example, let show the derivability of $S0 + SS0 = SSS0$ (one plus two equals three). First, $\vdash S0 + 0 = S0$, since it is an instance of (N3). Then, $\vdash S0 + S0 = S(S0 + 0)$, since it is an instance of (N4). By intersubstitutivity $\vdash S(S0+0) = SS0$. By the transitivity of '$=$', $\vdash S0+S0 = SS0$. By another instance of (N4), $\vdash S0 + SS0 = S(S0 + S0)$. By intersubstitutivity, $\vdash S(S0 + S0) = SSS0$. By transitivity, $\vdash S0 + SS0 = SSS0$.

The terms $S0, SS0, SSS0, \ldots$ are called *formal numerals*. As a convention in the metalanguage, if n is a number we shall use boldface $\boldsymbol{n}$ for the formal numeral that contains n occurrences of 'S'. Thus $\mathbf{0}$ is 0 and $\mathbf{3}$ is $SSS0$. The argument we have just given, iterated as necessary, tells us that if m, n, and p are numbers such that p is the sum of m and n, then $\vdash \boldsymbol{m} + \boldsymbol{n} = \boldsymbol{p}$.

Now let's show that $SS0 \times SS0 = SSSS0$ (two times two equals four) is derivable. First, $\vdash SS0 \times 0 = 0$, by (N5). Then (N6) gives us $\vdash SS0 \times S0 = (SS0 \times 0) + SS0$. By intersubstitutivity, $\vdash (SS0 \times 0) + SS0 = 0 + SS0$. By what we just noted about the derivability of addition statements, $\vdash 0 + SS0 = SS0$. By transitivity, $\vdash SS0 \times S0 = SS0$. Another instance of (N6) yields $\vdash SS0 \times SS0 = (SS0 \times S0) + SS0$; intersubstitutivity and transitivity yield $\vdash SS0 \times SS0 = SS0 + SS0$; since $\vdash SS0 + SS0 = SSSS0$, transitivity yields $\vdash SS0 \times SS0 = SS0 + SS0$. Consequently, by transitivity, $\vdash SS0 \times SS0 = SSSS0$.

Again, this argument is a general template. It should not be hard to see how to obtain, for any numbers m, n, and p, if p is the product of m and n then $\vdash \boldsymbol{m} \times \boldsymbol{n} = \boldsymbol{p}$.

Note that in the derivations we have just been showing to exist, no use of (N7), mathematical induction, was made. This is not surprising, since the formulas being shown derivable are all particular equations, whereas the point of mathematical induction is to provide derivations of general numerical laws, a task to which we now turn.

1.6 Basic laws of arithmetic

Our invocations of axioms of mathematical induction will follow a general pattern. We shall specify a formula $F(x)$, and show that $\vdash F(0)$ and $\vdash F(x) \supset F(Sx)$. Since the latter yields $\vdash \forall x(F(x) \supset F(Sx))$ by generalization, an axiom of mathematical induction will then give us $\vdash \forall x F(x)$ and hence also $\vdash F(x)$.

Our first task is to show that the axioms (S3) and (S4) of Σ_S are derivable in PA. For (S3), let $F(x)$ be the formula ${\sim}x = 0 \supset \exists y(x = Sy)$. By axiom (I1), $\vdash 0 = 0$; and this formula truth-functionally implies ${\sim}0 = 0 \supset \exists y(0 = Sy)$, that is, $F(0)$. Hence $\vdash F(0)$. Also by (I1), $\vdash Sx = Sx$. By existential generalization, $\vdash \exists y(Sx = Sy)$. This formula truth-functionally implies ${\sim}Sx = 0 \supset \exists y(Sx = Sy)$, that is, $F(Sx)$. Hence $\vdash F(Sx)$, so $\vdash F(x) \supset F(Sx)$ by truth-functional implication. We may conclude by mathematical induction that $\vdash F(x)$,that is, $\vdash {\sim}x = 0 \supset \exists y(x = Sy)$, as desired. To show that the axioms (S4) are derivable, we'll consider the example $\sim SSx = x$; the argument is generalizable to any positive number of occurrences of 'S'. Let $F(x)$ be the formula $\sim SSx = x$. Then $\vdash F(0)$, since $F(0)$ is an instance of axiom (N1). Now $F(x) \supset F(Sx)$ is $\sim SSx = x \supset \sim SSSx = Sx$, which is truth-functionally implied by $SSSx = Sx \supset SSx = x$, which is an instance of axiom (N2). Hence $\vdash F(x) \supset F(Sx)$, so using an axiom of mathematical induction we obtain $\vdash F(x)$, and so $\vdash \sim SSx = x$.

Our next aim is to show that the commutative law of addition is derivable. We do this in three stages. First we show $\vdash 0 + x = x$. Let $F(0)$ be $0 + x = x$. Then $\vdash F(0)$, since it is an instance of axiom (N3). By intersubstitutivity, $\vdash F(x) \supset S(0 + x) = Sx$. By axiom N(4), $\vdash 0 + Sx = S(0 + x)$. By transitivity of identity, $\vdash F(x) \supset 0{+}Sx = Sx$, that is, $\vdash F(x) \supset F(Sx)$. The result follows by mathematical induction.

Second, we show $\vdash Sz + x = S(z + x)$. Let $F(x)$ be $Sz + x = S(z + x)$. By (N3), $\vdash z + 0 = z$, so that $\vdash S(z + 0) = Sz$, so $\vdash Sz = S(z + 0)$ by symmetry. Another instance of (N3) yields $\vdash Sz{+}0 = Sz$. By transitivity, $\vdash Sz{+}0 = S(z{+}0)$, that is $\vdash F(0)$. By (N4), $\vdash Sz + Sx = S(Sz + x)$. Hence by intersubstitutivity, $\vdash Sz + x = S(z + x) \supset Sz + Sx = SS(z + x)$. By (N4) again, $\vdash z + Sx = S(z + x)$, so that $\vdash S(z + Sx) = SS(z + x)$. By symmetry and transitivity, $\vdash Sz + x = S(z + x) \supset Sz + Sx = S(z + Sx)$. This is just $F(x) \supset F(Sx)$. Thus we obtain the desired conclusion.

Finally, we show $\vdash x + y = y + x$. Let $F(x)$ be $x + y = y + x$. From what was shown two paragraphs above, $\vdash 0 + y = y$, and by axiom (N3) and symmetry $y = y + 0$, so by transitivity $\vdash F(0)$. By axiom (N4), $\vdash y + Sx = S(y + x)$. An instance of what was shown in the previous paragraph yields $\vdash Sx + y = S(x + y)$. By intersubstitutivity, $\vdash x + y = y + x \supset S(x + y) = S(y + x)$. By symmetry and transitivity of identity, $\vdash x{+}y = y{+}x \supset (Sx{+}y = y{+}Sx)$, that is, $\vdash F(x) \supset F(Sx)$.

We leave to the reader arguments for the derivability of other basic laws of arithmetic, for example the associativity of addition, the law of cancellation $y{+}x = z + x \supset y = z$, the commutativity and associativity of multiplication, and the distributive law $(y + z) \times x = (y \times x) + (z \times x)$. (See the Exercises.)

We now wish to show that the basic properties of the usual ordering relation of the natural numbers can be derived in PA. PA does not have primitive vocabulary to express this relation, but, since a number m is no greater than a number n iff some natural number added to m yields n, it makes sense to introduce the following metamathematical shorthand: by $x \leqslant y$ we mean the formula $\exists z(y = z + x)$. More generally, if s and t are any terms, by $s \leqslant t$ we mean the formula $\exists u(t = u + s)$, where u is the earliest variable among $z, z', z'', \ldots$ that is distinct from all variables in s and t.

Since $\vdash x = 0+x$ and $\vdash x = x+0$, by existential generalization we have $\vdash x \leqslant x$ and $\vdash 0 \leqslant x$. To show the derivability of transitivity, that is,

$$x \leqslant y \,.\, y \leqslant z \supset x \leqslant z$$

We might argue as follows. Suppose that $x \leqslant y$ and $y \leqslant z$. Then there are z' and z'' such that $z'+x = y$ and $z''+y = z$. By intersubstitutivity, $z'+(z''+x) = z$. By the associativity of addition, $(z'+z'')+x = z$. Hence there exists an x', namely $z'+z''$, such that $x'+x = z$. That is, $x \leqslant z$.

The argument of the foregoing paragraph is more informal than those we have used previously. That it establishes the derivability of transitivity can be seen, roughly speaking, by noting that all the moves are purely logical inferences from formulas known to be derivable; and that by dint of the logical axioms, PA can capture all such inferences. More precisely, the argument shows that the formula $(z'+x = y \,.\, z''+y = z) \supset (z'+z'')+x = z$ is truth-functionally implied by appropriate instances of associativity, (N4), and intersubstitutivity. Hence this formula is derivable. Together with existential generalization, this formula truth-functionally implies $(z'+x = y \,.\, z''+y = z) \supset \exists z'(z'+x = z)$; and the latter formula logically implies

$$\exists z(z+x = y) \,.\, \exists z'(z'+y = z) \supset \exists z'(z'+x = z)$$

which is just the formula $x \leqslant y \,.\, y \leqslant z \supset x \leqslant z$. Another way of seeing that the informal argument establishes derivability is by noting that the informal argument can be directly transcribed into a natural deduction system for logical inference, like that of Goldfarb's *Deductive Logic*, yielding a deduction in that system whose premises are formulas known to be derivable and whose conclusion is the desired transitivity formula; and any implication that can be shown by such a deduction is, as we show in the Appendix §2, derivable using the logical axioms of PA.

More generally, the fact that all logically correct steps can be captured in PA can be framed as the *quantificational completeness* or *logical completeness* of PA, namely, that all quantificationally valid formulas are derivable. (A formula is quantificationally valid iff it is true under all interpretations.) As a consequence, if a formula logically implies another (in the sense of quantification theory), and the first formula is derivable, then the second formula will be, too. The quantificational completeness of a formal system of logical axioms was first shown by Kurt Gödel, in his doctoral dissertation (1930). We will not be applying quantificational completeness in our arguments in any formal way, however, but rather only the informal and more humdrum fact that our logical axioms are sufficient to formalize all customary logical inferences. (This was already verified by Frege in 1879.)

Let us show

$$\vdash x \leqslant y \mathbin{.} y \leqslant x \supset x = y$$

Suppose $x \leqslant y$ and $y \leqslant x$. Then there are numbers z and z' such that $z + x = y$ and $z' + y = x$. By intersubstitutivity, $z' + (z + x) = x$, so by associativity $(z' + z) + x = x$. Since $\vdash 0 + x = x$, $(z' + z) + x = 0 + x$. By the law of cancellation $z' + z = 0$. By the law $\vdash x + y = 0 \supset y = 0$ (Exercise 1.?), $z = 0$. Hence $0 + x = y$, so that $x = y$.

The derivability of four further laws of ordering are left to the reader (see the Exercises):

$$x \leqslant 0 \supset x = 0$$

$$x \leqslant Sx$$

$$x \leqslant y \mathbin{.} {\sim} x = y \supset Sx \leqslant y$$

$$x \leqslant y \lor y \leqslant x$$

These laws express that the ordering is a *linear ordering*, that is, any two elements are comparable; that it has a least element, namely zero; and that the ordering is *discrete*, that is, for every number there is a next one in the ordering, namely its successor (there is nothing in between a number and its successor).

Chapter 2

Gödel's Proof

2.1 Gödel numbering

In order to investigate the syntax of system PA, treating it as a mere system of signs, Gödel saw that we may proceed as follows. First we specify a mapping from signs to numbers: that is, we correlate numbers with the primitive signs of the formal language and with strings of signs of the formal language. As a result syntactic properties and relations become correlated with number-theoretic properties and relations, which may be defined and investigated using purely number-theoretic means.

Here is the correlation of numbers number with the primitive signs of L_{PA} that we shall use

'0'	with 1	'x'	with 17
'S'	with 2	'y'	with 19
'$+$'	with 3	'z'	with 21
'$\times$'	with 4	'x''	with 23
'$=$'	with 5	'y''	with 25
'$\sim$'	with 6	'z''	with 27
'$\supset$'	with 7		and so on
'$\forall$'	with 8		
'('	with 9		
')'	with 10		

More precisely put, we define a function Γ from signs of the alphabet to integers:

$\Gamma(\sigma) = m$ iff either σ is '0' and $m = 1$, or σ is 'S' and $m = 2$, or σ is '$+$' and $m = 3$, and so on. Γ is one-to-one, that is, it carries distinct signs to distinct integers. Moreover, Γ is effective: given a sign of the alphabet we can find the number correlated, and given a number we can decide whether there is a sign to which the number is correlated and if so we can find which sign this is. Γ is obviously not onto, that is, there are integers that are not correlated with any sign of L_{PA}. In fact, we purposely left gaps so that, if we want to treat a language that augments L_{PA} (as we will in §5.4), we can add new correlations without disturbing the ones already made.

We now wish to correlate numbers with strings of signs. Clearly Γ yields a correlation of a finite sequence of numbers with each string of signs. The further step that is needed is to get from finite sequences of numbers to numbers. To do this, following Gödel we shall use products of prime powers. A prime number is an integer greater than 1 whose only integral divisors are itself and 1. In order, the first ten primes are 2, 3, 5, 7, 11, 13, 17, 19, 23, 29. We define a function γ from strings to numbers as follows: if $\sigma_1 \ldots \sigma_n$ is a string, then

$$\gamma(\sigma_1 \ldots \sigma_n) = 2^{\Gamma(\sigma_1)} \cdot 3^{\Gamma(\sigma_2)} \cdot \ldots \cdot p^{\Gamma(\sigma_n)}$$

where p is the n^{th} prime number. The number to which γ carries a string is called the *gödel number* of the string. For example, $\gamma(\text{'}+\text{'}) = 2^3 = 8$, $\gamma(\text{'}\forall S)\text{'}) = 2^8 3^2 5^{10} =$ 22,500,000,000, and $\gamma(\text{'}0 = 0\text{'}) = 2^1 3^5 5^1 = 2430$. The function γ is one-to-one: distinct strings are carried to distinct numbers. (This follows from the Unique Factorization Theorem, first proved by Gauss in 1798, which asserts that every number greater than 1 has a unique factorization into prime powers.) Moreover, γ is effective.

Having given the gödel numbering, we may now deal with numbers alone. More precisely, we know that there is a property of numbers that holds of just the gödel numbers of formulas of PA, for example. We want to define this property, and others like it, entirely within number theory. Our definitions will be purely number-theoretic without any mention of syntax. Our *interest* in defining properties like this is given by the gödel numbering; but the properties are intrinsically purely number-theoretic, and their number-theoretic structure does not in any way depend on the syntax of PA or on the correlation γ. The number theory we will be using is informal number theory, as might be seen in a typical mathematics class, not the formalized version enshrined by PA. Eventually we shall want to formalize some of our proceedings; in §2.4 we shall see how much.

2.2 Primitive recursive functions and relations

We wish to define functions and relations of numbers in a purely number-theoretic way that insures that the functions and relations are computable. In order to do this, we use a delimited set of methods of definition. The definitions will provide algorithms—purely mechanical procedures—for calculating the values of the functions and for ascertaining whether the relations hold or not of any specific arguments. Functions and relations that can be defined by these methods are called *primitive recursive.*

We start by stipulating that certain basic functions are primitive recursive: all constant functions (those which take all arguments to the same value), the identity function (which takes each number to itself), and the successor function. Clearly these are all computable. To define new primitive recursive functions, we may use the methods of *recursion* and *composition.*

Recursion. The idea here is to define the value of a function for argument $k+1$ in terms of its value for argument k. That is, we first specify a value in terms of previously defined functions when one argument is 0; and then we show how, using other previously defined functions, to obtain the value when that argument is $k+1$, in terms of the value when that argument is k. Here are several examples, with the first two also rephrased in words.

Addition	$n+0=n$
	$n+(k+1)=(n+k)+1$

The value of the function on n and 0 is given by the identity function on n; the value on n and $k+1$ is the successor of the value on n and k.

Multiplication	$n \cdot 0 = 0$
	$n \cdot (k+1) = (n \cdot k) + n$

The value of the function on n and 0 is given by the constant function 0; the value of n and $k+1$ is the sum of the value on n and k and n.

Factorial	$0! = 1$
	$(k+1)! = k! \cdot (k+1)$
Exponentiation	$n^0 = 1$
	$n^{k+1} = (n^k) \cdot n$
Truncated predecessor	$\mathrm{pred}(0) = 0$
	$\mathrm{pred}(k+1) = k$

Truncated difference $n \dot{-} 0 = n$
$n \dot{-} (k+1) = \text{pred}(n \dot{-} k)$

Thus $n \dot{-} k$ is the difference between n and k if $n \geq k$ and is 0 if $n \leq k$.

For functions of two arguments, a definition by recursion looks like this: if $\psi(n)$ and $\xi(j, n, k)$ are functions already known to be primitive recursive, then we can define a new function φ thus:

$$\varphi(n, 0) = \psi(n)$$
$$\varphi(n, k+1) = \xi(\varphi(n, k), n, k)$$

Thus the value of $\varphi(n, k+1)$ is defined as some known function that has as inputs the previous value $\varphi(n, k)$, n,and k. Not all variables on the left need actually appear on the right. For example, in the definition of addition, only the previous value $n + k$ appears on the right hand side of the second equation, but neither k nor n by itself do. Even the following counts as a definition by recursion, although no variables appear on the right-hand side:

$$\begin{aligned} \alpha(0) &= 1 \\ \alpha(k+1) &= 0. \end{aligned}$$

The function α takes every positive integer to 0, and takes 0 to 1. We call α the *switcheroo* function.

The general form of definition by recursion for functions with more than two arguments is like that given above, but with a sequence $n_1, n_2, \ldots, n_m$ of arguments taking the place of n.

Composition. This is simply the compounding of given functions and relations. For example, we may define

$$|k - n| = (k \dot{-} n) + (n \dot{-} k),$$

in which truncated difference and addition are compounded. This function gives the absolute value of difference between k and n. Composition also gives us the means to capture definition by cases. For example, suppose we wanted to define the function of k and n that yields k^2 if $k \leq n$ and n^2 if $n < k$. We can do this by using addition, multiplication, truncated difference, and switcheroo thus, thereby showing that this function is primitive recursive:

$$k \cdot k \cdot \alpha(k \dot{-} n) + n \cdot n \cdot \alpha(n + 1 \dot{-} k)$$

In sum, *a function is primitive recursive iff it can be defined by starting with the basic functions and iteratively applying the definition methods of recursion and composition.*

We also want a notion of a relation's being primitive recursive. A suitable notion can be defined in terms of primitive recursive functions as follows. The *characteristic function* of an m-place relation R is the m-place function χ such that if $R(n_1, \ldots, n_m)$ holds then $\chi(n_1, \ldots, n_m) = 1$, and if $R(n_1, \ldots, n_m)$ does not hold then $\chi(n_1, \ldots, n_m) = 0$. We define: *a relation is primitive recursive iff its characteristic function is primitive recursive.*

Since primitive recursive functions are computable, so are primitive recursive relations: to determine whether a primitive relation holds at an m-tuple, simply compute its characteristic function and see whether it is 1 or not. Now $\alpha(|k-n|)$ is the characteristic function of the relation $k = n$, and $\alpha(k \dot{-} n)$ is the characteristic function of the relation $k \leq n$; hence these relations are primitive recursive relations. The set of odd numbers is primitive recursive (we identify sets and properties with 1-place relations), since its characteristic function can be defined thus: $\chi(0) = 0, \chi(k+1) = \alpha(\chi(k))$.

Additional methods of definition are allowable in defining primitive recursive functions and relations, provided they can be reduced to applications of recursion and composition. Three methods in particular will be of great use to us.

New relations can be defined from given ones by truth-functional (Boolean) combination. For example, $k > n$ iff not $n \leq k$; and $k \geq n$ iff $k > n$ or $k = n$. We claim that any truth-functional combination of primitive recursive relations is primitive recursive. It then follows that $k > n$ is primitive recursive, and that $k \geq n$ is as well. To prove the claim, note first that if an m-place relation R is primitive recursive, then so is its complement, that is, the relation that holds of an m-tuple of numbers just in case R does not. For if χ is the characteristic function of R, then the complement of R has characteristic function χ', where $\chi'(n_1, \ldots, n_m) = \alpha(\chi(n_1, \ldots, n_m))$. Second, if R and S are m-place p.r. relations then so is the intersection of R and S, that is, the relation that holds of $n_1, \ldots, n_m$ iff $R(n_1, \ldots, n_m)$ and $S(n_1, \ldots, n_m)$. For if R has characteristic function χ_R and S has characteristic function χ_S then the intersection has characteristic function χ, where $\chi(n_1, \ldots, n_m) = \chi_R(n_1, \ldots, n_m) \cdot \chi_S(n_1, \ldots, n_m)$. It follows from these observations that if R and S are primitive relations then so is the union of R and S, and, indeed, that any truth-functional combination of primitive relations is primitive recursive.

Note. In the language we have been using for informal mathematics, '=' means identity, '+' means addition, and '0' means zero. This amounts to an ambiguous usage, since we have also used these three signs as signs of the formal language L_{PA}. The context should make clear when the signs are used in informal mathematics, and when as signs (or names for signs) of the formal language, but the reader should be alert to the need to be sensitive to this. In order to minimize the overlap in language, in informal mathematics we use '$\cdot$' for multiplication, as opposed to '$\times$ in the formal language, and numerical variables from the middle of the English alphabet ('i' to 'r'), not the end. Also, in informal mathematics, we will use somewhat different logical notation: ' & ' for *and*, '$\rightarrow$' for *if-then*, '$\leftrightarrow$'' for *iff*, and an overstrike bar (over a sign for a relation) to mean *not*, for example, $n > m$ iff $n \overline{\leq} m$. The quantifiers we use in informal mathematics will also look different from those of L_{PA}. For want of a good alternative, '$\vee$' will remain as ambiguous between its informal usage and its formal usage as (inclusive) or. **End of Note.**

Another definition method we will want to use for relations is *bounded quantification*. For example,

$$\begin{array}{lll} k \text{ divides } n & \text{iff} & (\exists p \leq n)(n = p \cdot k). \\ n \text{ is prime} & \text{iff} & n > 1 \;\&\; (\forall k \leq n)(k|n \rightarrow k = 1 \vee k = n), \end{array}$$

where '$k|n$' abbreviates 'k divides n'. The bound on the quantifier insures computability: one need make only a finite search in order to determine whether the new relation holds or not. Often an equivalent definition of the relation can be made without the bound — for example, it is true that k divides n iff $(\exists p)(n = p \cdot n)$ and that n is prime iff $n > 1$ & $(\forall k)(k|n \rightarrow k = 1 \vee k = n)$— but a definition without a bound does not guarantee computability.

It is straightforward to show that if a relation $R(k, n)$ is primitive recursive then so is the relation $(\forall k \leq p)R(k, n)$, which has arguments p and n. Let χ be the characteristic function of R, and define χ' by recursion thus: $\chi'(0, n) = \chi(0, n)$, $\chi'(p + 1, n) = \chi'(p, n) \cdot \chi(p + 1, n)$. Thus χ' is primitive recursive, and is the characteristic function of $(\forall k \leq p)R(k, n)$, since $\chi'(p, n)$ is 1 just in case each of $\chi(0, n), \ldots, \chi(p, n)$ is 1, that is, just in case each of $R(0, n), \ldots, R(p, n)$ holds. Bounded existential quantification can be obtained from bounded universal quantification by truth-funcctional operations, since $(\exists k \leq p)R(k, n)$ is the complement of $(\forall k \leq p)\overline{R}(k, n)$

A final definition-method we shall use frequently is *bounded leastness*. This is used to define a new function from a given relation. The notation we use is this: an expression

$$(\mu k \leq p)R(k)$$

denotes the least number $k \leq p$ such that R holds of k, if there is such a number, and denotes 0 otherwise. Thus if we define $\varphi(n) = (\mu k \leq n)(n = k + k)$ then $\varphi(n) = n/2$ if n is even and $\varphi(n) = 0$ if n is odd. Again, the point of having a bound on the leastness operator is for the sake of computability; and again it is straightforward to show that a function defined by bounded leastness from a primitive recursive relation is itself primitive recursive. (See the Exercises.)

In short, since the primitive recursive functions and relations are closed under definition by recursion, composition, truth-functional combination, bounded quantification, and bounded leastness, when we use any of these definition methods to define new functions and relations from functions and relations known to be primitive recursive, the newly defined functions and relations will also be primitive recursive.

We conclude this section by defining five primitive recursive functions and relations concerning prime numbers and prime factorizations.

$$\begin{aligned} \mathrm{pr}(0) &= 1; \\ \mathrm{pr}(k+1) &= (\mu n \leq \mathrm{pr}(k)! + 1)(n > \mathrm{pr}(k) \ \&\ n \text{ is prime}) \end{aligned} \tag{2.1}$$

For each $k > 0$, $\mathrm{pr}(k)$ is the k^{th} prime number, so that $\mathrm{pr}(1) = 2$, $\mathrm{pr}(2) = 3$, $\mathrm{pr}(3) = 5$, and so on. The bound comes from Euclid's observation that if p is a prime number then there is a prime number greater than p and no greater than $p! + 1$. For if $2 \leq n \leq p$ then n divides $p!$, and so leaves a remainder of 1 when divided into $p! + 1$. Hence either $p! + 1$ itself is prime, or it has a prime factor which must be greater than p. (This is Euclid's proof that there are infinitely many prime numbers.)

The definition of $\mathrm{pr}(k)$ compresses several steps into one: those several steps are definitions by truth-functional combination, bounded-leastness, composition, and recursion. We shall often be giving definitions in this compressed form. This one time, let us lay out the individual definitions step-by-step. First we note that the relation that holds of m and n iff ($n > m$ & n is prime) is primitive recursive, since it is a truth-functional combination of primitive recursive relations. Next we note that the function $\varphi(j, m) = \mu n \leq j(n > m \ \&\ n \text{ is prime})$ is primitive recursive, since it is defined from a primitive recursive relation by bounded-leastness. Now let $\psi(m) = \varphi(m! + 1, m)$; ψ is primitive recursive since it is obtained from primitive recursive functions by composition. Finally, we define $\mathrm{pr}(k)$ by: $\mathrm{pr}(0) = 1$, $\mathrm{pr}(k + 1) = \psi(\mathrm{pr}(k))$. This is a definition by recursion; we may conclude that $\mathrm{pr}(k)$ is primitive recursive.

$$[n]_k = (\mu i \leq n)(\mathrm{pr}(k)^i | n \ \&\ \overline{\mathrm{pr}(k)^{i+1} | n}) \tag{2.2}$$

$[n]_k$ is the exponent of $\mathrm{pr}(k)$ in the prime factorization of n.

$$\mathrm{Seq}(n) \leftrightarrow (\exists k \le n)(\forall i \le n)(\mathrm{pr}(i)|n \leftrightarrow i \le k)) \tag{2.3}$$

$\mathrm{Seq}(n)$ holds iff n is a *sequence number*, a number in whose prime factorization appear 2, 3, ..., up through $\mathrm{pr}(k)$ for some k. For sequence numbers n we shall speak of the 'exponents in n' to mean the exponents in the prime factorization of n.

$$\ell(n) = (\mu k \le n)(\mathrm{Seq}(n) \;\&\; \mathrm{pr}(k)|n \;\&\; \overline{\mathrm{pr}(k+1)|n}) \tag{2.4}$$

If n is a sequence number, $\ell(n)$ is the number of distinct prime factors of n. We call $\ell(n)$ the *length* of n. We included $\mathrm{Seq}(n)$ as a conjunct in this definition in order to secure that $\ell(n) = 0$ if n is not a sequence number.

We need an easy way of giving large bounds for several of the definitions below. Here is one way of doing this (there are others, of course): let $\mathrm{bg}(n)$ be the sequence number of length n in which each of the exponents is n. We leave to the reader the task of giving a primitive recursive definition of $\mathrm{bg}(n)$ (see the Exercises).

$$\begin{aligned} m * n = (\mu k \le \mathrm{bg}(m+n))\big(&\mathrm{Seq}(k) \;\&\; (\forall i \le \ell(m))([k]_i = [m]_i) \\ &\& \; (\forall i \le \ell(n))(i > 0 \to [k]_{i+\ell(m)} = [n]_i)\big) \end{aligned} \tag{2.5}$$

If m and n are sequence numbers, then $m*n$ is the sequence number k in which the exponents are first those in m and then those in n, and whose length is exactly $\ell(m) + \ell(n)$. (The definition puts constraints on $[k]_i$ for $1 \le i \le \ell(m) + \ell(n)$. Since $m * n$ is the least k fulfilling those constraints, it will not have any further prime factors.) Thus $30 * 30 = 30{,}030$, since $30 = 2^1 3^1 5^1$ and $30{,}030 = 2^1 3^1 5^1 7^1 11^1 13^1$, and $24 * 18 = 5880$, since $24 = 2^3 3^1$, $18 = 2^1 3^2$, and $5880 = 2^3 3^1 5^1 7^2$. If m is a sequence number but n is not, then $m*n = n*m = m$, since $\ell(n) = 0$, and similarly if n is a sequence number but m is not. If neither m nor n is a sequence number, then $m * n = 0$.

2.3 Arithmetization of syntax

In this section, we shall be defining functions and relations which, from a purely number-theoretical point of view, might seem somewhat unnatural. Although they are defined solely in terms of numbers, the *motivation* for these definitions stems from correspondences to the syntax of PA. We group these correspondences together as the *Mirroring Lemma*. A typical clause of this Lemma looks like this: if s and s'

are strings of signs of L_{PA} then $\gamma(s) * \gamma(s') = \gamma(s^\frown s')$. Thus the number-theoretic function $*$ mirrors the syntactic operation of concatenation. (This claim should be obvious, given the specification of the gödel numbering γ and the definition of the number-theoretic function $*$.) Each clause of the Mirroring Lemma asserts that a number-theoretic function or relation mirrors some syntactic notion.

$$\mathrm{paren}(n) = 2^9 * n * 2^{10}. \tag{2.6}$$

Thus $\mathrm{paren}(30) = 2^9 3^1 5^1 7^1 11^{10}$. Mirroring Lemma: Let s be a string of signs; then $\mathrm{paren}(\gamma(s)) = \gamma(\text{‘(’}^\frown s^\frown \text{‘)’})$, that is, $\mathrm{paren}(\gamma(s))$ is the gödel number of the string resulting by putting s in parentheses.

$$\mathrm{Var}(n) \leftrightarrow n \geq 17 \;\&\; n \text{ is odd} \tag{2.7}$$

$$\mathrm{Attm}(n) \leftrightarrow (\exists k \leq n)(n = 2^k \;\&\; (k = 1 \vee \mathrm{Var}(k))) \tag{2.8}$$

Mirroring Lemma: $\mathrm{Var}(n)$ iff $n = \Gamma(u)$ for some formal variable u. $\mathrm{Attm}(n)$ iff n is the gödel number of an *atomic term*, that is, ‘0’ or a formal variable.

$$\mathrm{succ}(n) = 2^2 * n \tag{2.9}$$

$$\mathrm{plus}(m, n) = \mathrm{paren}(m * 2^3 * n) \tag{2.10}$$

$$\mathrm{times}(m, n) = \mathrm{paren}(m * 2^4 * n) \tag{2.11}$$

$$\mathrm{Tmop}(i, j, k) \leftrightarrow k = \mathrm{succ}(i) \vee k = \mathrm{plus}(i, j) \vee k = \mathrm{times}(i, j) \tag{2.12}$$

Mirroring Lemma: Let s and s' be strings of signs; then $\mathrm{succ}(\gamma(s)) = \gamma(\text{‘}S\text{’}^\frown s)$, $\mathrm{plus}(\gamma(s), \gamma(s')) = \gamma(\text{‘(’}^\frown s^\frown \text{‘+’}^\frown s'^\frown \text{‘)’})$ and $\mathrm{times}(\gamma(s), \gamma(s')) = \gamma(\text{‘(}^\frown s^\frown \text{‘×’}^\frown s'^\frown \text{‘)’})$.

$$\mathrm{nmrl}(0) = 2; \mathrm{nmrl}(n + 1) = \mathrm{succ}(\mathrm{nmrl}(n)) \tag{2.13}$$

Thus $\mathrm{nmrl}(1) = 2^2 3^1 = 12$; $\mathrm{nmrl}(2) = 2^2 3^2 5^1 = 180$; $\mathrm{nmrl}(3) = 2^2 3^2 5^2 7^1 = 4500$. Mirroring Lemma: for every n, $\mathrm{nmrl}(n)$ is the gödel number of the formal numeral $\boldsymbol{n}$.

$$\begin{aligned}\mathrm{Tmseq}(n) \leftrightarrow \mathrm{Seq}(n) \;\&\; (\forall k \leq \ell(n))(k > 0 \rightarrow \mathrm{Attm}([n]_k) \quad \vee \\ (\exists i, j < k)\mathrm{Tmop}([n]_i, [n]_j, [n]_k))\end{aligned} \tag{2.14}$$

For example, Tmseq holds of $2^{2^{17}} 3^{2^1} 5^{2^9 3^{17} 5^3 7^1 11^{10}}$. Note that if $\mathrm{Tmseq}(n)$ holds then n must be a “second-order sequence number”, that is, a sequence number in which all the exponents are themselves sequence numbers. Mirroring Lemma: $\text{Tmseq}(n)$ holds iff n is a sequence number, and the exponents $[n]_1, [n]_2, \ldots [n]_{\ell(n)}$ in its prime factorization are the gödel numbers of a sequence $t_1, t_2, \ldots, t_{\ell(n)}$ of strings with the following property: each t_k either is an atomic term, or, for some strings t_i and t_j earlier in the sequence, is St_i or $(t_i + t_j)$ or $(t_i \times t_j)$. Such a sequence of strings shows how a term is built up from its constituent parts in accord with the formation rule for terms. Thus a string t is a term if and only if there is such a sequence whose last member is t. Hence, we define:

$$\text{Tm}(n) \leftrightarrow (\exists m \leq \text{bg}(n))(\text{Tmseq}(m) \ \& \ n = [m]_{\ell(m)}) \tag{2.15}$$

The bound on m is large enough. For suppose t is a term, and let i be the number of occurrences of the signs 'S', '$+$', and '$\times$' in t. Let $t_1 \ldots, t_j$ be a sequence of strings as in the preceding paragraph such that $t_j = t$ and j is as small as possible. Then $j \leq 2^i + 1$ (this may be proved by induction on j), and of course $2^i + 1 \leq \gamma(t)$. Also, each t_i is a subterm of t, so that $\gamma(t_i) \leq \gamma(t)$. Hence the sequence number m whose prime factorization has exponents $\gamma(t_1), \gamma(t_2), \ldots, \gamma(t_j)$ is at most $\text{bg}(\gamma(t))$. We may conclude that the following Mirroring Lemma clause holds: $\text{Tm}(n)$ iff $n = \gamma(t)$ for some term t of L_{PA}.

A similar series of definitions will yield a primitive recursive relation that mirrors the property of being a formula.

$$\text{Atform}(n) \leftrightarrow (\exists j, k \leq n)(\text{Tm}(j) \ \& \ \text{Tm}(k) \ \& \ n = j * 2^5 * k) \tag{2.16}$$

$$\text{neg}(n) = 2^6 * n \tag{2.17}$$

$$\text{cond}(m, n) = \text{paren}(m * 2^7 * n) \tag{2.18}$$

$$\text{gen}(k, n) = 2^8 * 2^k * \text{paren}(n) \tag{2.19}$$

$$\begin{aligned}\text{Formop}(i, j, k) \leftrightarrow k = \text{neg}(i) \vee k = \text{cond}(i, j) \vee \\ (\exists m \leq k)(\text{Var}(m) \ \& \ k = \text{gen}(m, i))\end{aligned} \tag{2.20}$$

$$\begin{aligned}\text{Formseq}(n) \leftrightarrow \text{Seq}(n) \ \& \ (\forall k \leq \ell(n))\big(k > 0 \rightarrow \text{Atform}([n]_k) \vee \\ (\exists i, j < k)\text{Formop}([n]_i, [n]_j, [n]_k)\big)\end{aligned} \tag{2.21}$$

$$\text{Form}(n) \leftrightarrow (\exists m \leq \text{bg}(n)(\text{Formseq}(m) \ \& \ n = [m]_{\ell(m)}) \tag{2.22}$$

The Mirroring Lemma clauses for (2.17) – (2.21) are left to the reader. By reasoning parallel to that for $\mathrm{Tm}(n)$, we can then infer that $\mathrm{Form}(n)$ holds iff $n = \gamma(F)$ for some formula F of L_{PA}.

Our next aim is to define a primitive recursive function that mirrors substitution of terms for free variables. To do this, we must first mirror the notions of bound and free occurrences of variables. Let the i^{th} place in a string be the address for the i^{th} sign in the string, counting from the left. Thus in '$\forall x((x+0)=x)$' the fifth place is the location of the occurrence of x that follows a left parenthesis, while the tenth place is the location of the rightmost occurrence of x.

$$\mathrm{Bound}(i,k,n) \leftrightarrow \mathrm{Form}(n) \;\&\; \mathrm{Var}(k) \;\&\; (\exists p,q,r \le n)(n = p * \mathrm{gen}(k,q) * r \;\& \\ \ell(p)+1 \le i \le \ell(p) + \ell(\mathrm{gen}(k,q))) \tag{2.23}$$

Mirroring Lemma: $\mathrm{Bound}(i,k,n)$ holds if $n = \gamma(F)$ for some formula F, $k = \Gamma(u)$ for some formal variable u, and the i^{th} place in F lies within the scope of a quantifier binding u. (Note that u need not occur at the i^{th} place. This feature of the definition will make it easier to mirror 't is free for u in F' (Exercise 2.?).)

$$\mathrm{Free}(i,k,n) \leftrightarrow \mathrm{Form}(n) \;\&\; \mathrm{Var}(k) \;\&\; [n]_i = k \;\&\; \overline{\mathrm{Bound}}(i,k,n) \tag{2.24}$$

Mirroring Lemma: $\mathrm{Free}(i,k,n)$ holds if $n = \gamma(F)$ for some formula F, $k = \Gamma(u)$ for some formal variable u, and u has a free occurrence at the i^{th} place in F.

In syntax, substitution of a term for a free variable u is the simultaneous replacement of all free occurrences of u by occurrences of the term. In order to mirror this by a primitive recursive function, we need to break it down into a step-by-step procedure of substituting for the free occurrences of the variable one by one. We will make the substitutions starting with the last free occurrence (the rightmost one) and continuing right-to-left. The reason for this is that if the term being substituted has length > 1, a substitution perturbs all the addresses to the right of the occurrence being replaced. By making the substitutions from right to left, at each step the addresses of the occurrences that are yet to be replaced remain unperturbed.

Define $(\max k \le m)R(k)$ as

$$(\mu k \le m)(R(k) \;\&\; (\forall j \le m)(j > k \to \overline{R}(j))).$$

Then $(\max k \le m)R(k)$ is the largest $k \le m$ such that $R(k)$ holds, and 0 if there is no such k.

$$\begin{aligned} \mathrm{occ}(0,k,n) &= (\max i \le \ell(n))\mathrm{Free}(i,k,n) \\ \mathrm{occ}(m+1,k,n) &= (\max i < \mathrm{occ}(m,k,n))\mathrm{Free}(i,k,n) \end{aligned} \tag{2.25}$$

If $n = \gamma(F)$ for some formula F and $k = \Gamma(u)$ for some formal variable u, then $\mathrm{occ}(0, k, n)$, $\mathrm{occ}(1, k, n)$, $\mathrm{occ}(2, k, n)$, ... give the addresses, from largest address down, of the places where u is free in F. If u has m free occurrences in F, then $\mathrm{occ}(i, k, n)$ is nonzero for $0 \leq i < m$, while $\mathrm{occ}(m, k, n) = 0$. Thus our next function gives the number of free occurrences of u in F.

$$\mathrm{nocc}(k, n) = (\mu m \leq \ell(n))(\mathrm{occ}(m, k, n) = 0) \tag{2.26}$$

$$\begin{aligned}\mathrm{subat}(n, p, i) = (\mu m \leq \mathrm{bg}(p + n))(\exists q, r \leq n)(n = q * 2^{[n]_i} * r \\ \& \ i = \ell(q) + 1 \ \& \ m = q * p * r)\end{aligned} \tag{2.27}$$

If n and p are sequence numbers and $0 < i \leq \ell(n)$, then $\mathrm{subat}(n, p, i)$ is the sequence number whose first $i - 1$ exponents match the first $i - 1$ exponents in n, whose next $\ell(p)$ exponents match those in p, and whose final $\ell(n) - i$ exponents match the final $\ell(n) - i$ exponents in n. Thus, for example, $\mathrm{subat}(30, 72, 2) = 2^1 * 2^3 3^2 * 2^1 = 9450$. Mirroring Lemma: if $n = \gamma(s)$ for some string s and $p = \gamma(s')$ for some string s', and i is at most the length of s, then $\mathrm{subat}(n, p, i)$ is the gödel number of the string obtained from s by substituting s' for whatever appears in s at the i^{th} place.

$$\begin{aligned}\mathrm{subst}(n, k, p, 0) &= n \\ \mathrm{subst}(n, k, p, i + 1) &= \mathrm{subat}(\mathrm{subst}(n, k, p, i), p, \mathrm{occ}(i, k, n))\end{aligned} \tag{2.28}$$

Mirroring Lemma: if $n = \gamma(F)$ for some formula F, $k = \Gamma(u)$ for some formal variable u, and $p = \gamma(s)$ for some string s, then $\mathrm{subst}(n, k, p, 1)$ is the gödel number of the result of substituting s for the rightmost free occurrence of u in F; $\mathrm{subst}(n, k, p, 2)$ is the result of substituting s for the two rightmost free occurrences of v in F; and so on.

$$\mathrm{sub}(n, k, p) = \mathrm{subst}(n, k, p, \mathrm{nocc}(k, n)) \tag{2.29}$$

Mirroring Lemma: if $n = \gamma(F(u))$ for some formula $F(u)$, $k = \Gamma(u)$ for some formal variable u, and $p = \gamma(t)$ for some term t, then $\mathrm{sub}(k, p, n) = \gamma(F(t))$.

The next function will be of great importance in the proofs of this Chapter and the next, although it will might look a little mysterious now.

$$\mathrm{diag}(n) = \mathrm{sub}(n, 19, \mathrm{nmrl}(n)). \tag{2.30}$$

So if $n = 2^{19}3^5 5^1$, then $\mathrm{diag}(n) = 2^2 3^2 5^2 7^2 \ldots \mathrm{pr}(n)^2 \mathrm{pr}(n + 1)^1 \mathrm{pr}(n + 2)^5 \mathrm{pr}(n + 3)^1$. Mirroring Lemma: if $n = \gamma(F(y))$ for some formula $F(y)$, then $\mathrm{diag}(n) =$

$\gamma(F(\boldsymbol{n}))$, that is, $\mathrm{diag}(n)$ is the gödel number of the formula obtained from $F(y)$ by substituting $\boldsymbol{n}$ for all free occurrences of y. (So if there are no free occurrences of y in $F(y)$, then $\mathrm{diag}(n) = n$. Also, if n is not the gödel number of a formula, then $\mathrm{diag}(n) = n$.) For reasons that will become clear only later, diag is called the *gödel diagonal function.*

Our next task is to define a primitive recursive relation that mirrors the property of being an axiom.

$$\mathrm{T1Ax}(n) \leftrightarrow \mathrm{Form}(n) \mathbin{\&} (\exists k, m \leq n)(n = \mathrm{cond}(k, \mathrm{cond}(m, k))) \tag{2.31}$$

Mirroring Lemma: $\mathrm{T1Ax}(n)$ iff $n = \gamma(F)$ for a formula F of L_{PA} that is an axiom generated from schema (T1).

We leave to the reader the task of providing primitive recursive definitions that mirror the other axioms. (See the Exercises.) These will culminate in a primitive recursive definition of a 1-place relation $\mathrm{Ax}(n)$ that yields the Mirroring Lemma clause: $\mathrm{Ax}(n)$ iff $n = \gamma(F)$ for some formula F that is an axiom of PA.

$$\mathrm{Infop}(i, j, k) \leftrightarrow j = \mathit{cond}(i, k) \vee (\exists p \leq k)(\mathrm{Var}(p) \mathbin{\&} k = \mathrm{gen}(p, i)) \tag{2.32}$$

$$\begin{aligned}\mathrm{Drvtn}(n) \leftrightarrow \mathrm{Seq}(n) \mathbin{\&} (\forall k < \ell(n))\big(k > 0 \rightarrow \mathrm{Ax}([n]_k) \vee \\ (\exists i, j < k)\mathrm{Infop}([n]_i, [n]_j, [n]_k)\big)\end{aligned} \tag{2.33}$$

Mirroring Lemma: $\mathrm{Drvtn}(n)$ holds iff n is a sequence number, and the exponents $[n]_1, [n]_2, \ldots [n]_{\ell(n)}$ in its prime factorization are the gödel numbers of a sequence of formulas that is a derivation in PA. If this holds, we say that n *encodes* the derivation.

$$\mathrm{Der}(m, n) \leftrightarrow \mathrm{Drvtn}(m) \mathbin{\&} n = [m]_{\ell(m)} \tag{2.34}$$

Mirroring Lemma: $\mathrm{Der}(m, n)$ holds iff m encodes a derivation in PA of a formula with gödel number n.

2.4 Numeralwise representability

We now define a syntactic notion which gives a sense in which number-theoretic functions and relations may be formalized within PA, and so establishes a link between the informal number theory of the previous two sections and the formal system. For vividness, we'll frame the definition for 2-place relations. Let R be such a relation. A formula $F(x, y)$ of L_{PA}, whose only free variables are x and y, *numeralwise represents* the relation R iff for all integers k and n, if $R(k, n)$ then $\vdash F(\boldsymbol{k}, \boldsymbol{n})$

if $\overline{R}(k, n)$ then $\vdash {\sim}F(\boldsymbol{k}, \boldsymbol{n})$.

Similarly, a 1-place relation (that is, a set) is numeralwise represented by a formula $F(x)$, whose only free variable is x, iff $\vdash F(\boldsymbol{n})$ whenever the relation holds of n (whenever n is in the set) and $\vdash {\sim}F(\boldsymbol{n})$ whenever it doesn't. A 3-place relation would be numeralwise represented by a formula $F(x, y, z)$ with just those three free variables; if the relation holds of a triple then the corresponding numerical instance of $F(x, y, z)$ is derivable and if it doesn't then the corresponding numerical instance is refutable. A relation is *numeralwise representable* iff there is a formula that numeralwise represents it.

The formula $x = y$ numeralwise represents the identity relation. To show this we must show, for any integers k and n, if k and n are identical then $\vdash \boldsymbol{k} = \boldsymbol{n}$, and if k and n are distinct then $\vdash {\sim}\boldsymbol{k} = \boldsymbol{n}$. If k and n are identical, then $\boldsymbol{k}$ and $\boldsymbol{n}$ are the same formal numeral, so that $\boldsymbol{k} = \boldsymbol{n}$ is the same formula as $\boldsymbol{k} = \boldsymbol{k}$. And since $x = x$ is an axiom of PA, it follows that $\vdash \boldsymbol{k} = \boldsymbol{k}$.

Now suppose k and n are distinct. We illustrate the argument by an example. Suppose $k = 4$ and $n = 2$. Axiom (N2) is $\vdash Sx = Sy \supset x = y$. Hence, $\vdash SSSS0 = SS0 \supset SSS0 = S0$ and $\vdash SSS0 = S0 \supset SS0 = 0$, so that by truth-functional logic $\vdash SSSS0 = SS0 \supset SS0 = 0$. But by axiom (N1) $\vdash {\sim}SS0 = 0$. Hence, by truth-functional logic, $\vdash {\sim}SSSS0 = SS0$, that is, $\vdash {\sim}\mathbf{4} = \mathbf{2}$. It should be clear how to generalize this argument to show $\vdash {\sim}\boldsymbol{k} = \boldsymbol{n}$ whenever $k > n$. For the case $k < n$, we may then invoke the symmetry of '='.

We also want to formulate a notion of numeralwise representation for functions. Here, we want PA to capture the fact that a function has a unique value for each argument.

Let φ be an 2-place function from numbers to numbers. A formula $F(x, y, z)$ *numeralwise represents* the function φ (in PA) iff for all numbers m,n, and q such that $\varphi(m, n) = q$

$$\vdash F(\boldsymbol{m}, \boldsymbol{n}, z) \equiv z = \boldsymbol{q}$$

Note that since $\vdash \boldsymbol{q} = \boldsymbol{q}$, the condition yields $\vdash F(\boldsymbol{m}, \boldsymbol{n}, \boldsymbol{q})$; indeed the condition is equivalent to $\vdash F(\boldsymbol{m}, \boldsymbol{n}, \boldsymbol{q}) \centerdot (F(\boldsymbol{m}, \boldsymbol{n}, z) \supset z = \boldsymbol{q})$. The condition formalizes the claims tht q is the value of φ for arguments m and n, and that q is the only value. For a 1-place function φ, the condition would read: for all n and q such that $\varphi(n) = q$, $\vdash F(\boldsymbol{n}, y) \equiv y = \boldsymbol{q}$. We leave to the reader the task of giving the general form of the condition; in each case the formula that numeralwise represents the function has one more free variable than the function has arguments.

The formula $x + y = z$ numeralwise represents addition. In §1.5 we've shown that, for any k, n, and q, if q is the sum of k and n then $\vdash \boldsymbol{k} + \boldsymbol{n} = \boldsymbol{q}$. We must also show that if q is the sum of k and n then $\vdash \boldsymbol{k} + \boldsymbol{n} = z \supset z = \boldsymbol{q}$. By the transitivity of identity, $\vdash z = \boldsymbol{k} + \boldsymbol{n} \,.\, \boldsymbol{k} + \boldsymbol{n} = \boldsymbol{q} \supset z = \boldsymbol{q}$. Since $\vdash \boldsymbol{k} + \boldsymbol{n} = \boldsymbol{q}$, by truth-functional logic and the symmetry of identity we have $\vdash \boldsymbol{k} + \boldsymbol{n} = z \supset z = \boldsymbol{q}$. A similar argument shows that the formula $x \times y = z$ numeralwise represents multiplication.

Numeralwise representation is, in one sense, a weak constraint on a formalization of a relation or function. It requires only that the formalization of "pointwise facts" about the relation or function be derivable in PA: for all particular arguments, whether the relation holds or not, and for all particular arguments, what the value of the function is and that it is the only value. For other purposes we might well want to require more, for example, that formalizations of general laws that the relation or the function obeys be derivable in PA. We might not take the formula $x + y = z$ to be a good formalization of addition unless, say, the commutative law were derivable using it. However, numeralwise representation is all that is needed for the First Incompleteness Theorem.

Representability Theorem: Every primitive recursive relation and function is numeralwise representable in PA.

We put off the proof until Chapter 4. As we shall see, it is straightforward, amounting primarily to verifying that manipulations of finite sequences of numbers can be formalized in PA. For now we note only that the proof is entirely syntactic and is *constructive*: it provides a recipe for constructing, given any primitive recursive definition of a function or a relation, a formula that numeralwise represents that function or relation.

2.5 Proof of incompleteness

As a last preliminary, we define the notion of ω-consistency. PA is ω-*consistent* iff for no formula $F(x)$ are all the following derivable: $\sim\forall x F(x)$ as well as the numerical instances $F(\boldsymbol{n})$ for every n. Since $\sim\forall x F(x)$ is logically equivalent to $\exists x \sim F(x)$, we can rephrase the definition thus: if $\vdash F(\boldsymbol{n})$ for every n, then $\exists x \sim F(x)$ is not derivable. Thus, ω-consistency requires that if all numerical instances of a formula are derivable then the formula expressing the existence of a counterexample cannot derivable. If we take $F(x)$ to be $\sim H(x)$ and cancel the double negation, then we obtain another way of phrasing the condition: if $\exists x H(x)$ is derivable, then not all of $\sim H(\mathbf{0})$, $\sim H(\mathbf{1})$, $\sim H(\mathbf{2})$, ... are derivable. That is, if it can be derived that there is a number with a certain property, it cannot be derived that each particular number fails to have the property. Note that ω-consistency is a syntactic property, although somewhat more complex than consistency. Note too that ω-consistency implies consistency: for if the system is inconsistent every formula is derivable, so the system is ω-inconsistent. As we shall see, consistency does not imply ω-consistency: it is possible for a system to be consistent but ω-inconsistent. But clearly we would want to require PA to be ω-consistent.

Let us list the results of previous sections that will be used in the proof. From the previous section we need the Representability Theorem. From the arithmetization of syntax we need just two results: the existence of a primitive recursive relation $\mathrm{Der}(k, n)$ such that $\mathrm{Der}(k, n)$ holds iff n is the gödel number of a formula and k encodes a derivation of that formula; and the existence of a primitive recursive function $\mathrm{diag}(n)$ such that if n is the gödel number of a formula $F(y)$ then $\mathrm{diag}(n)$ is the gödel number of $F(\boldsymbol{n})$. From the logical material in Chapter 1, we need only the fact that if a universal quantification $\forall x F(x)$ is derivable, then so are all its numerical instances $F(\boldsymbol{k})$. This of course follows from the instantiation axiom (Q1) and modus ponens.

Now let Q be the 2-place relation defined thus: for all integers k and n,

$$Q(k, n) \leftrightarrow \overline{\mathrm{Der}}(k, \mathrm{diag}(n))$$

Then Q is primitive recursive;. By the Representability Theorem, there is a formula $A(x, y)$ that numeralwise represents Q. That is, for all k and n,

(a) if $Q(k, n)$ then $\vdash A(\boldsymbol{k}, \boldsymbol{n})$;

(b) if $\overline{Q}(k, n)$ then $\vdash \sim A(\boldsymbol{k}, \boldsymbol{n})$.

Let p be the gödel number of $\forall x A(x, y)$. Then, from the property of $\mathrm{diag}(n)$ just noted, $\mathrm{diag}(p)$ is the gödel number of $\forall x A(x, \boldsymbol{p})$. Note that $\forall x A(x, \boldsymbol{p})$ is a sentence, that is, contains no free variables. We can now complete the proof in five steps.

(1) If $\vdash \forall x A(x, \boldsymbol{p})$ then, for each k, $\vdash A(\boldsymbol{k}, \boldsymbol{p})$.

This is clear, since each $A(\boldsymbol{k}, \boldsymbol{p})$ is an instance of $\forall x A(x, \boldsymbol{p})$.

(2) If $\vdash \forall x A(x, \boldsymbol{p})$ then there exists a number k such that $\vdash \sim A(\boldsymbol{k}, \boldsymbol{p})$.

For suppose $\vdash \forall x A(x, \boldsymbol{p})$. Then there is a number k that encodes a derivation of $\forall x A(x, \boldsymbol{p})$, so that $\mathrm{Der}(k, q)$, where q is the gödel number of $\forall x A(x, \boldsymbol{p})$. As noted above, $q = \mathrm{diag}(p)$. Hence $\mathrm{Der}(k, \mathrm{diag}(p))$, that is, $\overline{Q}(k, p)$. But then, by (b) above, $\vdash \sim A(\boldsymbol{k}, \boldsymbol{p})$

(3) If PA is consistent then $\forall x A(x, \boldsymbol{p})$ is not derivable.

This follows from (1) and (2), since together they imply that if $\vdash \forall x A(x, \boldsymbol{p})$ then PA is inconsistent.

(4) If PA is consistent then, for each k, $\vdash A(\boldsymbol{k}, \boldsymbol{p})$.

For suppose PA consistent. By (3), $\forall x A(x, \boldsymbol{p})$ is not derivable. Hence there is no number that encodes a derivation of it. By Mirroring, for each k, $\overline{\text{Der}}(k, \text{diag}(p))$. That is, for each k, $Q(k, p)$. By (a) above, for each k, $\vdash A(\boldsymbol{k}, \boldsymbol{p})$.

(Of course, if PA is inconsistent then also $\vdash A(\boldsymbol{k}, \boldsymbol{p})$ for each k, since everything is derivable, but we do not need this fact.)

(5) If PA is ω-consistent, then $\sim\forall x A(x, \boldsymbol{p})$ is not derivable.

Suppose PA is ω-consistent. Then PA is consistent. By (4), $A(\boldsymbol{k}, \boldsymbol{p})$ is derivable for each integer k. So if $\sim\forall x A(x, \boldsymbol{p})$ were also derivable, PA would be ω-inconsistent.

> **Gödel's First Incompleteness Theorem** If PA is ω-consistent then there is a sentence G of L_{PA} such that neither G nor $\sim G$ is derivable in PA, and hence PA is syntactically incomplete.

This follows from (3) and (5), taking G to be the sentence $\forall x A(x, \boldsymbol{p})$. This sentence is often called the *Gödel sentence*.

The core of the foregoing proof is step (2). Traditionally, to show a fact of the form "if F is derivable then so is H" we show that H can be derived from F, or from a formula obtained from F by universal generalization. Indeed, step (1) has precisely this character (other examples can be found in the Exercises for §1.2). But this is not at all what is going on in step (2). Rather, the supposition that $\vdash \forall x A(x, \boldsymbol{p})$ is exploited as a metalinguistic fact; this fact is then mirrored as a number theoretic fact (namely, that there is a number k such that $\text{Der}(k, q)$, where q is the gödel number of $\forall x A(x, \boldsymbol{p})$); and that number-theoretic fact is then formalized in the system, by means of numeralwise representation. In a somewhat loose way of speaking, we might say that we are not drawing an inference from the content $\forall x A(x, \boldsymbol{p})$ might be taken to express, but rather from the fact of its derivability.

The formula $A(x, y)$ was so chosen that for any formula $F(y)$ with gödel number n, if $\vdash F(\boldsymbol{n})$ then, for some k, $\vdash \sim A(\boldsymbol{k}, \boldsymbol{n})$. That is, since the gödel number of $F(\boldsymbol{n})$ is $\text{diag}(n)$, $\vdash F(\boldsymbol{n})$ tells us that there exists a number k such that $\text{Der}(k, \text{diag}(n))$, so that $\vdash \sim A(\boldsymbol{k}, \boldsymbol{n})$ by numeralwise representation. Obtaining step (2) is a matter only of choosing the right formula $F(y)$. Namely, we choose $F(y)$ to be $\forall x A(x, y)$. Call its gödel number p. Thus we obtain the result that if $\vdash \forall x A(x, \boldsymbol{p})$ then there exists a k such that $\vdash \sim A(\boldsymbol{k}, \boldsymbol{p})$.

The formula $\forall x A(x, y)$ formalizes a number-theoretic property. Since $A(x, y)$ numeralwise represents the relation $\overline{\mathrm{Der}}(k, \mathrm{diag}(n))$, we could say that $\forall x A(x, y)$ formalizes the one-place relation that holds of n iff for no k do we have $\mathrm{Der}(k, \mathrm{diag}(n))$; and by mirroring this holds iff the formula with gödel number $\mathrm{diag}(n)$ is not derivable. Now $\forall x A(x, y)$ itself has a gödel number, namely, p. So the formula $\forall x A(x, \boldsymbol{p})$ formalizes the statement that this one-place relation holds of p, which mirrors the statement that the formula with gödel number $\mathrm{diag}(p)$ is not derivable. But $\mathrm{diag}(p)$ is the gödel number of that very formula $\forall x A(x, \boldsymbol{p})$! That is, the Gödel sentence formalizes a number-theoretic condition that mirrors this syntactic claim: the Gödel sentence is not derivable. It is no wonder that the Gödel sentence is not derivable: for if it were, the claim it formalizes would not be correct, and there would be a derivation of it, which would be encoded by a number k such that $\vdash \sim A(\boldsymbol{k}, \boldsymbol{n})$, producing an inconsistency in PA.

That the Gödel sentence formalizes a number-theoretic fact that mirrors the claim that the Gödel sentence is underivable is often expressed in picturesque terms thus: the Gödel sentence asserts that it is not underivable. Or even more vividly, "the Gödel sentence says 'I am not derivable'." Formulas don't talk, of course, so this is figurative language. Unfortunately, it can also cause confusion, so it is important to see exactly how to put the point precisely and unfiguratively. We do this in the next section.

Step (4) also merits comment, since it expresses an important phenomenon that Gödel's proof brings to light. What is shown here (assuming PA consistent) is that a formula $F(x)$ can be derived whenever a formal numeral is substituted for x; but the universal quantification $\forall x F(x)$ cannot be derived. Loosely speaking: the property that $F(x)$ expresses can be derived to hold of each particular number, but "every number possesses the property" cannot be formally derived. For what is shown above is that for each k, the formalization of $\overline{\mathrm{Der}}(k, \mathrm{diag}(p))$ is derivable, but the formalization of $(\forall k)\overline{\mathrm{Der}}(k, \mathrm{diag}(p))$ is not derivable.

2.6 'I am not derivable'

The most direct way of making precise sense of the claim that the Gödel sentence asserts its own underivability is to adopt a semantical stance and use the notion of the truth of formulas in the intended interpretation of L_{PA}. All uses of "truth" below are to be understood in this sense. First note that the formula $A(\boldsymbol{k}, \boldsymbol{n})$ is true iff $\overline{\mathrm{Der}}(k, \mathrm{diag}(n))$: this follows from the soundness of PA for the intended interpretation and the choice of $A(x, y)$ as numeralwise representing that relation.

(The recourse to soundness at this point can actually be avoided by inspection of the proof of the Representability Theorem. That $A(\boldsymbol{k}, \boldsymbol{n})$ is true iff $\overline{\mathrm{Der}}(k, \mathrm{diag}(n))$ will follow directly from the construction of the formula $A(x, y)$. See §4.3.) Since the universe of the intended interpretation is the natural numbers, for each n, $\forall x A(x, \boldsymbol{n})$ is true iff $\overline{\mathrm{Der}}(k, \mathrm{diag}(n))$ for every natural number k. By mirroring, this condition obtains iff the formula with gödel number $\mathrm{diag}(n)$ is not derivable. Now let p be the gödel number of $\forall x A(x, y)$; then $\forall x A(x, \boldsymbol{p})$ is true iff the formula with gödel number $\mathrm{diag}(p)$ is not derivable. Since the formula with gödel number $\mathrm{diag}(p)$ is $\forall x A(x, \boldsymbol{p})$, this shows

(†) $\forall x A(x, \boldsymbol{p})$ is true iff $\forall x A(x, \boldsymbol{p})$ is not derivable.

That is, the condition for the truth of the Gödel sentence is a number-theoretic fact that mirrors the underivability of the Gödel sentence. Note that mirroring is essential here. In the most direct sense, the G̈del sentence asserts a number-theoretic statement; it is true iff a certain number-theoretic condition holds. It is only using mirroring that we obtain the biconditional between the truth of the Gödel sentence and a syntactic condition.

From (†) and the soundness of PA a quick semantical argument for incompleteness can be formulated. G cannot be derivable, since if it is then by (†) it is false, which would violate soundness. So G is not derivable, and hence it is true. Hence $\sim G$ is false, and so by soundness it cannot be derivable. Gödel's First Incompleteness Theorem is often stated semantically thus: there is a true sentence of L_{PA} that is not derivable in PA. (As we shall see in §3.2, though, the syntactic proof of the previous section yields an important further result unobtainable from the semantic proof.)

There are two ingredients to obtaining (†). First is gödelization, that is, the arithmetization of syntax, which shows that syntactic notions can be captured by formulas of L_{PA}, by formalizing the number-theoretic notions that mirror the syntactic ones. Second is the use of the function $\mathrm{diag}(n)$, which allows the construction of a formula that can appear on both sides of the biconditional (†).

Gödel's strategy can be viewed this way. Define "formula m at n" as: the result of substituting $\boldsymbol{n}$ for any free occurrences of y in the formula with gödel number m (if m is not the gödel number of a formula, let formula m at n be an arbitrary object that is not a formula). By gödelization, the relation "formula m at n is not derivable" can be captured by a formula of L_{PA}: all we need do is formalize the number-theoretic relation $(\forall k)\overline{\mathrm{Der}}(k, \mathrm{sub}(m, 19, \mathrm{nmrl}(n)))$. Now identify the two variables, that is, consider only the case when $m = n$. (In the plane, these pairs form the diagonal; hence the use of "diag".) So we have a formula $F(y)$ with one free variable that captures the 1-place relation "formula n at n is not derivable", that is, for each n, $F(\boldsymbol{n})$ is true iff formula n at n is not derivable. Let p be the gödel number of $F(y)$. We then have $F(\boldsymbol{p})$ is true if formula p at p is not derivable; and formula p at p is just $F(\boldsymbol{p})$. Thus we obtain (†).

So far in this section we have been using the notion of truth in the intended interpretation for formulas of L_{PA}. However, there is a way of capturing (†) syntactically, namely, by formalizing it within L_{PA}. The right hand side of (†) expresses a syntactic property, which is formalizable via gödelization in a direct way. Let $D(x, y)$ be a formula that numeralwise represents the relation $\mathrm{Der}(k, n)$. Then if H is a formula with gödel number n, the formula $\exists x D(x, \boldsymbol{n})$ is a formalization of the assertion that $(\exists k)\mathrm{Der}(k, n)$, which mirrors the assertion that H is derivable. Thus the underivabiity of the Gödel sentence can be formalized by $\sim\exists x D(x, \boldsymbol{q})$, where q is the gödel number of the Gödel sentence. The left hand side of (†) is the ascription of truth to the Gödel sentence, but inside L_{PA} this can be formalized by the Gödel sentence itself, since the metalinguistic ascription of truth to a sentence can be captured in the object language by the assertion of that sentence. We claim that the resulting formalization is derivable in PA, that is,

$$(\ddagger) \vdash \forall x A(x, \boldsymbol{p}) \equiv \sim\exists x D(x, \boldsymbol{q})$$

To show this, we need to go into more detail about the construction of $A(x, y)$. That task will occupy us in at the beginning of Chapter 3. Moreover we shall see that, just as (†) yields a concise semantic proof of the First Incompleteness Theorem, (‡) provides a quick way of reformulating the syntactic proof of the preceding section.

Chapter 3

Formalized Metamathematics

3.1 The Fixed Point Lemma

As we pointed out in the preceding section, Gödel's proof can be analyzed into two central components, gödelization and the use of the diagonal function. The contributions each component makes to the proof can be highlighted if we look more closely at how the Gödel sentence may be constructed. In the proof, we used $\forall x A(x, \boldsymbol{p})$ for the Gödel sentence, where $A(x, y)$ numeralwise represents the relation $\overline{\mathrm{Der}}(k, \mathrm{diag}(n))$. It is natural to think of such a formula as built up from numeralwise representations of Der and diag. So let $D(x, y)$ be a formula of L_{PA} that numeralwise represents the relation $\mathrm{Der}(k, n)$ and let $\Delta(y, z)$ be a formula that numeralwise represents the function $\mathrm{diag}(n)$. Then let $A(x, y)$ be the formula

$$\forall z(\Delta(y, z) \supset {\sim}D(x, z))$$

Now let p be the gödel number of $\forall x A(x, y)$ and let $q = \mathrm{diag}(p)$. $\forall x A(x, \boldsymbol{p})$ is the Gödel sentence; its gödel number is $\mathrm{diag}(p)$, that is, q. Since $\Delta(y, z)$ numeralwise represents diag, we have

$$\vdash \Delta(\boldsymbol{p}, z) \equiv z = \boldsymbol{q}$$

Hence the formula $A(x, \boldsymbol{p})$ is provably equivalent to

$$\forall z(z = \boldsymbol{q} \supset {\sim}D(x, z))$$

which by the laws of identity is equivalent to ${\sim}D(x, \boldsymbol{q})$. From the equivalence of $A(x, \boldsymbol{p})$ and ${\sim}D(x, \boldsymbol{q})$ we may infer the equivalence of $\forall x A(x, \boldsymbol{p})$ and $\forall x \sim D(x, \boldsymbol{q})$, and the latter formula is of course equivalent to $\sim\exists x D(x, \boldsymbol{q})$. Thus we have shown

$$(\ddagger) \vdash \forall x A(x, \boldsymbol{p}) \equiv \sim\exists x D(x, \boldsymbol{q})$$

Once we have (‡), we need pay no further attention to the internal structure of the Gödel sentence or to the use of diag. The rest of the work is done by gödelization, in particular, that the number-theoretic relation $\mathrm{Der}(k, n)$ mirrors the syntactic relation 'derivation of' and that the formula $D(x, y)$ numeralwise represents $\mathrm{Der}(k, n)$. We can then formulate a quick proof of Gödel's First Theorem. First, suppose $\vdash \forall x A(x, \boldsymbol{p})$. By mirroring, $\mathrm{Der}(k, q)$ for some k, so that by numeralwise representation, $\vdash D(\boldsymbol{k}, \boldsymbol{q})$ for some k, whence $\vdash \exists x D(x, \boldsymbol{q})$. But from $\vdash \forall x A(x, \boldsymbol{p})$, (‡) and truth-functional logic we also have $\vdash \sim\exists x D(x, \boldsymbol{q})$. Hence PA is inconsistent. Second, suppose $\vdash \sim\forall x A(x, \boldsymbol{p})$. If PA is consistent, $\forall x A(x, \boldsymbol{p})$ is not derivable, so by mirroring $\overline{\mathrm{Der}}(k, q)$ for each k; so by numeralwise representation $\vdash \sim D(\boldsymbol{k}, \boldsymbol{q})$ for each k. But, from $\vdash \sim\forall x A(x, \boldsymbol{p})$, (‡), and truth-functional logic we have $\vdash \exists x D(x, \boldsymbol{q})$. Hence PA is ω-inconsistent.

The method just used to construct the Gödel sentence can be applied more generally, so as to yield the following theorem, sometimes also called the "Diagonal Lemma".

Fixed Point Lemma. Let $F(y)$ be any formula. Then there is a formula H such that $\vdash H \equiv F(\boldsymbol{m})$, where m is the Gödel number of H.

Proof. Let $\Delta(x, y)$ numeralwise represent the function diag, let $F'(y)$ be $\forall z(\Delta(y, z) \supset F(z))$, let k be the Gödel number of $F'(y)$, and let H be $F'(\boldsymbol{k})$. Let m be the gödel number of H; then $m = \mathrm{sub}(k, 19, \mathrm{nmrl}(k)) = \mathrm{diag}(k)$, so that

$$\vdash \Delta(\boldsymbol{k}, z) \equiv z = \boldsymbol{m}$$

From this it follows that $F'(\boldsymbol{k})$ is provably equivalent to $\forall z(z = \boldsymbol{m} \supset F(z))$, and hence to $F(\boldsymbol{m})$. $\square$

The formula $F(y)$ may contain free variables aside from y. If it does not, that is, if y is the only free variable in $F(y)$ then H will be a sentence; and if it does contain other free variables then the free variables of H will be precisely those other variables.

A graphic way of stating the Lemma is possible with some new notation: if F is a formula, then let $\ulcorner F \urcorner$ be the formal numeral for the Gödel number of F; that is, $\ulcorner F \urcorner$ is $\boldsymbol{k}$, where $k = \gamma(F)$. Thus $\ulcorner\,\urcorner$ is a function that carries formulas to formal numerals. Its interaction with the number-theoretic function nmrl(n) and the gödel numbering γ is given by the following diagram:

$$\begin{tikzcd}
F \arrow[r, "\gamma"] \arrow[d, "\ulcorner\,\urcorner"'] & \gamma(F) \arrow[d, "\text{nmrl}"] \\
\ulcorner F \urcorner \arrow[r, "\gamma"'] & \text{nmrl}(\gamma(F))
\end{tikzcd}$$

That is, for each formula F, $\gamma(\ulcorner F \urcorner) = \text{nmrl}(\gamma(F))$. The Fixed Point Lemma then says: for every formula $F(y)$ there is a formula H such that

$$\vdash H \equiv F(\ulcorner H \urcorner).$$

Any such formula H is called a *fixed point* of $F(y)$.

It should be noted that there are other fixed points of a formula $F(y)$ aside from that constructed in the proof of the Fixed Point Lemma. In fact, there are infinitely many distinct ones. This can be seen as follows. Let J be any sentence derivable in PA, and let H' be the fixed point constructed as in the proof but for the formula $F(y) \centerdot J$. Then $\vdash H' \equiv F(\ulcorner H' \urcorner) \centerdot J$. Since J is derivable, it follows by truth-functional logic that $\vdash H' \equiv F(\ulcorner H' \urcorner)$, so that H' is a fixed point of $F(y)$. Moreover H' is distinct from H, since H is $\forall z(\Delta(\boldsymbol{k}, z) \supset F(z))$ and H' is $\forall z(\Delta(\boldsymbol{k'}, z) \supset F(z)) \centerdot J$, where $k \neq k'$. Clearly, using a different J in this construction will yield yet a different H'; hence we can generate infinitely many different ones.

Nor should it be thought that these different fixed points will necessarily be equivalent. The original fixed point H is derivably equivalent to $F(\ulcorner H \urcorner)$; the new fixed point H' is derivably equivalent to $F(\ulcorner H' \urcorner)$. As these formulas ascribe $F(y)$ to different objects (different Gödel numbers), they say different things. (Your saying 'I'm hungry' is not equivalent to my saying 'I'm hungry'.)

One can even *insure* that certain different fixed points are inequivalent. Let $F(y)$ numeralwise represent the primitive recursive relation that holds of a number n iff it is the Gödel number of a formula of which $0 = 0$ is not a subformula. There will be such an $F(y)$ that itself does not contain $0 = 0$ as a subformula (if it did, we could always replace the subformula $0 = 0$ by any other derivable formula), and, similarly, there will be a formula $\Delta(y, z)$ that numeralwise represents the function diag(n) that does not contain $0 = 0$ as a subformula. Hence the fixed point H

constructed in the proof of the Fixed Point Theorem will not contain $0 = 0$ as a subformula, so that $\vdash F(\ulcorner H \urcorner)$, and hence $\vdash H$. On the other hand, if we let J be $0 = 0$ and carry out the construction of two paragraphs back, then H' does contain $0 = 0$ as a subformula, so that $\vdash {\sim}F(\ulcorner H' \urcorner)$, and hence $\vdash {\sim}H'$. Thus H and H' are not equivalent.

The quick proof of Gödel's Theorem given just before the Fixed Point Lemma can be viewed as an application of the Lemma to the formula ${\sim}\exists x D(x, y)$, where $D(x, y)$ numeralwise represents the relation $\mathrm{Der}(n, k)$. Let us define a 1-place number-theoretic relation $\mathrm{Dvbl}(n)$ as $(\exists k)\mathrm{Der}(k, n)$ (note this is not a primitive recursive definition, since there is no bound on the quantifier). This relation mirrors derivability, that is, $\mathrm{Dvbl}(n)$ holds iff n is the gödel number of a formula derivable in PA. Since $D(x, y)$ numeralwise represents Der, we can think of the formula $\exists x D(x, y)$ as formalizing $\mathrm{Dvbl}(n)$, and so, by mirroring, as being a formal expression of derivability. Thus Gödel's Theorem can be obtained by applying the Fixed Point Lemma to a formal expression of underivability. (This analysis of Gödel's proof, along with the formulation of the Fixed Point Lemma, was first given by Rudolf Carnap in 1934.) The quick proof can be further streamlined so as to highlight the needed properties of the expression of derivability. Let $\mathrm{Prov}(y)$ be a formula meant as such a formal expression, and suppose it obeys the following two conditions:

- **Adequacy:** For any formula F, if $\vdash F$ then $\vdash \mathrm{Prov}(\ulcorner F \urcorner)$.

- **Faithfulness:** For any formula F, if $\vdash \mathrm{Prov}(\ulcorner F \urcorner)$ then $\vdash F$.

Now let G be a fixed point of ${\sim}\mathrm{Prov}(y)$. To show G not derivable in PA, if PA is consistent, we need just the Adequacy condition: since $\vdash G \equiv {\sim}\mathrm{Prov}(\ulcorner G \urcorner)$, if $\vdash G$, then $\vdash {\sim}\mathrm{Prov}(\ulcorner G \urcorner)$ by truth-functional logic, but also $\vdash \mathrm{Prov}(\ulcorner G \urcorner)$ by Adequacy, so that PA would be inconsistent. Moreover, we can show ${\sim}G$ not derivable in PA, if PA is consistent, using just Faithfulness: if $\vdash {\sim}G$, then $\vdash \mathrm{Prov}(\ulcorner G \urcorner)$ by truth-functional logic, whence $\vdash G$ by Faithfulness, so that, again, PA would be inconsistent.

At this point the acute reader should be wondering what happened to the hypothesis of ω-consistency. The answer is that it figures in the proof that there exists a formula $\mathrm{Prov}(y)$ fulfilling the two conditions. That is, we can show that if $D(x, y)$ numeralwise represents the relation Der, then the formula $(\exists x)D(x, y)$ obeys the Adequacy Condition and—provided that PA is ω-consistent—also the Faithfulness Condition. For suppose $\vdash F$. By Mirroring, there exists a k such that $\mathrm{Der}(k, \gamma(F))$. By numeralwise representation, for such a $k \vdash D(\boldsymbol{k}, \ulcorner F \urcorner)$. By existential generalization, $\vdash \exists x D(x, \ulcorner F \urcorner)$. Thus Adequacy is fulfilled. Now suppose $\vdash \exists x D(x, \ulcorner F \urcorner)$. We show that if not $\vdash F$ then PA is ω-inconsistent. If not $\vdash F$, then $\text{Der}(k, \gamma(F))$ for no k, in which case, by numeralwise representation, $\vdash \sim D(\boldsymbol{k}, \ulcorner F \urcorner)$ for every k. This together with $\vdash \exists x D(x, \ulcorner F \urcorner)$ is an ω-inconsistency. Hence, if PA is ω-consistent then $\exists x D(x, y)$ fulfills the Faithfulness condition.

Now, if $\text{Prov}(y)$ numeralwise represented the relation $\text{Dvbl}(n)$, then consistency alone would be enough to insure Faithfulness: for if not $\vdash F$, then $\vdash \sim\text{Prov}(\ulcorner F \urcorner)$ by numeralwise representation, whence not $\vdash \text{Prov}(\ulcorner F \urcorner)$ by consistency. But in fact, we cannot require that $\text{Prov}(y)$ numeralwise represent $\text{Dvbl}(n)$, since if PA is consistent then $\text{Dvbl}(n)$ is not numeralwise representable! (This claim follows from a straightforward application of the Fixed Point Theorem. See the Exercises.) Note that this shows that the relation $\text{Drvbl}(n)$ is not primitive recursive.

Caution. The Adequacy Condition does not imply that $F \supset \text{Prov}(\ulcorner F \urcorner)$ is derivable. Indeed, assuming that PA is ω-consistent, there are formulas F such that $F \supset \text{Prov}(\ulcorner F \urcorner)$ is not derivable. Similarly, the Faithfulness Condition does not imply that $\text{Prov}(\ulcorner F \urcorner) \supset F$ is derivable. If PA is consistent, there are formulas F such that $\text{Prov}(\ulcorner F \urcorner) \supset F$ is not derivable. (See the Exercises.)

3.2 Gödel's Second Incompleteness Theorem

Gödel's Second Theorem states that if PA is consistent then the consistency of PA is not derivable in PA. Now the consistency of PA is a metamathematical statement. What does it mean to say that this metamathematical statement is or is not derivable? Of course we mean that a formal expression of this statement is or is not derivable. And by a formal expression we mean a formula that formalizes whatever number-theoretic fact mirrors the consistency of PA. That is, by mirroring, PA is consistent iff the class of Dvbl-numbers has such-and-such a property, which is number-theoretic in nature. The formal expression of consistency is obtained by formalizing the such-and-such property.

Recall that to prove Gödel's First Theorem one proves that if PA is consistent then G is not derivable, where G is the Gödel sentence. Number-theoretically speaking, this is a proof that if the class of Dvbl-numbers has such-and-such a property then q is not a Dvbl-number, where q is the Gödel number of G. If we formalize this number-theoretic assertion we get the formula

$$(\text{Con}) \supset \sim\text{Prov}(\boldsymbol{q}),$$

where (Con) is the formal expression of consistency and $\text{Prov}(y)$ is the formal expression of derivability. Indeed, by formalizing the proof of the number-theoretic assertion—a proof which, as Gödel emphasized, is purely number-theoretic and uses only simple arithmetical principles—we obtain the derivability in PA of this formula. But we saw above that

$$\vdash G \equiv {\sim}\mathrm{Prov}(\boldsymbol{q}).$$

Hence

$$\vdash (\mathrm{Con}) \supset G.$$

We may conclude that if (Con) were derivable in PA, then G could be derivable in PA. But then, by the First Theorem, PA would be inconsistent.

Thus the heart of the proof of Gödel's Second Incompleteness Theorem is to show that a formalization of Gödel's First Incompleteness Theorem is derivable in PA. To fill in the details of this outline, we first have to fix on a formal expression of consistency. There are many candidates. PA is consistent iff for no formula F are F and ${\sim}F$ both derivable. Hence one could formalize consistency by formalizing the assertion: for no number n do we have both $\mathrm{Dvbl}(n)$ and $\mathrm{Dvbl}(\mathrm{neg}(n))$. Alternatively, PA is consistent iff there exists an underivable formula; hence one could formalize the assertion: there exists an n such that $\mathrm{Form}(n)$ and $\overline{\mathrm{Dvbl}}(n)$. The essential thing about the formal expression of consistency is that it function formally (that is, within PA) in an appropriate way in relation to the formal representation of derivability. What this comes to is that if $\mathrm{Prov}(y)$ is the representation of derivability, then the representation (Con) of consistency should fulfill the following condition:

($\sharp$) For each formula F, $\vdash (\mathrm{Con}) \supset {\sim}(\mathrm{Prov}(\ulcorner F \urcorner) \,\boldsymbol{.}\, \mathrm{Prov}(\ulcorner {\sim}F \urcorner))$

As we shall see, this condition can be secured in a pleasingly simple way.

The formalization of the argument for Gödel's First Theorem also requires the derivability in PA of certain formulas involving $\mathrm{Prov}(y)$. For example, since PA includes modus ponens we have: if $F \supset G$ is derivable , then if F is derivable so is G. (Number-theoretically, if $\mathrm{Dvbl}(\mathrm{cond}(m, n))$, then if $\mathrm{Dvbl}(m)$ then $\mathrm{Dvbl}(n)$.)We need to be able to derive the formalization of this fact in PA. Thus we should like to have the derivability in PA of the formula $\mathrm{Prov}(\ulcorner F \supset G \urcorner) \supset (\mathrm{Prov}(\ulcorner F \urcorner) \supset \mathrm{Prov}(\ulcorner G \urcorner))$ for all formulas F and G. The derivability of such formulas is not assured by the Adequacy Condition. We are thus led to consider what formal properties a formula ought to have in order that the formula be a suitable formal expression of the notion of derivability. Once we make this explicit, we shall be in a position to give a perspicuous proof of Gödel's Second Theorem.

We call a formula Prov(y) a *standard provability predicate* iff it obeys the following conditions:

Adequacy. For each formula F, if $\vdash F$ then $\vdash \text{Prov}(\ulcorner F \urcorner)$.

Formal Modus Ponens. For all formulas F and G,

$$\vdash \text{Prov}(\ulcorner F \supset G \urcorner) \supset (\text{Prov}(\ulcorner F \urcorner) \supset \text{Prov}(\ulcorner G \urcorner)).$$

Formal Adequacy. For each formula F,

$$\vdash \text{Prov}(\ulcorner F \urcorner) \supset \text{Prov}(\ulcorner \text{Prov}(\ulcorner F \urcorner) \urcorner).$$

The condition of Formal Adequacy amounts to the derivability in PA of a formal expression of the fact that Prov(y) obeys Adequacy. For Adequacy states that if F is derivable then so is the formula $\text{Prov}(\ulcorner F \urcorner)$. If the notion 'is derivable' is formalized by Prov(y), the formalization of Adequacy is $\text{Prov}(\ulcorner F \urcorner) \supset \text{Prov}(\ulcorner \text{Prov}(\ulcorner F \urcorner) \urcorner)$.

Note. Not every standard provability predicate can be thought of as expressing provability. E.g., any formula that numeralwise represents the relation Form will fulfill the three conditions. The natural formalization of derivability, however, is what we have in mind in talking of a standard provability predicate. This natural formalization is $\exists x D(x, y)$, where $D(x, y)$ numeralwise represents the relation Der. As we saw at the end of §3.1, the Adequacy Condition follows at once. Formal Modus Ponens requires that $D(x, y)$ be a "good" formalization of the relation Der in a sense that goes beyond numeralwise representation. To be specific, it requires that $D(x, y)$ can be used in a formal derivation of the elementary number-theoretic fact that, for any i and j and any formulas F and G, if $\text{Der}(i, \gamma(F \supset G))$ and $\text{Der}(j, \gamma(F))$, then there exists an integer k such that $\text{Der}(k, \gamma(G))$ (in fact one can take $k = i * j * 2^{\gamma(G)}$). Finally, to obtain Formal Adequacy we would have to be able to use $D(x, y)$ in a formalization of the proof of Adequacy, and so in a formalization of the fact that $D(x, y)$ numeralwise represents the relation Der. Although there are many details here, the arithmetical reasoning needed is elementary (as can be seen from the proof of the Representability Theorem in Chapter 4), and poses no difficulty for formalization.

We do not include Faithfulness among the conditions for a standard provability predicate because then the existence of a standard provability predicate would require a hypothesis like ω-consistency; whereas a formula Prov(y) fulfilling the three conditions above can be shown to exist without any consistency-like suppositions.

One consequence of Adequacy and Formal Modus Ponens will be used often enough below to merit special mention:

If $\vdash F \supset G$ then $\vdash \text{Prov}(\ulcorner F \urcorner) \supset \text{Prov}(\ulcorner G \urcorner)$.

For if $\vdash F \supset G$ then $\vdash \text{Prov}(\ulcorner F \supset G \urcorner)$ by Adequacy. By Formal Modus Ponens and modus ponens we obtain $\vdash \text{Prov}(\ulcorner F \urcorner) \supset \text{Prov}(\ulcorner G \urcorner)$. In what follows, we shall call this principle 'Quick FMP'.

Having considered what conditions a formal representation of derivability should satisfy, let us now return to the question of a formal representation of the assertion that PA is consistent. If $\text{Prov}(y)$ is a standard provability predicate, we may use it to formulate a simple formal expression of consistency. Namely, let (Con) be the formula $\sim\text{Prov}(\ulcorner S0 = 0 \urcorner)$. It is straightforward to show that (Con) obeys condition ($\sharp$). For let F be any formula. We have

$$\vdash F \supset (\sim F \supset S0 = 0)$$

by truth-functional logic. Applying Quick FMP we obtain

$$\vdash \text{Prov}(\ulcorner F \urcorner) \supset \text{Prov}(\ulcorner \sim F \supset S0 = 0 \urcorner)$$

From Formal Modus Ponens and truth-functional logic we have

$$\vdash \text{Prov}(\ulcorner F \urcorner) \supset (\text{Prov}(\ulcorner \sim F \urcorner) \supset \text{Prov}(\ulcorner S0 = 0 \urcorner))$$

so by truth-functionally equivalence we then have

$$\vdash \text{Prov}(\ulcorner F \urcorner) \,\boldsymbol{.}\, \text{Prov}(\ulcorner \sim F \urcorner) \supset \text{Prov}(\ulcorner S0 = 0 \urcorner)$$

Condition ($\sharp$) is just the contrapositive of this formula.

The formula (Con) also has the following amusing property: Let F be any formula; then $\vdash \sim\text{Prov}(\ulcorner F \urcorner) \supset \text{(Con)}$. (See the Exercises.) The amusing property gives further support to the notion that (Con) is an appropriate formalization of the statement that PA is consistent, since it shows that the formalization of the elementary fact that if there is an underivable formula then PA is consistent, is itself derivable.

Fully elaborated, then, Gödel's Second Incompleteness Theorem reads as follows:

Gödel's Second Incompleteness Theorem Let (Con) be a sentence of L_{PA} that fulfills condition ($\sharp$), where $\text{Prov}(y)$ is a standard provability predicate. Then if PA is consistent, (Con) is not derivable in PA.

Proof. Let G be any fixed point of $\sim$Prov(y). Thus $\vdash G \equiv \sim\text{Prov}(\ulcorner G \urcorner)$. As we saw in the previous section, it follows from the Adequacy of Prov(y) that if PA is consistent then G is not derivable. We also have:

(1)	$\vdash \text{Prov}(\ulcorner G \urcorner) \supset \sim G$	by the specification of G
(2)	$\vdash \text{Prov}(\ulcorner \text{Prov}(\ulcorner G \urcorner) \urcorner) \supset \text{Prov}(\ulcorner \sim G \urcorner)$	by Quick FMP
(3)	$\vdash \text{Prov}(\ulcorner G \urcorner) \supset \text{Prov}(\ulcorner \text{Prov}(\ulcorner G \urcorner) \urcorner)$	by Formal Adequacy
(4)	$\vdash \text{Prov}(\ulcorner G \urcorner) \supset \text{Prov}(\ulcorner \sim G \urcorner)$	by t-f logic, from (2) and (3)
(5)	$\vdash \text{Prov}(\ulcorner G \urcorner) \supset \sim(\text{Con})$	by t-f logic, from (4) and ($\sharp$)
(6)	$\vdash (\text{Con}) \supset \sim\text{Prov}(\ulcorner G \urcorner)$	by t-f logic, from (5)
(7)	$\vdash (\text{Con}) \supset G$	by t-f logic, from (6) and the specification of G.

From (7) it follows that if PA is consistent then (Con) is not derivable. For if (Con) were derivable then, by modus ponens, G would be derivable, and so PA would be inconsistent. □

At the end of the previous section we showed the following: let Prov(y) fulfill Adequacy, and let G be a fixed point of $\sim$Prov(y); then if G is derivable, PA is inconsistent. The foregoing proof of Gödel's Second Theorem is a direct formalization of the proof we gave.

Gödel's Second Theorem differs from the First Theorem in the subtlety of what it says. For much hinges on what we take to be a formal statement of the consistency of the system. We have required (and used in the proof) that the formal consistency statement have property ($\sharp$), and that Prov(y) be a standard provability predicate.

A simple example shows that some such subtlety is needed. Let Prov$'(y)$ be the formula $\text{Prov}(y) \mathbin{.} y \neq \ulcorner S0 = 0 \urcorner$, where Prov$(y)$ is a standard provability predicate. Then Prov$'(y)$ fulfills the Adequacy Condition. For if $\vdash F$, then $\vdash \text{Prov}(\ulcorner F \urcorner)$, and moreover either F is not the formula $S0 = 0$, in which case $\vdash \sim\ulcorner F \urcorner = \ulcorner S0 = 0 \urcorner$, whence $\vdash \text{Prov}'(\ulcorner F \urcorner)$, or else F is the formula $S0 = 0$, in which case PA is inconsistent, whence again $\vdash \text{Prov}'(\ulcorner F \urcorner)$. But if we take (Con) to be $\sim\text{Prov}'(\ulcorner S0 = 0 \urcorner)$, then clearly (Con) is derivable in PA. The hitch here, of course, is that this formula (Con) does not have property ($\sharp$). This shows that Prov$'(y)$ is not standard: in fact, it fails to fulfill Formal Modus Ponens (although it does fulfill Formal Adequacy).

There are more complicated examples: we can construct formulas (Con) that do possess property ($\sharp$), but such that the predicate Prov(y) mentioned in ($\sharp$) is not standard (lacks, say, Formal Adequacy); and such that (Con) is derivable in PA. Of course, we want to say that any such formula (Con) is not, in any intuitive sense, a formal statement of the consistency of PA, because the predicate Prov(y) is not a good formalization of derivability. As has been pointed out, Formal Modus Ponens and Formal Adequacy are simply the formalizations of assertions about derivability which can be shown by elementary metamathematical arguments. And for a predicate to qualify as a good formalization of derivability, it must be usable in derivations that formalize elementary metamathematical arguments about derivability.

3.3 The First Incompleteness Theorem Sharpened

Gödel's proof of incompleteness requires the hypothesis of ω-consistency. In the most streamlined rendition of this proof, as we saw in Section 3.1 above, ω-consistency is used to show the faithfulness of a provability predicate; and faithfulness is used to show the underivability of the negation of the Gödel sentence. In 1936 J. Barkley Rosser sharpened Gödel's result by showing how to prove incompleteness using only consistency, not ω-consistency, as a hypothesis. Rosser's basic idea was to replace the Faithfulness Condition with another one, which we shall call the Rosser Condition. There are two desiderata: a provability predicate that fulfills the Adequacy and Rosser Conditions should be usable in a fixed-point argument to establish the incompleteness of PA (relying only on consistency); and the existence of such a provability predicate should be demonstrable without recourse to ω-consistency.

> **Rosser's Lemma.** Suppose a formula Rov(y) fulfills the following two conditions for all formulas F:
>
> **Adequacy.** If $\vdash F$ then $\vdash \mathrm{Rov}(\ulcorner F \urcorner)$
>
> **Rosser.** If $\vdash {\sim}F$ then $\vdash {\sim}\mathrm{Rov}(\ulcorner F \urcorner)$.
>
> Let R be a fixed point of ${\sim}\mathrm{Rov}(y)$. If PA is consistent, then neither R nor ${\sim}R$ is derivable.

Proof. Suppose $\vdash R$; by Adequacy, $\vdash \mathrm{Rov}(\ulcorner R \urcorner)$; since $\vdash R \equiv {\sim}\mathrm{Rov}(\ulcorner R \urcorner)$, $\vdash {\sim}R$; thus PA is inconsistent. Suppose $\vdash {\sim}R$; by Rosser, $\vdash {\sim}\mathrm{Rov}(\ulcorner R \urcorner)$; since $\vdash R \equiv {\sim}\mathrm{Rov}(\ulcorner R \urcorner)$, $\vdash R$; thus PA is inconsistent. $\Box$

Note that no formula Rov(y) that fulfills the Rosser Condition can be a standard provability predicate. For that condition yields $\vdash {\sim}\mathrm{Rov}(\ulcorner S0 = 0 \urcorner)$.

To prove the Theorem it now suffices to show the existence of a suitable formula Rov(y). Here Rosser came up with a most ingenious and fertile idea. A *refutation* of a formula F is a derivation of ${\sim}F$. We formalize the following notion, which we dub 'rovability': a formula F is rovable iff there is a derivation of F but no smaller refutation of F (where 'smaller' means simply: with smaller gödel number). Formalizations of the following heuristic arguments will then yield the two conditions, as we shall see. First, suppose F is derivable; thus there is a derivation of F. Check all the (finitely many) smaller derivations that there are, and you'll see (provided PA is consistent) that none is a refutation of F. Hence F is rovable. Second, suppose $\sim F$ is derivable; thus there is a refutation of F. No smaller derivation is one of F (provided PA is consistent), and this can be checked. Hence if there is some derivation of F, then there is a smaller refutation of F (namely, the given one). Hence F is not rovable.

To formalize this we shall need two properties of the formula $x \leqslant y$ defined in §1.6:

(a) for every integer k, $\vdash x \leqslant \boldsymbol{k} \vee \boldsymbol{k} \leqslant x$;

(b) for each formula $F(x)$ and each integer k, if $\vdash F(\boldsymbol{j})$ for $j = 0, 1, ..., k$, then $\vdash x \leqslant \boldsymbol{k} \supset F(x)$.

Property (a) follows by instantiation from the last law given in §1.6. For property (b), note first that from $F(\boldsymbol{j})$ is logically equivalent to $x = \boldsymbol{j} \supset F(x)$. Hence property (b) will follow if we have, for each number k, $\vdash x \leqslant \boldsymbol{k} \supset x = 0 \vee x = \mathbf{1} \vee ... \vee x = \boldsymbol{k}$. The proof of this is left to the reader. (See the Exercises.)

We may now define the formula $\text{Rov}(y)$. Let Ref be the two-place primitive recursive relation defined by $\text{Ref}(k, n) \leftrightarrow \text{Der}(k, \text{neg}(n))$, and let $D(x, y)$ and $\text{R}(x, y)$ numeralwise represent the relations Der and Ref, respectively. Let $B(x, y)$ be

$$D(x, y) \,.\, \forall z(z \leqslant x \supset \sim R(z, y))$$

and, finally, let $\text{Rov}(y)$ be $(\exists x)B(x, y)$.

(1) $\text{Rov}(y)$ fulfills the Adequacy Condition.

For suppose $\vdash F$. Then for some k, $\text{Der}(k, \gamma(F))$, so that $\vdash D(\boldsymbol{k}, \ulcorner F \urcorner)$. Now,if PA is consistent, then $\overline{\text{Ref}}(j, \gamma(F))$ for each $j \leq k$. Hence $\vdash \sim R(\boldsymbol{j}, \ulcorner F \urcorner)$ for $j = 0, 1, ..., k$. By (b) above and universal generalization, we then have $\vdash \forall z(z \leqslant \boldsymbol{k} \supset \sim R(z, \ulcorner F \urcorner))$. Thus

$$\vdash D(\boldsymbol{k}, \ulcorner F \urcorner) \,.\, \forall z(z \leqslant \boldsymbol{k} \supset \sim R(z, \ulcorner F \urcorner))$$

from which $\vdash \text{Rov}(\ulcorner F \urcorner)$ follows by existential generalization. If, on the other hand, PA is inconsistent, then every formula is derivable, so in particular $\vdash \text{Rov}(\ulcorner F \urcorner)$.

(2) $\text{Rov}(y)$ fulfills the Rosser Condition.

For suppose $\vdash \sim F$. Thus there is a k such that $\mathrm{Ref}(k, \gamma(F))$, so that $\vdash R(\boldsymbol{k}, \ulcorner F \urcorner)$. Hence $\vdash \boldsymbol{k} \leqslant x \supset (\exists z)(z \leqslant x \,.\, R(z, \ulcorner F \urcorner))$, or, equivalently,

$$\vdash \boldsymbol{k} \leqslant x \supset \sim\forall z(z \leq x \supset \sim R(z, \ulcorner F \urcorner))$$

Now if PA is consistent then $\overline{\mathrm{Der}}(j, \gamma(F))$ for each $j \leq k$. Hence $\vdash \sim D(\boldsymbol{j}, \ulcorner F \urcorner)$ for $j = 0, 1, ..., k$. By (b) above,

$$\vdash x \leqslant \boldsymbol{k} \supset \sim D(x, \ulcorner F \urcorner)$$

By (a) above and modus ponens, we obtain

$$\vdash \sim D(x, \ulcorner F \urcorner) \vee \sim\forall z(z \leqslant x \supset \sim R(z, \ulcorner F \urcorner))$$

that is, $\vdash \sim[D(x, \ulcorner F \urcorner) \,.\, \forall z(z \leqslant x \supset \sim R(z, \ulcorner F \urcorner))]$. Hence, by universal generalization, $\vdash \forall x \sim [D(x, \ulcorner F \urcorner) \,.\, \forall z(z \leqslant x \supset \sim R(z, \ulcorner F \urcorner))]$. That is, $\vdash \sim\mathrm{Rov}(\ulcorner F \urcorner)$.

Rosser's Lemma, (1) and (2) then yield

Rosser's Sharpened Incompleteness Theorem. If PA is consistent then PA is not syntactically complete.

Rosser's Theorem has several interesting consequences. It can be used to show, on the assumption that PA is consistent, the existence of infinitely many logically inequivalent formally undecidable sentences in PA. (A sentence F is formally undecidable in PA iff F is neither derivable nor refutable in PA.) The argument goes thus. Assume PA consistent. Let R_1 be the Rosser sentence for PA; that is, the fixed point of $\sim\mathrm{Rov}(y)$. Thus R_1 is formally undecidable in PA. Now let PA_1 be the formal system obtained from PA by adding the sentence R_1 as a new axiom. Then PA_1 is consistent (else $\sim R_1$ would be derivable in PA, so PA itself would be inconsistent). We may repeat Rosser's argument for PA_1: that is, let R_2 be a fixed point of $\sim\mathrm{Rov}_1(y)$, where $\mathrm{Rov}_1(y)$ is a rovability predicate for the formal system PA_1. Then R_2 is formally undecidable in PA_1. A fortiori, R_2 is formally undecidable in PA. Now let PA_2 be the formal system obtained from PA_1 by adding R_2 as a new axiom. Again, PA_2 is consistent. And the Rosser sentence R_3 for the formal system PA_2 is formally undecidable in PA_2; a fortiori it is formally undecidable in PA. And so on.

This argument would not work if all we had was Gödel's original Theorem, and used Gödel sentences instead of Rosser sentences—even if we assume the ω-consistency of PA. For to carry the argument through we would need to require that PA_1 is ω-consistent; and this does not follow from the ω-consistency of PA. That is, the above argument exploits the fact that consistency is "inherited" in passing from PA_i to PA_{i+1}. Any analogous argument using Gödel's Theorem would have to show that ω-consistency is inherited; and this is not so easy. (We could show this if we adopt a semantical position and assume PA is sound. For the fact that the Gödel sentence is true in the intended interpretation allows us to conclude that the system like PA but including the Gödel sentence as a new axiom is also sound, and hence ω-consistent.)

3.4 Löb's Theorem

In 1952 Leon Henkin asked the following question: What about the formula that "says" "I am provable"? That is, let $\mathrm{Prov}(y)$ be a standard provability predicate, and suppose H is a fixed point of $\mathrm{Prov}(y)$, so that $\vdash H \equiv \mathrm{Prov}(\ulcorner H \urcorner)$. Is this formula H derivable? No heuristic indicates an answer: for if H is derivable then what H "says" is true, so there is no conflict with soundness; and if H is not derivable, then what H "says" is false, so again there is no conflict. But in 1955 M.H. Löb settled the matter by an ingenious, and not straightforwardly intuitive, proof.

> **Löb's Theorem.** Let F be a formula such that $\vdash \mathrm{Prov}(\ulcorner F \urcorner) \supset F$. Then $\vdash F$.

Henkin's question is thereby answered: the formula that "says" "I am provable" is indeed provable. For if $\vdash H \equiv \mathrm{Prov}(\ulcorner H \urcorner)$, then by truth-functional logic $\vdash \mathrm{Prov}(\ulcorner H \urcorner) \supset H$, so by Löb's Theorem, $\vdash H$. The interest of Löb's Theorem goes further. As we shall see, it can be exploited to yield much new information about derivability.

Now the formula $\mathrm{Prov}(\ulcorner F \urcorner) \supset F$ can be construed as a formalization (or as close to it as we can get) of the assertion that if F is derivable then F is true. Thus it is a formalization of something we would like to believe, namely soundness. Löb thus shows that soundness for a formula F is derivable only if F is derivable. Conversely, if $\vdash F$ then, by truth-functional logic, also $\vdash \mathrm{Prov}(\ulcorner F \urcorner) \supset F$. Thus we get the derivability of $\mathrm{Prov}(\ulcorner F \urcorner) \supset F$, trivially, only in the case F is itself derivable. As Rohit Parikh quipped, "PA could'nt be more modest about its own veracity."

Before presenting Löb's proof, we give an informal heuristic, formulated by Henkin after he read Löb's proof. The heuristic argument uses the notion of truth freely—that is what leads to the paradoxical nature of the conclusion. The actual proof of Löb's Theorem is more or less what is obtained by formalizing the heuristic, after 'is true' is replaced by 'is derivable'. Let Leon be the sentence: if Leon is true then all students are above average. Suppose that Leon is true; so if Leon is true then all students are above average; so by modus ponens, all students are above average. In the preceding sentence we have shown, on the supposition that Leon is true, that all students are above average That is, we have shown: If Leon is true, then all students are above average. Thus we have shown Leon is true. And then, by modus ponens, all students are above average. Paradox!

The rigorous proof of Löb's Theorem follows. Let F be a formula such that $\vdash \text{Prov}(\ulcorner F \urcorner) \supset F$. We wish to show that $\vdash F$. Let H be a fixed point of the formula $\text{Prov}(y) \supset F$. That is,

$$\vdash H \equiv (\text{Prov}(\ulcorner H \urcorner) \supset F).$$

The argument that follows contains several truth-functional steps; to highlight the truth-functional forms involved, we shall abbreviate the formula $\text{Prov}(\ulcorner H \urcorner) \supset F$ by J and the formula $\text{Prov}(\ulcorner \text{Prov}(\ulcorner H \urcorner) \urcorner)$ by K.

(1)	$\vdash H \supset J$	by the specification of H
(2)	$\vdash \text{Prov}(\ulcorner H \urcorner) \supset \text{Prov}(\ulcorner J \urcorner)$	by Quick FMP
(3)	$\vdash \text{Prov}(\ulcorner J \urcorner) \supset (K \supset \text{Prov}\ulcorner F \urcorner)$	by Formal Modus Ponens
(4)	$\vdash \text{Prov}(\ulcorner H \urcorner) \supset K$	by Formal Adequacy
(5)	$\vdash \text{Prov}(\ulcorner H \urcorner) \supset \text{Prov}(\ulcorner F \urcorner)$	by t-f logic, from (2), (3) and (4)
(6)	$\vdash \text{Prov}(\ulcorner F \urcorner) \supset F$	by supposition
(7)	$\vdash \text{Prov}(\ulcorner H \urcorner) \supset F$	by t-f logic, from (5) and (6)
(8)	$\vdash H$	by (7) and the specification of H
(9)	$\vdash \text{Prov}(\ulcorner H \urcorner)$	by Adequacy
(10)	$\vdash F$	t-f logic, from (7) and (9). □

Löb's Theorem has many applications, and can be used to show both derivabilities and underivabilities. For example, as was first pointed out by Georg Kreisel, Gödel's Second Theorem is an easy corollary of Löb's. For suppose $\vdash$ (Con). Recall that (Con) is $\sim\text{Prov}(S0 = 0)$. By truth-functional logic, $\vdash \text{Prov}(S0 = 0) \supset H$ for any formula H; in particular, $\vdash \text{Prov}(S0 = 0) \supset S0 = 0$. And then, by Löb's Theorem, $\vdash S0 = 0$, so that PA is inconsistent. Another example concerns the Rosser sentence R. In the Exercises, we have seen that $\vdash G \supset \text{Con}$, where G is the Gödel sentence; using Löb's Theorem, we can show the same does not hold of R, if PA is consistent. For suppose $\vdash R \supset$ (Con). As we also saw in the Exercises, $\vdash$ (Con) $\supset \sim\text{Prov}(\ulcorner \sim R \urcorner)$. Hence, by truth-functional logic, $\vdash R \supset \sim\text{Prov}(\ulcorner \sim R \urcorner)$.

By contraposition, $\vdash$ Prov($\ulcorner{\sim}R\urcorner$) $\supset {\sim}R$. By Löb's Theorem $\vdash {\sim}R$. Hence, by Rosser's Theorem, PA is inconsistent.

A simple example of using Löb's Theorem to show a derivability is the following: suppose A is a formula such that $\vdash A \supset$ (Prov($\ulcorner F\urcorner$) $\supset F$) for every F; then A is refutable. For the hypothesis implies that $\vdash A \supset$ (Prov($\ulcorner{\sim}A\urcorner$) $\supset {\sim}A$). This truth-functionally implies $\vdash$ Prov($\ulcorner{\sim}A\urcorner$) $\supset {\sim}A$, whence $\vdash {\sim}A$.

A more elaborate example is this: suppose H and F are formulas such that $\vdash$ Prov($\ulcorner H\urcorner$) $\supset$ (Prov($\ulcorner F\urcorner$) $\supset F$); then $\vdash$ Prov($\ulcorner H\urcorner$) $\supset F$. For let K be Prov($\ulcorner H\urcorner$) $\supset F$. By Formal Modus Ponens, $\vdash$ Prov($\ulcorner K\urcorner$) **.** Prov($\ulcorner$Prov($\ulcorner H\urcorner$)$\urcorner$) $\supset$ Prov($\ulcorner F\urcorner$). Since Prov($\ulcorner H\urcorner$) $\supset$ Prov($\ulcorner$Prov($\ulcorner H\urcorner$)$\urcorner$) and Prov($\ulcorner H\urcorner$) $\supset$ (Prov($\ulcorner F\urcorner$) $\supset F$) are both derivable, by truth-functional logic $\vdash$ Prov($\ulcorner K\urcorner$) $\supset K$, so that K is derivable. This result may be applied to show that the formalized version of Löb's Theorem is derivable, that is, for each formula F

$$\vdash \text{Prov}(\ulcorner \text{Prov}(\ulcorner F\urcorner) \supset F\urcorner) \supset \text{Prov}(\ulcorner F\urcorner)$$

Since Formal Modus Ponens gives us

$$\vdash \text{Prov}(\ulcorner \text{Prov}(\ulcorner F\urcorner) \supset F\urcorner) \supset (\text{Prov}(\ulcorner \text{Prov}(\ulcorner F\urcorner)\urcorner) \supset \text{Prov}\ulcorner F\urcorner)$$

we can obtain the desired result by applying the result of the preceding paragraph.

Other applications of Löb's Theorem are contained in the exercises. Here we offer one more: we use Löb's Theorem to show one way in which ω-consistency is a rather more intricate notion than simple consistency. Define a sequence $A_0, A_1, \ldots$ of formulas thus: A_0 is $S0 = 0$; A_1 is Prov($\ulcorner S0 = 0\urcorner$); A_2 is Prov($\ulcorner$Prov($\ulcorner S0 = 0\urcorner$)$\urcorner$); and, in general, A_{i+1} is Prov($\ulcorner A_i\urcorner$). We assume Prov(y) has the form $\exists x D(x, y)$. (Note that ${\sim}A_1$ is just the formula (Con).)

(1) $\vdash A_i \supset A_{i+1}$ for each i.

For $i = 0$ this follows since $\vdash {\sim}S0 = 0$. For $i > 0$ this follows from Formal Adequacy.

(2) If PA is consistent then no A_i for $i > 0$ is refutable in PA.

Were ${\sim}A_i$ derivable in PA for $i > 0$, then by (1), ${\sim}A_1$ would be derivable in PA. That is, $\vdash$ (Con). But then, by Gödel's Second Theorem, PA would be inconsistent.

(3) If PA is ω-consistent then no A_i is derivable in PA.

The proof is by induction on i. For $i = 0$ this follows just from the consistency of PA. Suppose A_i is not derivable. Then, by numeralwise expressibility, $\vdash {\sim}D(\boldsymbol{n}, \ulcorner A_i\urcorner)$ for each number n. But A_{i+1} is just $(\exists x)D(x, \ulcorner A_i\urcorner)$. Hence, were A_{i+1} derivable, PA would be ω-inconsistent.

(4) The system obtained by adding any A_i as a new axiom to PA is ω-inconsistent.

This is immediate from the proof of (3).

(5) If PA is ω-consistent then $A_{i+1} \supset A_i$ is not derivable for any i.
This follows at once from (3) and Löb's Theorem.

We have shown the existence of a sequence of formulas, each of which is strictly weaker than the one that precedes it (in the sense that it is provably implied by the one that precedes it, but does not provably imply it) but each of which is ω-inconsistent with PA. Contrast the case of simple consistency: if F and G are each inconsistent with PA, then F and G are both refutable in PA, and hence $\vdash F \equiv G$.

Chapter 4

Formalizing Primitive Recursion

4.1 Δ_0, Σ_1, and Π_1 formulas

In this chapter we show that all primitive recursive functions and relations are numeralwise representable in PA. In fact we show that the numeralwise representations may be taken to be formulas of a restricted syntactic form. This will have the consequence that the sentences that have been of interest to us in the preceding two chapters, like the Gödel sentence, the Rosser sentence, and (Con), may also be taken to be of restricted syntactic forms.

The Δ_0-*formulas of* L_{PA} are those in which all quantifiers are bounded. More precisely: every atomic formula is a Δ_0-formula; if F and G are Δ_0-formulas, then so are $\sim F$ and $F \supset G$; if F is a Δ_0-formula, v a variable, and t a term not containing v, then $\forall v(v \leqslant t \supset F)$ is a Δ_0-formula.

(Recall that L_{PA} does not contain the existential quantifier; we have used $\exists v$ as metamathematical shorthand for $\sim\forall v\sim$. So Δ_0-formulas can express bounded existential quantifiers, since $\exists v(v \leqslant t \mathbin{.} F)$ is shorthand for $\sim\forall v \sim (v \leqslant t \mathbin{.} F)$, which is logically equivalent to $\sim\forall v(v \leqslant t \supset \sim F)$.)

The *numerical value* of a closed term, that is, a term without variables, is defined in the obvious way: the numerical value of 0 is 0, the numerical value of St is one more than the numerical value of t, and the numerical values of terms $s + t$ and $s \times t$ are the sum and the product, respectively, of the numerical values of s and of t. Now the property of truth for Δ_0-sentences may be defined by induction on the construction of the sentence: an atomic sentence $s = t$ is true iff the numerical value of s is equal to the numerical value of t; $F \supset G$ is true iff F is false or G is true; $\sim F$ is true iff F is not true; and $\forall v(v \leqslant t \supset F(v))$ is true iff $F(0), F(\mathbf{1}), \ldots, F(\boldsymbol{k})$

are all true, where k is the numerical value of t. (Since here $\forall v(v \leq t \supset F(v))$ is a sentence, the term t must be a closed term.) The property of truth for Δ_0-sentences is primitive recursive; and for any Δ_0-sentence F, if F is true then $\vdash F$, and if F is not true, then $\vdash {\sim}F$ (see the Exercises). Thus PA is Δ_0-*complete.*

A Σ_1-formula is a formula $\exists v_1 \ldots \exists v_n F$, where F is Δ_0. That is, a Σ_1-formula may contain unbounded quantifiers, but they must all be existential quantifiers and must govern the rest of the formula. Symmetrically, a Π_1-formula is a formula $\forall v_1 \ldots \forall v_n F$, where F is Δ_0; here all the unbounded quantifiers are universal. (In these, we also let $n = 0$, so that every Δ_0-formula counts as both a Σ_1-formula and a Π_1-formula.)

Some logical equivalences should be remarked on at once. The negation of a Σ_1-formula is equivalent to a Π_1-formula, and vice-versa. The conjunction of two Σ_1-formulas is logically equivalent to a Σ_1-formula, as is the disjunction of two Σ_1-formulas. And the conjunction or disjunction of two Π_1-formulas is logically equivalent to a Π_1-formula. Hence we shall often speak somewhat loosely, for example, of a conjunction of Σ_1-formulas as though it were itself a Σ_1-formula, and a negation of a Σ_1-formula as thought it were itself a Π_1-formula.

In §4.3 we shall prove the

Strengthened Representability Theorem Every primitive recursive function is numeralwise representable in PA by a Σ_1-formula.

It follows from the Theorem that every primitive recursive relation is numeralwise representable in PA both by a Σ_1-formula and by a Π_1-formula. For suppose R is a primitive recursive relation, say, of two arguments. By definition, the characteristic function χ of R is primitive recursive, so there is a Σ_1-formula $F(x, y, z)$ that numeralwise represents χ. The formula $F(x, y, S0)$ is Σ_1 and the formula ${\sim}F(x, y, 0)$ is Π_1. Both formulas, we claim, numeralwise represent R. For if $R(k, n)$ then $\chi(k, n) = 1$, so that $\vdash F(\boldsymbol{k}, \boldsymbol{n}, z) \equiv z = S0$, and consequently $\vdash F(\boldsymbol{k}, \boldsymbol{n}, S0)$ and $\vdash {\sim}F(\boldsymbol{k}, \boldsymbol{n}, 0)$. And if $\overline{R}(k, n)$, then $\chi(k, n) = 0$ so that $\vdash F(\boldsymbol{k}, \boldsymbol{n}, z) \equiv z = 0$, and consequently $\vdash {\sim}F(\boldsymbol{k}, \boldsymbol{n}, S0)$ and $\vdash F(\boldsymbol{k}, \boldsymbol{n}, 0)$.

Let us now look at the formulas we considered in Chapters 2 and 3. Let us assume that we take those formulas to be of the least complexity possible.

The Gödel sentence is Π_1. As we originally defined it in §2.4 the Gödel sentence is obtained from a numeralwise representation of the primitive recursive relation $\overline{\mathrm{Der}}(k, \mathrm{diag}(n))$ by universally quantifying one free variable and replacing the other free variable with a numeral. Since the numeralwise repesentation may be taken to be Π_1, the Gödel sentence will also be Π_1.

The natural provability predicate $\text{Prov}(y)$ is Σ_1, since it is the existential quantification of a formula that numeralwise represents the primitive recursive relation $\text{Der}(k, n)$.

The proof of the Fixed Point Lemma given in §3.1 shows that there exists a fixed point of a formula $F(y)$ of the form $\forall z(\Delta(\boldsymbol{p}, z) \supset F(z))$ where $\Delta(y, z)$ is a representation of the diag function. If we take $\Delta(y, z)$ to be Σ_1 and if $F(y)$ is Π_1, then the fixed point will also be Π_1. Thus if we obtain the Gödel sentence in the manner of Chapter 3, by applying the Fixed Point Lemma to $\sim\text{Prov}(y)$, once again it will be Π_1. The Rosser sentence is also Π_1, since it is a fixed point of $\sim\text{Rov}(y)$, which is Π_1.

The formula (Con) is Π_1, since it is $\sim\text{Prov}(\ulcorner S0 = 0\urcorner)$ and $\text{Prov}(y)$ is Σ_1.

4.2 Σ_1-completeness and Σ_1-soundness

A Σ_1-sentence $\exists v_1 \ldots \exists v_m F(v_1, \ldots, v_m)$, where $F(v_1, \ldots, v_m)$ is Δ_0, is true iff there exist numbers $k_1, \ldots, k_m$ such that $F(\boldsymbol{k_1}, \ldots, \boldsymbol{k_m})$ is true. If this condition holds then $\vdash F(\boldsymbol{k_1}, \ldots, \boldsymbol{k_m})$ by Δ_0-completeness, so $\vdash \exists v_1 \ldots \exists v_m F(v_1, \ldots, v_m)$ by existential generalization. Thus PA is Σ_1-*complete*, in the sense that all true Σ_1 sentences are derivable.

(We do not have Σ_1-completeness is any stronger sense: a false Σ_1 sentence may fail to be refutable. Indeed, we have just seen that the Gödel sentence G is Π_1; hence $\sim G$ is a false Σ_1-sentence that is not refutable, assuming PA is consistent.)

The converse property to Σ_1-completeness is called Σ_1-*soundness*: every derivable Σ_1-sentence is true. Σ_1-soundness implies that if $\exists x F(x)$ is a derivable Σ_1-sentence then some numerical instance $F(\boldsymbol{k})$ is derivable. For if $\exists x F(x)$ is true, then $F(\boldsymbol{k})$ is true for some k, so that $\vdash F(\boldsymbol{k})$ by Σ_1-completeness. Note also that Σ_1-soundness implies consistency.

Since the Gödel sentence can be taken to be Π_1, Σ_1- soundness can be used instead of ω-consistency to show that it is not refutable. (Recall that consistency is sufficient to show that it is not derivable.) For if $\forall x A(x, \boldsymbol{p})$ is the Gödel sentence and is Π_1, then $\sim\forall x A(x, \boldsymbol{p})$ is equivalent to a Σ_1-sentence $\exists x {\sim} A(x, \boldsymbol{p})$. Were it derivable then by Σ_1-soundness there would be a k such that $\sim A(\boldsymbol{k}, \boldsymbol{p})$ is derivable, but as we showed $\vdash A(\boldsymbol{k}, \boldsymbol{p})$ for each k, so that we would have a violation of consistency.

Σ_1-soundness also implies the faithfulness of the natural provability predicate $\exists x D(x, y)$ where $D(x, y)$ numeralwise represents $\text{Der}(k, n)$ and is Σ_1. For if $\vdash \text{Prov}(\ulcorner F\urcorner)$ then by Σ_1-soundness there is a k such that $\vdash D(\boldsymbol{k}, \ulcorner F\urcorner)$; for this k we have $\text{Der}(k, \gamma(F))$, so that by mirroring $\vdash F$. Thus Σ_1-soundness may be used in the streamlined proof of Gödel's First Theorem given at the end of §3.1.

From Σ_1-soundness we may also infer that if we add an irrefutable Π_1-sentence as a new axiom, no Σ_1-sentences that were previously underivable become derivable, so that the extended system remains Σ_1-sound. For let J be the Π_1-sentence, and let K be a Σ_1-sentence derivable in the expanded system. By the Deduction Theorem (see Exercise 2.?), $J \supset K$ is derivable in PA. $J \supset K$ is equivalent to a Σ_1-sentence, so by the Σ_1-soundness of PA it is true. Since J is irrefutable, $\sim J$ is not true; hence K is true. Since PA is Σ_1-complete, $\vdash K$.

Finally, we note that Σ_1-soundness is equivalent to the restriction of ω-consistency to Σ_1-sentences: there is no Σ_1-sentence $\exists x F(x)$ such that $\vdash \exists x F(x)$ and also $\vdash \sim F(\boldsymbol{n})$ for every n. This property is called *1-consistency.* Suppose PA is Σ_1-sound and $\vdash \exists x F(x)$, where $\exists x F(x)$ is Σ_1. Then $\vdash F(\boldsymbol{k})$ for some k, as we noted above, and $\sim F(\boldsymbol{k})$ is not derivable, by consistency. (For the implication in the other direction, from 1-consistency to Σ_1-soundness, see the Exercises.)

If PA is Σ_1-sound, a Σ_1- sentence is derivable only if some numerical instance of it is derivable. This does not hold of more complex existential sentences, assuming PA is consistent. For example, let $F(x)$ be the formula $(x = 0 \,.\, R \vee x = S0 \,.\, \sim R)$, where R is the Rosser sentence. Since $\vdash R \vee \sim R$, we also have $\vdash \exists x F(x)$. But $F(\boldsymbol{k})$ is not derivable for any k. For if k is greater than 1 then $\vdash \sim F(\boldsymbol{k})$, while $\vdash F(0) \supset R$ and $\vdash F(S0) \supset \sim R$. Rosser's Theorem tells us that neither $\vdash R$ nor $\vdash \sim R$. Hence $F(\boldsymbol{k})$ is not derivable for any k.

As we've noted, Σ_1-soundness implies consistency. The converse is not true. In general there are systems that are consistent but not Σ_1-sound: for example, take the system obtained by adding the negation of the Gödel sentence as an additional axiom to PA. But even just restricted to PA there is no implication: it can be shown that if F formalizes the Σ_1-soundness of PA, then not $\vdash$ (Con) $\supset F$. (See the Exercises.)

Σ_1-completeness, in contrast, is an entirely elementary property. It follows from provabilities that can be established by elementary means in PA. This has the consequence that formalized Σ_1-completeness is derivable in PA: for any Σ_1-sentence F,

$$\vdash F \supset \mathrm{Prov}(\ulcorner F \urcorner)$$

Formal Adequacy is a particular case of formalized Σ_1- completeness, since $\mathrm{Prov}(\ulcorner F \urcorner)$ is a Σ_1-sentence. (More details on formalized Σ_1-completeness are contained in the Appendix, §6.)

From Δ_0- and Σ_1-completeness, we see that Gödel proved a best possible result. The simplest formula that could possibly be true but not provable in PA would be

Π_1; and that's exactly what the Gödel sentence is.

4.3 Proof of Representability

To show that every primitive recursive function is numeralwise representable we need to show three things: that the basic functions (the constant functions, the identity function, and the successor function) are numeralwise representable; that any function defined by composition from representable functions is itself representable; and that any function defined by recursion from representable functions is itself representable. As we shall see, the first two are easy, even when we add the requirement that the representations be Σ_1. The third, however, requires some work. To prove it, we must convert recursive definitions into explicit definitions that can be formalized in L_{PA}.

Consider a representative case: a 2-place function φ defined by

$$\varphi(n,0) = \theta(n)$$
$$\varphi(n,k+1) = \psi(n,k,\varphi(n,k))$$

where θ and ψ are functions known to be representable. The definition tells us how to compute $\varphi(n,k)$ for any n and k, in a stepwise fashion. First we obtain $\varphi(n,0)$by computing $\theta(n)$; call this number j_0. Next we obtain $\varphi(n,1)$ by computing $\psi(n,0,j_0)$; call the result j_1. To find $\varphi(n,2)$ we compute $\psi(n,1,j_1)$; call the result j_2. To find $\varphi(n,3)$ we compute $\psi(n,2,j_2)$. And so on. The integer j_k obtained by this process will be the value of $\varphi(n,k)$. Thus

> $\varphi(n,k) = p$ iff there exists a sequence $\langle j_0,\ldots,j_k\rangle$ of integers such that $j_0 = \theta(n)$, $j_{i+1} = \psi(n,i,j_i)$ for each $i < k$, and $j_k = p$.

This is an explicit definition: φ does not occur on the right side of the biconditional. To formalize it in L_{PA}, we must be able to deal formally with finite sequences. More precisely, we need to formalize the operation of extracting the members of a sequence from an integer that encodes the sequence. (In Chapter 2 we saw how to encode sequences using products of prime powers. That is of no help here, since we first need to deal with sequences in order to show that exponentiation is representable.) The following Lemma, sometimes called "Gödel's β-function Lemma", enables us to do this.

> **Sequence Encoding Lemma.** There exists a two-place primitive recursive function β such that

(a) for any k and any sequence $\langle j_0, \ldots, j_k \rangle$ of integers, there exists an integer s such that $\beta(s, i) = j_i$ for each $i, 0 \leq i \leq k$;

(b) β is numeralwise representable in PA by a Δ_0-formula $B(x, y, z)$ such that $\vdash B(x, y, z) \mathbin{\bullet} B(x, y, z') \supset z = z'$.

To prove the Lemma, we start by defining the *pairing function* π:

$$\pi(p, q) = \frac{1}{2} \cdot (p + q) \cdot (p + q + 1) + q$$

π maps pairs of integers one-one and onto the integers. To see this, given any r, let n be the largest number such that $\frac{1}{2} \cdot n \cdot (n + 1) \leq r$. Then $r - (\frac{1}{2}n \cdot (n + 1)) \leq n$, and $r = \pi(p, q)$ for $q = r - \frac{1}{2}(n \cdot (n + 1))$ and $p = n - q$, and for this pair p, q only.

We shall encode sequences by a pair of numbers p, q, and then take $\pi(p, q)$ as the number s in condition (i). Given a sequence $\langle j_0, \ldots, j_k \rangle$ let p be any prime that is both larger than k and larger than all the j_i, and let q be

$$0 + p \cdot j_0 + p^2 \cdot 1 + p^3 \cdot j_1 + p^4 \cdot 2 + p^5 \cdot j_2 + \ldots + p^{2k} \cdot k + p^{2k+1} \cdot j_k$$

.

(Thus, if q is written as a numeral in p-ary notation, it will have $2k + 2$ digits: counting from the right the odd places will be occupied by $0, 1, 2, ..., k$ and the even places by $j_0, j_1, j_2, \ldots, j_k$.) Let $\mathrm{Q}(p, m)$ be the relation "p is a prime and m is a power of p"; clearly $\mathrm{Q}(p, m)$ is numeralwise representable by a Δ_0 formula. Let $\mathrm{R}(s, i, j)$ be the relation

$(\exists m, n, p, q, r \leq s)(s = \pi(p, q) \;\&\; \mathrm{Q}(p, m) \;\&\; n < m^2 \;\&\; q = n + m^2 \cdot i + m^2 \cdot p \cdot j + m^2 \cdot p^2 \cdot r)$

We may then let $\beta(s, i) = \mu j \mathrm{R}(s, i, j)$. Condition ($a$) of the Lemma then follows, for $s = \pi(p, q)$, with p and q as above.

Now $\mathrm{R}(s, i, j)$ is also numeralwise representable by a Δ_0-formula; call it $F(x, y, z)$. Then let $B(x, y, z)$ be $F(x, y, z) \mathbin{\bullet} \forall x'(x' \leqslant z \mathbin{\bullet} F(x, y, x') \supset z = x')$. $B(x, y, z)$ is clearly Δ_0 and numeralwise represents $\beta(s, i)$.

For condition (b) of the Lemma, note that $B(x, y, z) \mathbin{\bullet} B(x, y, z')$ implies both $z' \leqslant z \supset z = z'$ and $z \leqslant z' \supset z' = z$. Since $\vdash z \leqslant z' \vee z' \leqslant z$, ($b$) follows.

The Sequence Encoding Lemma allows us to formalize the explicit definition of φ given above: by using numeralwise representations of θ and ψ as well as $B(x, y, z)$, we could easily write a formula that expresses the following: there exists an s such that $\beta(s,0) = \theta(n)$, $\beta(s,i+1) = \psi(n,i,\beta(s,i))$ for each $i < k$, and $\beta(s,k) = p$. We would thereby obtain a numeralwise representation of φ. However, since we want the representation to be Σ_1, a difficulty emerges: we may assume that the representations θ and ψ are Σ_1, but the formalization of the clause "$\beta(s,i+1) = \psi(n,i,\beta(s,i))$ for each $i < k$" will contain a bounded universal quantifier governing the Σ_1-formula representing ψ, and this would no longer be Σ_1.

To overcome this difficulty, we shall require a stronger notion of representation by a Σ_1-formula. (For readability, in what follows, we are going to use u, v, and w as if they were formal variables of L_{PA}, and we will use $v, w, z \leqslant u$ as shorthand for $v \leqslant u \mathbin{\centerdot} w \leqslant u \mathbin{\centerdot} z \leqslant u$.)

A formula $\exists u F(u,x,y,z)$ is an *excellent representation* of a 2-place function φ iff if $F(u,x,y,z)$ is Δ_0 and for all n, k and p such that $p = \varphi(n,k)$

(i) there exists an integer q such that $F(\boldsymbol{q},\boldsymbol{n},\boldsymbol{k},\boldsymbol{p})$;

(ii) $\vdash \exists u F(u,\boldsymbol{n},\boldsymbol{k},z) \supset z = \boldsymbol{p}$.

Since $\vdash \exists u F(x,\boldsymbol{n},\boldsymbol{k},\boldsymbol{p})$ follows from (i), an excellent representation $\exists u F(u,x,y)$ does indeed numeralwise represent φ. The additional requirements on excellent representations are that the formula contain only one unbounded existential quantifier, and that if $p = \varphi(n,k)$ not just $\exists u F(u,\boldsymbol{n},\boldsymbol{k},\boldsymbol{p})$ but also some numerical instance $F(\boldsymbol{q},\boldsymbol{n},\boldsymbol{k},\boldsymbol{p})$ is derivable. (As shown in the previous section, we could infer the derivability of a numerical instance by invoking Σ_1 soundness. But we do not want to make the Representability Theorem dependent on such a metamathematical supposition. Instead, we will build the derivability of a numerical instance into the construction of the representation.)

The definition of excellent representation for functions of one argument and of more than two arguments is similar. We also will call a Δ_0 formula that represents a primitive recursive function an excellent representation. (To conform literally to the definition above, we would need to add an existential quantifier binding a variable that does not actually appear in the formula.)

We now show that every primitive recursive function has an excellent representation. The basic functions are all representable by atomic formulas of L_{PA}, which are, of course, Δ_0 and so are excellent representations For composition, let us take a simple example, which is easily generalized. Suppose ζ is a 1-place function defined by $\zeta(k) = \nu(\xi(k))$, and suppose $\exists u N(u,x,y)$ and $\exists u X(u,x,y)$ are excellent representations of ν and ξ. Then let $Z(u,x,y)$ be

$$\exists v \exists w \exists z (v,w,z \leqslant u \mathbin{\centerdot} X(v,x,z) \mathbin{\centerdot} N(w,z,y))$$

We claim $\exists u Z(u,x,y)$ is an excellent representation of ζ. Clearly $Z(u,x,y)$ is Δ_0. Let any k be given and let $j = \xi(k)$, and $p = \nu(j)$, so that $p = \zeta(k)$. By supposition, there are numbers q and r such that $\vdash X(\boldsymbol{q},\boldsymbol{k},\boldsymbol{j}) \mathbin{.} N(\boldsymbol{r},\boldsymbol{j},\boldsymbol{p})$. Let s be the maximum of q, r, and j. Since $\vdash \boldsymbol{q},\boldsymbol{r},\boldsymbol{k} \leqslant \boldsymbol{s}$, we have

$$\vdash \exists v \exists w \exists z(v,w,z \leqslant \boldsymbol{s} \mathbin{.} X(v,\boldsymbol{k},z) \mathbin{.} N(w,z,\boldsymbol{p})$$

that is, $\vdash Z(\boldsymbol{s},\boldsymbol{k},\boldsymbol{p})$. And we also have

$$\vdash \exists u Z(u,\boldsymbol{k},\boldsymbol{n},y) \supset y = \boldsymbol{p}$$

because $\exists u Z(u,\boldsymbol{k},y)$ provably implies $\exists u \exists v \exists z(X(u,\boldsymbol{k},z) \mathbin{.} N(v,z,y))$, which by supposition provably implies $\exists v \exists z(z = \boldsymbol{j} \mathbin{.} N(v,z,y))$; by dint of the laws of identity this provably implies $\exists u N(u,\boldsymbol{j},y)$, which again by supposition provably implies $y = \boldsymbol{p}$.

For recursion, let us consider the example of the 2-place function function φ defined from θ and ψ as at the beginning of this section. Let $\exists u T(u,x,y)$ be an excellent representation of θ, and $\exists u R(u,v,x,y,z)$ be one of ψ. For readability, we introduce more shorthand: we exploit (b) of the Sequence Encoding Lemma and use $(v)_x$ as shorthand meant to express "the number z such that $B(v,x,z)$". That is, a formula $F((v)_x)$ is shorthand for $\exists z(z \leqslant v \mathbin{.} B(v,x,z) \mathbin{.} F(z))$. Condition (b) assures us that we can treat the shorthand exactly as if $(v)_x$ were a term of L_{PA}: that is, $\vdash F((v)_x) \mathbin{.} (v)_x = y \supset F(y)$. (See the Exercises.)

Let $P(u,x,y,z)$ be

$$\exists v(v \leqslant u \mathbin{.} \exists u'(u' \leqslant u \mathbin{.} T(u',x,(v)_0)) \mathbin{.} H(u,v,x,y) \mathbin{.} (v)_y = z)$$

where $H(u,v,x,y)$ is

$$\forall w(Sw \leqslant y \supset \exists u''(u'' \leqslant u \mathbin{.} R(u'',x,w,(v)_w,(v)_{Sw})))$$

We claim that $\exists u P(u,x,y,z)$ is an excellent representation of φ. Clearly $P(u,x,y,z)$ is Δ_0. Suppose any k is given, and let $p = \varphi(k)$. Let $j_0, \ldots j_k$ be defined as before, that is, $j_0 = \theta(n)$ and $j_{i+1} = \psi(n,i,j_i)$, and then let s be a number such that$\beta(s,i) = j_i$ for each $i \leq k$. Let q be such that $\vdash T(\boldsymbol{q},\boldsymbol{n},\boldsymbol{j_0})$, and for each $i < k$, let q_i be such that $\vdash R(\boldsymbol{q_i},\boldsymbol{n},\boldsymbol{i},\boldsymbol{j_i},\boldsymbol{j_{i+1}})$. Since $\exists u T(u,x,y)$ is an excellent representation of θ, and $\exists u R(u,v,x,y,z)$ is an excellent representation of ψ there will be such numbers q and $q_0,\ldots,q_{k-1}$. Finally, let r be the maximum of s, q, and the q_i.

We show first that $\vdash P(\boldsymbol{r},\boldsymbol{n},\boldsymbol{k},\boldsymbol{p})$. Now $\vdash T(\boldsymbol{q},\boldsymbol{n},\boldsymbol{j_0})$ and also $\vdash (\boldsymbol{s})_0 = \boldsymbol{j_0}$; hence $\vdash \exists u'(u' \leqslant \boldsymbol{r} \mathbin{.} T(u',\boldsymbol{n},(\boldsymbol{s})_0)$. For each $i < k$, we have $\vdash (\boldsymbol{s})_{\boldsymbol{i}} = \boldsymbol{j_i} \mathbin{.} ((\boldsymbol{s})_{\boldsymbol{i+1}} =$

$\boldsymbol{j_{i+1}}$, so that $\vdash \boldsymbol{q_i} \leqslant \boldsymbol{r} \,.\, R(\boldsymbol{q_i}, \boldsymbol{n}, \boldsymbol{i}, \boldsymbol{(s)_i}, (\boldsymbol{(s)_{i+1}}$. Consequently $\vdash \forall w(Sw \leqslant \boldsymbol{k} \supset \exists u''(u'' \leqslant \boldsymbol{r} \,.\, R(u'', \boldsymbol{n}, w, (\boldsymbol{s})_w, (\boldsymbol{s})_{Sw})))$, that is, $\vdash H(\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{n}, \boldsymbol{k})$. And finally we have $\vdash \boldsymbol{(s)_k)} = \boldsymbol{p}$.

To complete the proof, we need to show $\vdash \exists u P(u, \boldsymbol{n}, \boldsymbol{k}, z) \supset z = \boldsymbol{p}$. The following argument may easily be formalized in PA. Suppose $P(u, \boldsymbol{n}, \boldsymbol{k}, z)$. Let v be such that $\exists u' T(u', \boldsymbol{n}, (v)_0)) \,.\, H(u, v, \boldsymbol{n}, \boldsymbol{k}) \,.\, (v)_k = z)$. Since $\exists u T(u, x, y)$ is an excellent representation of θ and $j_0 = \theta(n)$, we have $(v)_0 = \boldsymbol{j_0}$. Now, letting the variable w in $H(u, v, \boldsymbol{n}, \boldsymbol{k})$ be 0, we have $\exists u'' R(u'', \boldsymbol{n}, 0, (v)_0, (v)_{S0})$. Since $\exists u R(u, v, x, y, z)$ is an excellent representation of ψ and $j_1 = \psi(n, 0, j_0)$, we have $(v)_{S0} = j_1$. Continuing in this way, taking the variable w in $H(u, v, \boldsymbol{n}, \boldsymbol{k})$ to take successively the values $1, 2, \ldots, k-1$, we obtain $(v)_{\boldsymbol{i}} = \boldsymbol{j_i}$ for each $i \leq k$. Since $(v)_{\boldsymbol{k}} = z$, and $j_k = p$, we obtain $z = \boldsymbol{p}$.

Chapter 5

Formalized Semantics

5.1 Tarski's Theorem

Tarski's Theorem says that no sufficiently rich formal language contains its own truth predicate. To see what this means we must first define the notion of truth predicate. Let L be a formal language, and suppose we have fixed an intended interpretation of L. This interpretation provides a notion of the truth of a sentence of L; that is, we use "true" to mean true under this intended interpretation. To give formal representation to this notion of truth, we suppose that L is gödelized, and that L contains numerals, i.e., syntactic objects which under interpretation denote the integers. A truth predicate for L is a formula $Tr(y)$ such that, for each sentence F of L, the biconditional

$$\mathrm{Tr}(\ulcorner F \urcorner) \equiv F$$

is true. (In this definition, $\mathrm{Tr}(y)$ may be a formula in L or a formula in some language that extends L. In the latter case, by the truth of the biconditional we mean its truth under the intended interpretation of this more inclusive language.)

This definition is motivated by the following philosophical observation of Tarski's. Suppose we wish to give an account of the notion is true, as applied to the sentences of a language like English. As a minimal condition on any such account, we should be able to derive from it all sentences

"Snow is white" is true iff snow is white

"Tucson is in Arizona" is true iff Tucson is in Arizona

"$2+2=5$" is true iff $2+2=5$

and so on. After all, one might argue, it is biconditionals like this which give us the sense of the notion of truth. So whatever else our account tells us about truth, it had better yield those biconditionals. The biconditionals are sometimes called *Tarski paradigms*, *Tarski biconditionals*, or *T-sentences.* We may phrase the condition generally: the account must yield everything of the form

$$S \text{ is true iff } p$$

where for p we substitute a sentence and for S a name of that sentence. So let us consider the language L_{PA}, with its intended interpretation. Tarski's Theorem can be obtained immediately from the Fixed Point Theorem. For suppose $T(y)$ is any formula of L_{PA} with one free variable. By the Fixed Point Theorem applied to the formula $\sim T(y)$, there is a sentence H of L_{PA} such that $\vdash \sim T(\ulcorner H \urcorner) \equiv H$. Hence by the soundness of PA, $\sim T(\ulcorner H \urcorner) \equiv H$ is true in the intended interpretation of PA. But then $T(y)$ cannot be a truth predicate for L_{PA}; it fails to act like a truth predicate on the formula H.

Clearly the same argument can work more generally. Given a language L, suppose Σ is a formal system that is sound for the intended interpretation of L and in which the primitive recursive function diag can be numeralwise represented. Then the Fixed Point Theorem may be proved, and as above we obtain, given any purported truth predicate $T(y)$, a sentence H on which $T(y)$ fails to act like a truth predicate. Note that, first, the sentence H is obtained constructively. That is, the proof of the Fixed Point Theorem provides a method for constructing H, given $T(y)$. Note second that the failure of $T(y)$ to act like a truth predicate on H is exhibited proof-theoretically, by the derivability of a biconditional that *contradicts* Tarski's paradigm $T(\ulcorner H \urcorner) \equiv H$. Of course, to obtain such a proof-theoretic counterexample, we have to bring in a formal system Σ. To obtain Tarski's Theorem directly, that is, without any use of formal systems, we would have to proceed entirely semantically. We carry this out immediately below.

The proof of Tarski's Theorem is just a formalization of the Liar Paradox, whose starkest form is this: let (*) be the sentence "Sentence (*) is not true". If (*) is true then it is not true, and if it is not true then it is true; contradiction. The paradox arises, so it seems, from the assumption that the words "is true" function as a truth predicate for all English sentences, including those containing the words "is true". The only thing needed to carry the argument out in a formal setting is a way of getting the effect of the self-reference, of having (*) be a sentence that talks about itself. That is the role of Gödel's function diag.

In the proof given above, the formal system Σ entered only through the use of the Fixed Point Theorem. A proof of Tarski's Theorem that does not mention formal systems at all can be obtained if we replace this step by a use of a semantic form of the Fixed Point Theorem, to wit: for every formula $F(y)$ of L there is a formula H such that $F(\ulcorner H \urcorner) \equiv H$ is true. Given this, for every formula $T(y)$ of L there will be a sentence H such that $\sim T(\ulcorner H \urcorner) \equiv H$ is true, and so $T(y)$ fails to be a truth predicate. To prove the semantic Fixed Point Theorem, we introduce the notion of definability in L.

> A formula $F(v_1, \ldots, v_m)$ of L *defines* an m-place relation R just in case, for all $n_1, \ldots, n_m$, $F(\boldsymbol{n_1}, \ldots, \boldsymbol{n_m})$ is true iff $R(n_1, \ldots, n_m)$. A relation is *definable* in L iff there exists a formula of L that defines it.

Definability is a semantic correlate of numeralwise representability. Indeed, if $F(v_1, \ldots, v_m)$ numeralwise represents R in a formal system that is sound for the intended interpretation of L, then $F(v_1, \ldots, v_m)$ defines R in L. The converse does not hold. In general, there are many relations that are definable but not numeralwise representable. For example, the 1-place relation Bew is, as we have seen, not numeralwise representable in PA (assuming PA consistent). Yet it is definable in L_{PA}, since the formula $\exists x D(x, y)$ defines, it, where $D(x, y)$ is any formula that numeralwise represents the relation $\mathrm{Der}(m, n)$.

Note that Tarski's Theorem may be stated thus: the class of gödel numbers of true sentences of L is not definable in L.

A function is said to be definable in L iff its graph is definable in L. Now suppose that the function diag is definable in L. Thus there exists a formula $\Delta(x, y)$ of L such that $\Delta(\boldsymbol{k}, \boldsymbol{n})$ is true iff $n = \mathrm{diag}(k)$. Given a formula $F(y)$ of L, let p be the gödel number of $\exists z(\Delta(y, z) \mathbin{.} F(z))$ and let H be $\exists z(\Delta(\boldsymbol{p}, z) \mathbin{.} F(z))$. Note that the gödel number of H is $\mathrm{diag}(p)$. Since $\mathrm{diag}(p)$ is the only value of z for which $\Delta(p, z)$ is true, H will be true iff $F(\boldsymbol{q})$ is true, where $q = \mathrm{diag}(p)$. That is, $H \equiv F(\boldsymbol{q})$ is true. But $F(\boldsymbol{q})$ is just $F(\ulcorner H \urcorner)$; thus the semantic Fixed Point Theorem is proved.

This proof differs little from the proof of the syntactic Fixed Point Theorem of §**??**. The only difference is this: we take $\Delta(x, y)$ to define diag, rather than to numeralwise represent it in some formal system. Therefore, instead of the derivability of $H \equiv F(\ulcorner H \urcorner)$ in the formal system, our conclusion is the truth of this biconditional. Note that the only properties of L needed are these: the function diag is definable in L; and L contains the usual logical signs (quantifiers and truth-functions). This, in the end, is the cash value of the condition "sufficiently rich" that we used in our original statement of Tarski's Theorem.

Now Tarski's Theorem says there is no truth predicate for L in L. It does not preclude the existence of a truth predicate for L in some language extending L. Tarski went on to show how truth predicates for various formal languages can indeed be constructed in more extensive formal languages. We now turn to an examination of his definition of truth, taking L_{PA} as the formal language to be treated.

5.2 Defining truth for $L_{\mathbf{PA}}$

We wish to define truth in the intended interpretation of L_{PA}. Truth is a property of sentences, that is, formulas without free variables; and the truth of a complex sentence depends in a straightforward way on the truth of sentences from which it is constructed.

> A sentence $\sim F$ is true iff F is not true; a sentence $F \vee G$ is true iff F is true or G is true; and a sentence $\forall v F(v)$ is true iff $F(\boldsymbol{n})$ is true for every n.

Note that this last clause is correct because every member of the universe of discourse is, in the intended interpretation, denoted by some formal numeral $\boldsymbol{n}$. Consequently, we can easily formulate an inductive definition of truth, that defines the truth of a sentence in terms of the truth of sentences of lesser logical complexity.

As a base clause for this inductive definition, we need a definition of truth for atomic sentences, that is, for atomic formulas that contain no variables. Atomic sentences have the form $s = t$, where s and t are terms of L_{PA} without variables. A term without variables has a fixed numerical value in the intended interpretation of L_{PA}. For example, the term $((SS0 \times SSS0) + S0)$ has the value 7. It is not hard to see that there is a primitive recursive function nv that takes each such term to its numerical value. (See the Exercises.) We use it to define truth for atomic sentences:

$$\mathrm{Tr}_{\mathrm{At}}(n) \equiv \mathrm{Sent}(n) \ \&\ \mathrm{Atform}(n) \ \&\ (\exists j)(\exists k)(j, k \le n \ \&\ n = j{*}2^5{*}k \ \&\ \mathrm{nv}(j) = \mathrm{nv}(k)).$$

(Here $\mathrm{Sent}(n)$ is the p.r. relation true just of the gödel numbers of sentences, i.e., $\mathrm{Sent}(n)$ iff $\mathrm{Form}(n)$ & $(\forall i, j)(i, j \le n \to \overline{\mathrm{Free}}(i, j, n))$.) Clearly $\mathrm{Tr}_{\mathrm{At}}(n)$ holds iff n is the gödel number of a true atomic sentence. Note, by the way, that $\mathrm{Tr}_{\mathrm{At}}$ is primitive recursive.

The inductive definition of truth is obtained by mirroring the inductive characterization displayed above:

$$(*) \qquad \begin{aligned} \mathrm{Tr}(n) \ \equiv\ & \mathrm{Sent}(n) \ \&\ [(\mathrm{Atform}(n) \ \&\ \mathrm{T}_{\mathrm{At}}(n)) \\ & \vee (\exists i < n)(n = \mathrm{neg}(i) \ \&\ \mathrm{Sent}(i) \ \&\ \overline{\mathrm{Tr}}(i)) \end{aligned}$$

$$\begin{aligned}&\vee (\exists i, j < n)(n = \mathrm{dis}(i,j) \mathbin{\&} (\mathrm{Tr}(i) \vee \mathrm{Tr}(j)))\\ &\vee (\exists i, k < n)(n = \mathrm{gen}(k,i) \mathbin{\&} (\forall p)(\mathrm{Tr}(\mathrm{sub}(i,k,\mathrm{nmrl}(p)))))].\end{aligned}$$

Definition $(*)$ uniquely determines the numbers of which Tr holds. This may be shown by an induction on logical complexity. Let $\mathrm{lc}(n) = k$ if n is the gödel number of a sentence of L_{PA} containing k occurrences of the logical signs "$\sim$", "$\vee$", "$\forall$"; also let $\mathrm{lc}(n) = 0$ if n is not the gödel number of a sentence. Clearly, $\mathrm{Tr}(n)$ is uniquely determined for all n with $\mathrm{lc}(n) = 0$, since for such n we have either $\overline{\mathrm{Sent}}(n)$ or $\mathrm{Atform}(n)$; and if $\mathrm{Tr}(k)$ is determined for all k with $\mathrm{lc}(k) < \mathrm{lc}(n)$, then the clauses of $(*)$ suffice to fix whether or not $\mathrm{Tr}(n)$. It should also be clear that the property Tr so determined is the one we want: $\mathrm{Tr}(n)$ holds iff n is the gödel number of a true sentence of L_{PA}.

Although definition $(*)$ is inductive, it is not a primitive recursive definition, because there is an unbounded quantifier on the right-hand side. Whether or not $\mathrm{Tr}(n)$ holds when $n = \mathrm{gen}(k,i)$ depends upon whether Tr holds of an infinite number of other numbers. This makes it impossible to obtain an explicit definition by the method of Chapter 4. Indeed, Tarski's Theorem tells us that an explicit definition—that is, a biconditional of the form $\mathrm{Tr}(n) \equiv X$ where X does not contain Tr—cannot be obtained if X is limited to number-theoretic notions, that is, to primitive recursive functions and relations, truth-functions, and quantifiers that range over the integers.

To obtain an explicit definition of truth for L_{PA} a more powerful metalanguage must be used. In fact, we can obtain such an explicit definition by allowing the metalanguage to include variables and quantifiers ranging over sets of integers. Using "X" as such a variable, let $(*_X)$ be obtained from (*) by replacing each occurrence of "Tr(___)" with an occurrence of "___ $\in X$" ("$n \in X$" means "n belongs to the set X"). As we noted above, there is one and only one set X such that $(*_X)$ holds for all n. Hence we may define

$$(**) \qquad \mathrm{Tr}(m) \equiv (\exists X)[(\forall n)(*_X) \mathbin{\&} m \in X].$$

And since $\forall n(*_X)$ holds for one and only one X, we could equally well define

$$(***) \qquad \mathrm{Tr}(m) \equiv (\forall X)[(\forall n)(*_X) \to m \in X].$$

Note 5.1: To show that the predicate Tr as defined by $(**)$ or by $(***)$ itself obeys $(*)$ for all n, one would have to show that $(\forall n)(*_X)$ holds for one and only one set X. As indicated above, uniqueness can be shown by induction on $\mathrm{lc}(n)$. The existence of such an X also can be shown by induction on $\mathrm{lc}(n)$. Call a set X *p-good* iff $(*_X)$ holds for all n such that $\mathrm{lc}(n) \le p$. It is fairly simple to show, by induction on p, that for each p there exists a p-good set, and that all p-good sets agree on those n such that $\mathrm{lc}(n) \leq p$. A set X such that $(*_X)$ holds for all n can then be obtained by stitching together p-good sets for each p. Indeed, one can even avoid the necessity of showing the existence of a set X such that $(\forall n)(*_X)$ by framing a definition of Tr directly in terms of p-good sets. We can define

$$\mathrm{Tr}(m) \equiv (\exists X)[(\forall n)(\mathrm{lc}(n) \leq \mathrm{lc}(m) \to (*_X)) \ \&\ m \in X]$$

,

that is, $\mathrm{Tr}(m)$ holds iff m belongs to some $\mathrm{lc}(m)$-good set. Equivalently, we could define

$$\mathrm{Tr}(m) \equiv (\forall S)[(\forall n)(\mathrm{lc}(n) \leq \mathrm{lc}(m) \to (*_X)) \to m \in X]$$

,

that is, $\mathrm{Tr}(m)$ holds iff m belongs to every $\mathrm{lc}(m)$-good set. It will still follow that Tr itself obeys $(*)$ for all n. **End of Note.**

5.3 Uses of the truth-definition

In this section we sketch how the definition of truth, formalized in a formal language that extends L_{PA}, can be used to show the soundness of formal system PA. We consider a formal language that extends L_{PA} by the inclusion of variables ranging over sets of numbers. Let $\mathrm{Tr}(x)$ be the formalization in such a language of the definition of $\mathrm{Tr}(m)$, using any of the alternatives from the end of the previous section. We assume a formal system has been specified which extends PA and which allows us to derive the formalized version of $(\forall n)(*)$. In the remainder of this section we use "$\vdash^*$" to mean derivability in such a system. (Details about a suitable formal language and formal system are in §**??** below). From the derivability of a formalized version of $(\forall n)(*)$ follows the derivability of the Tarski paradigms: that is, for each sentence F of L_{PA},

$$\vdash^* \mathrm{Tr}(\ulcorner F \urcorner) \equiv F.$$

(The proof of this is not entirely straightforward, but basically it proceeds by induction on the logical complexity of F.)

To say that PA is sound is to say that all derivable formulas are true. Here, however, we are using "true" to apply not just to sentences but to formulas with free variables as well. A formula with free variables is said to be true iff it is true for all values of its variables. This holds iff the universal closure of the formula is true, and also iff all numerical instances of the formula are true, where a numerical instance of a formula is the result of replacing all free variables with formal numerals.. Let $\mathrm{NI}(m, n)$ be the p.r. relation that holds iff m is the gödel number of a formula and n is the gödel number of a numerical instance of that formula. Let $\mathrm{Prov}(y)$ be a standard formalization of derivability in PA. The following Claim is, then, that the soundness of PA is derivable in the formal system.

Claim. $\vdash^* \forall y(\mathrm{Prov}(y) \supset \forall z(\mathrm{NI}(y, z) \supset \mathrm{Tr}(z)))$

Sketch of Proof. One shows the derivability of the formalizations of "Every axiom of PA is true" and of "The rules of inference preserve truth". Indeed, the derivability of the latter, and of the former for the logical axioms of PA, follows in a straightforward way from the derivability of $(*)$. For example, consider an axiom of the form $F \supset (F \vee G)$. All numerical instances of this axiom have the form $F' \supset (F' \vee G')$, where F' and G' are sentences. From $(\forall n)(*)$ we can infer (gödelizations) of the following claims: if F' and G' are sentences, then $F' \supset (F' \vee G')$ is true iff either F' is not true or $(F' \vee G')$ is true, and $(F' \vee G')$ is true iff either F' is true or G' is true. Hence $F' \supset (F' \vee G')$ is true whenever F' and G' are sentences; and hence the numerical instances of any formula $F \supset (F \vee G)$ is true.

The same sort of argument works for the other logical axioms and for the rules of inference. These arguments can easily be formalized to yield derivations in the formal system. The individual nonlogical axioms of PA can most easily be derived true by using the Tarski paradigms applied to their universal closures, since they are also axioms of the extended system. That is, for example, $\vdash^* \mathrm{Tr}(\ulcorner\forall x(\sim Sx = 0)\urcorner)$ because $\vdash^* \mathrm{Tr}(\ulcorner\forall x(\sim Sx = 0)\urcorner) \equiv \forall x(\sim Sx = 0)$ and $\vdash^* \forall x(\sim Sx = 0)$. It remains to show that the truth of all induction axioms $F(0) \mathbin{.} \forall x(F(x) \supset F(Sx)) \supset \forall x F(x)$, where $F(x)$ is any formula of L_{PA}, can be derived in the formal system. We must show that, for any numerical instance $F'(x)$ of $F(x)$, if $F'(0)$ and $\forall x(F'(x) \supset F'(Sx))$ are true then so is $\forall x F'(x)$. It should not be surprising that to derive this we must use induction. Consider the property that holds of a number n iff $F'(n)$ is true. If $F'(0)$ and $\forall x(F'(x) \supset F'(Sx))$ are true then 0 has the property and the successor of any number with the property also has the property. Hence by induction every number has the property; hence $\forall x F'(x)$ is true. Note that, because we are showing this for any numerical instance $F'(x)$ of any formula $F(x)$, the property must invoke the notion of truth; hence the formalization of this argument will use an induction axiom that contains the formula Tr(x). □

Now let Con(PA) be the formalization of the consistency of PA, i.e., the formula $\sim$Prov$(\ulcorner S0 = 0\urcorner)$.

Corollary $\vdash^*$ Con(PA)

Proof. Since $(*)$ is derivable, so are Tarski paradigms; hence $\vdash^*$ Tr$(\ulcorner S0 = 0\urcorner) \equiv S0 = 0$. Since the formal system extends PA, $\vdash^* \sim S0 = 0$. Hence $\vdash^* \sim$Tr$(\ulcorner S0 = 0\urcorner)$. By the Claim, $\vdash^* \sim$Prov$(\ulcorner S0 = 0\urcorner)$. □

Note that a similar argument shows that in the formal system we can derive the formal statement of the ω-consistency of PA. The Corollary shows that our envisaged extension of PA can derive formulas of L_{PA}, that are not derivable in PA. Thus, not only is it expressively richer—it can formalize notions that PA cannot, like truth for L_{PA}—but also it is deductively stronger with respect to that part of the language that is common to it and L_{PA}.

5.4 Second-order Arithmetic

In this section we specify some extensions of L_{PA} and of PA that could be used to obtain the result of §**??**. We introduce L_{SA}, *the language of second-order arithmetic.* To obtain this language, we add to the alphabet of L_{PA} variables X, Y, Z, X', Y', $Z', \ldots$. These are called *set variables*; the are intended to range over sets of integers. We now refer to variables $x, y, z, x', \ldots$ as *numerical variables.* To the formation rules we add the following clauses:

> If t is a term and V is a set variable, then $V(t)$ is a formula (an atomic formula); if F is a formula and V is a set variable, then $\forall V(F)$ is a formula.

That completes the specification of the language. This is a *two-sorted language*, since there are two sorts of variables, which play different syntactic roles. A variable of one sort may not be substituted for another, that is, such a substitution will lead from a formula to a non-formula. The intended interpretation of an atomic formula $V(t)$ is that the number denoted by t belongs to the set V.

We now consider axioms for a formal system in this language. The schemata for logical axioms remain as before, although of course restrictions on sort of variable have to be observed. That is, there are separate universal instantiation axiom schemata for each sort: $\forall vF \supset F(v/t)$ for v a numerical variable and t a term, and $\forall VF \supset F(V/U)$, for V and U set variables. The formal system SA, or full second-order arithmetic, has the following non-logical axioms: the number-theoretic axioms (N1)–(N6) of PA, together with the induction axiom

$$X(0) \mathbin{.} (\forall x)(X(x) \supset X(Sx)) \supset \forall xX(x)$$

in which the set variable X is a free variable; and the *comprehension axioms*

$$\exists X\forall x(X(x) \equiv F(x))$$

whenever $F(x)$ is a formula of L_{SA} in which X does not occur free. As a result of including the comprehension axioms, mathematical induction can be framed as a single axiom rather than as a schema. That is, if $F(x)$ is any formula, $F(0) \mathbin{.} \forall x(F(x) \supset F(Sx)) \supset \forall xF(x)$ can be derived from the single axiom of mathematical induction and the comprehension axiom $\exists X\forall x(X(x) \equiv F(x))$.

SA is a very powerful formal system. It goes far beyond number theory: since real numbers can be encoded as sets of integers, theorems of the theory of real numbers and of the calculus can be derived in it. For that reason, SA is sometimes called "classical analysis". (Of course, despite its strength, SA is still incomplete, assuming it is consistent. That is, since clearly SA can be gödelized, all p.r. functions can be represented, and a standard provability predicate can be specified, all the results of Chapter 3 apply to SA.)

Now the language L_{SA} can be used to formalize the definition of truth for L_{PA}. And SA is certainly powerful enough to derive the formalization of $(\forall n)(*)$. SA also contains the mathematical induction axioms needed to carry through the proof of the soundness of PA. Hence we see that there are sentences of L_{PA} that are derivable in SA but not in PA. (Assuming PA consistent.) Mathematicians used to ask whether any theorem about the numbers proved using "analytic methods" (methods that invoked the real numbers and their laws) could in principle be proved using only "elementary methods", that is, based on the usual first-order properties of the integers. What we have seen shows the answer to this question is negative.

Now SA is far more powerful a system than is needed to show the soundness of PA. The system ACA (arithmetical comprehension axioms) suffices. This theory has, instead of all comprehension axioms, just those obtained from the comprehension schema by replacing $F(x)$ with a formula containing no bound set variables. Thus the only formulas that determine sets of integers are those restricted to quantifying over integers. However, because of this restriction on comprehension axioms, we must add a schema of mathematical induction, that is,

$$F(0) \mathbin{.} \forall x(F(x) \supset F(Sx)) \supset \forall x F(x)$$

for every formula $F(x)$ of L_{SA}.

That ACA suffices to derive the formalized version of $(\forall n)(*)$ relies on the fact that, for any p, a p-good set can be defined by a formula with no bound set variables. Then one of the definitions of $\mathrm{Tr}(m)$ suggested at the end of §**??** can be used. The schema of induction is needed, rather than the formulation using a free set variable, because the proof of soundness of PA needs induction for formulas involving the formalization $\mathrm{Tr}(x)$ of the definition of $\mathrm{Tr}(m)$, and that formalization contains bound set variables.

The formal system with language L_{SA} whose comprehension axioms are those of ACA and whose induction axiom is the single axiom using a free set variable, a formal system called ACA_0, is too weak to prove the soundness of PA. In fact, any formula of L_{PA} derivable in ACA_0 is derivable in PA. (In logicians' jargon, ACA_0 is a *conservative extension* of PA.) Thus, ACA_0 can express things that cannot be expressed in L_{PA}, and derive interesting things about them—it can formulate a truth-definition and prove the Tarski paradigms— but cannot derive any purely number-theoretic facts beyond what is derivable in PA.

In SA or ACA the consistency of PA is derivable; hence in those systems, the Gödel sentence for PA is also derivable. Now we know that, if PA is consistent, then in PA one cannot derive any formula that asserts an underivability in PA. What we have just seen is that in SA or ACA one can derive such statements about underivability in PA.

Nonetheless, even in SA one cannot derive every correct statement about underivability in PA (assuming SA consistent). For let $\mathrm{Prov}_{\mathrm{SA}}(y)$ be a provability predicate for SA formulated in language L_{PA}. We claim first that, for any formula F of L_{SA},

$$\vdash \mathrm{Prov}_{\mathrm{SA}}(\ulcorner F \urcorner) \supset \mathrm{Prov}(\ulcorner \mathrm{Prov}_{\mathrm{SA}}(\ulcorner F \urcorner)\urcorner),$$

that is, in PA we can derive the statement "If a formula is derivable in SA, then in PA it can be derived that the formula is derivable in SA". The following reasoning, once formalized, establishes this: if F is derivable in SA, then for some m $\mathrm{Der}_{\mathrm{SA}}(m, \gamma(F))$, where $\mathrm{Der}_{\mathrm{SA}}$ mirrors the notion of "derivation in SA". Since $\mathrm{Der}_{\mathrm{SA}}$ can be numeralwise represented in PA, $\vdash_{\mathrm{PA}} \mathrm{Der}_{\mathrm{SA}}(m, \ulcorner F \urcorner)$, so that $\vdash_{\mathrm{PA}} \exists x \mathrm{Der}_{\mathrm{SA}}(x, \ulcorner F \urcorner))$, that is $\vdash_{\mathrm{PA}} \mathrm{Prov}_{SA}(\ulcorner F \urcorner)$.

Now let F be "$S0 = 0$", and take the contrapositive:

$$\sim\!\mathrm{Prov}(\ulcorner \mathrm{Prov}_{SA}(\ulcorner S0 = 0 \urcorner)\urcorner) \supset \sim\!\mathrm{Prov}_{SA}(\ulcorner S0 = 0 \urcorner).$$

Since this conditional is derivable in PA, it is derivable in SA. The consequent of this conditional is just Con(SA), and so is not derivable in SA (assuming SA consistent). Hence the antecedent is not derivable in SA, that is, in SA we cannot derive the statement that the inconsistency of SA is not derivable in PA.

5.5 Partial truth predicates

Although there is no truth predicate for L_{PA} in L_{PA}, there are truth predicates in L_{PA} for fragments of L_{PA}. In §4.1, we defined the Δ_0, Σ_1, and Π_1 formulas. Also we saw that truth for Δ_0 sentences is primitive recursive (see the Exercises).

A Σ_1 sentence has the form $\exists v_1 \ldots \exists v_m H(v_1, \ldots, v_m)$, where $H(v_1, \ldots, v_m)$ is a Δ_0 formula having only the free variables indicated. Such a sentence is true iff there exists a sequence $\langle p_1, \ldots, p_m\rangle$ of numbers such that $H(\boldsymbol{p_1}, \ldots, \boldsymbol{p_m})$ is true. Now the gödel number of $H(\boldsymbol{p_1}, \ldots, \boldsymbol{p_m})$ is a primitive recursive function of the gödel number of $\exists v_1 \ldots \exists v_m H(v_1, \ldots, v_m)$ and the (encoding of) the sequence $\langle p_1, \ldots, p_m\rangle$, while the truth of $H(\boldsymbol{p_1}, \ldots, \boldsymbol{p_m})$ is a primitive recursive matter, as $H(\boldsymbol{p_1}, \ldots, \boldsymbol{p_m})$ is a Δ_0 sentence. Hence the truth of $H(\boldsymbol{p_1}, \ldots, \boldsymbol{p_m})$ is a primitive recursive relation of the gödel number of $\exists v_1 \ldots \exists v_m H(v_1, \ldots, v_m)$ and the (encoding of) the sequence $\langle p_1, \ldots, p_m\rangle$. This relation can therefore be formalized with a Σ_1 formula. The statement "there exists a sequence $\langle p_1, \ldots, p_m\rangle$ of numbers such that $H(\boldsymbol{p_1}, \ldots, \boldsymbol{p_m})$ is true" can then be formalized by adding more existential quantifiers at the front. It follows that truth for Σ_1 sentences can be formalized by a Σ_1 formula. And then we also obtain that truth for Π_1 sentences can be formalized by a Π_1 formula, since the truth of a Π_1 sentence F is just the falsity of the Σ_1 sentence $\sim F$.

Now let us define, inductively: a Σ_{n+1}-formula is a formula $\exists v_1 \ldots \exists v_m H$, where H is a Π_n-formula, and a Π_{n+1}-formula is a formula $\forall v_1 \ldots \forall v_m H$, where H is a Σ_n formula. (As before, we allow m to be 0. This has the effect of making this a cumulative hierarchy of formulas: if F is Σ_n or Π_n, it is both Σ_p and Π_p for all $p > n$.)

We claim that, for each n, truth for Σ_n sentences can be expressed by a Σ_n-formula, and (consequently) truth for Π_n sentences can be expressed by a Π_n-formula. The argument goes by induction on n, and essentially repeats what we just noticed for Σ_1 sentences. Assume that truth for Π_n sentences can be expressed by a Π_n-formula. Let q be the gödel number of a Σ_{n+1} sentence $F = \exists v_1 \ldots \exists v_m H(v_1, \ldots, v_m)$, where $H(v_1, \ldots, v_m)$ is Π_n. F is true iff there exists a sequence $\langle p_1, \ldots, p_m\rangle$ such that $H(\boldsymbol{p_1}, \ldots, \boldsymbol{p_m})$ is true. The gödel number of the latter formula is a primitive recursive function of the (encoding of) the sequence

$\langle p_1, \ldots, p_m \rangle$ and q. Lets say this function is $\varphi(\langle p_1, \ldots, p_m \rangle, q)$. We have: F is true iff there exists a sequence $\langle p_1, \ldots, p_m \rangle$ such that $\varphi(\langle p_1, \ldots, p_m \rangle, q)$ is the gödel number of a true Π_n sentence. That is, F is true iff there exists a sequence $\langle p_1, \ldots, p_m \rangle$ and there exists a r such that

1. $r = \varphi(\langle p_1, \ldots, p_m \rangle, q)$;
2. r is the gödel number of a true Π_n sentence.

Clause (2) can be expressed by a Π_n formula, by assumption; clause (1) can be expressed by either a Π_1 or a Σ_1 formula. It follows that this entire condition can be expressed by a Σ_{n+1} formula.

Call a formula of L_{PA} *essentially* Σ_n or Π_n if it can be transformed into a Σ_n or a Π_n formula, respectively, by the usual prenexing rules. Thus every formula of L_{PA} is essentially Σ_n or essentially Π_n for some n, and it is straightforward to extend the formalizations of Σ_n truth and Π_n truth to essentially Σ_n and essentially Π_n sentences (see the Exercises).

The existence of these partial truth predicates in L_{PA} has an interesting consequence. Pick any n. Consider the system obtained from PA by restricting all axioms and all formulas in derivations to essentially Σ_n formulas. Call this system PA$_n$. Let $\text{Prov}^*(w, y)$ be a formalization of the notion of derivability in PA$_w$, and let $\text{T}_n(y)$ be the Σ_n formalization of truth for Σ_n sentences. It shouldnt be surprising that PA can prove the soundness of PA$_n$, i.e., for any formula F derivable in PA$_n$, F is true (or, more properly speaking, every numerical instance of F is true). That is, we have

$$\vdash \forall y[\text{Prov}^*(\boldsymbol{n}, y) \supset \forall z(\text{NI}(y, z) \supset \text{T}_n(z))].$$

But since $\vdash \sim\text{T}_n(\ulcorner S0 = 0 \urcorner)$, it follows that

$$\vdash \sim\text{Prov}^*(\boldsymbol{n}, \ulcorner S0 = 0 \urcorner).$$

That is, for each n, PA proves "PA$_n$ is consistent".

But, assuming PA is consistent, PA cannot prove "for each n, PA$_n$ is consistent", that is, $\nvdash \forall x \sim \text{Prov}^*(x, \ulcorner S0 = 0 \urcorner)$. For, it can be shown that $\vdash \forall y[\text{Prov}(y) \supset \exists x(\text{Prov}^*(x, y))]$, that is, we can derive in PA the formalization of the obvious fact that if a formula is derivable in PA it is derivable in some PA$_n$. But then $\vdash \forall x \sim \text{Prov}^*(x, \ulcorner S0 = 0 \urcorner) \supset \sim\text{Prov}(\ulcorner S0 = 0 \urcorner)$, so if PA could derive "for each n, PA$_n$ is consistent" then it could derive "PA is consistent", violating Gödel's Second Theorem.

5.6 Truth for other languages

The method used in the preceding sections for L_{PA} can be applied to a formal language provided that the language contains names for each member of the universe of discourse of the intended interpretation. If this condition does not hold, a difficulty arises in trying to define the truth of universal quantifications $\forall x F(x)$, since the truth of such a sentence is no longer specifiable in terms of the truth of its instances $F(c)$, where c is a term of the language. All we can say is that $\forall x F(x)$ is true iff $F(x)$ is true when x is assigned any value from the universe of discourse. But then we must provide a definition not just of truth but of the broader notion "truth under an assignment of values to the free variables". In this, we must treat all formulas of the language, not just sentences.

Let U be the universe of discourse of the intended interpretation of a formal language L. Assignments of values from U to the variables of language L may be identified with finite sequences of elements of U; such a sequence $\langle s_1, \ldots, s_k \rangle$ can be taken to assign s_1 to the alphabetically first variable of L, s_2 to the alphabetically second variable of L, ... , s_k to the alphabetically k^{th} variable of L. For variables later than the alphabetically k^{th}, let us take s_k, the last member of the sequence, as the assigned value. In other words, if σ is a finite sequence of members of U of length m, let $(\sigma)_i$ be the i^{th} member of σ if $i \leq m$ and let $(\sigma)_i = (\sigma)_m$ if $i > m$. We shall take σ to assign $(\sigma)_i$ to the alphabetically i^{th} variable of language L for every i.

We say that a finite sequence σ *satisfies* a formula F iff F is true under the assignment of values that σ encodes. Note that σ assigns values to all variables of the language; but if the alphabetically i^{th} variable of L does not occur free in F, then whether or not a sequence σ satisfies F will not depend on $(\sigma)_i$ (provided that i is less than the length of σ).

We seek an inductive definition of satisfaction. The only trick lies in the treatment of quantification. Now $\forall v F(v)$ is true under an assignment of values to variables just in case the formula $F(v)$ is true under every assignment that differs from the given one at most in what is assigned to the variable v. For in that case, $F(v)$ is satisfied no matter what value is assigned to v, while the values of the other variables remain fixed.

Let $\text{Sat}_{\text{At}}(\sigma, n)$ be the satisfaction relation for atomic formulas; i.e., it holds if n is the gödel number of an atomic formula of L that is satisfied by σ. Let Atform, neg, dis, gen, and Form be functions and relations that mirror the obvious syntactical operations and properties for language L; and for each i let var(i) be the number correlated with the alphabetically i^{th} variable of L. The inductive definition is then:

$$\begin{aligned}\mathrm{Sat}(\sigma,n) \equiv \mathrm{Form}(n) \,\&\, [&(\mathrm{Atform}(n) \,\&\, \mathrm{Sat}_{\mathrm{At}}(\sigma,n))\\ &\vee (\exists i<n)(n=\mathrm{neg}(i) \,\&\, \mathrm{Form}(i) \,\&\, \overline{\mathrm{Sat}}(\sigma,i))\\ &\vee (\exists i,j<n)(n=\mathrm{dis}(i,j) \,\&\, (\mathrm{Sat}(\sigma,i) \vee Sat(\sigma,j)))\\ &\vee (\exists i,k<n)(n=\mathrm{gen}(var(k),i)\\ &\qquad \& \, (\forall\sigma')((\forall j)((j\neq k \to (\sigma')_j=(\sigma)_j) \to \mathrm{Sat}(\sigma',i))].\end{aligned}$$

It remains to see how $\mathrm{Sat}_{\mathrm{At}}$ may be defined. Details here will depend on the vocabulary of the language L. Suppose, for example, that L contains, as nonlogical vocabulary, just finitely many predicate-signs $P_1,\ldots,P_m$, each of which is two-place. (Thus L contains no constants or function-signs.) Suppose the intended interpretation of L is given by a universe of discourse U and the m two-place relations $\Phi_1,\ldots,\Phi_m$ on U. Let $v_1,v_2,\ldots$ be the variables of L in alphabetic order. Then we would define

$$\begin{aligned}\mathrm{Sat}_{\mathrm{At}}(\sigma,\gamma(F)) \equiv [&(F \text{ has the form } P_1v_iv_j \,\&\, \Phi_1(\sigma)_i(\sigma)_j)\\ &\vee (F \text{ has the form } P_2v_iv_j \,\&\, \Phi_2(\sigma)_i(\sigma)_j)\\ &\vee \ldots \vee (F \text{ has the form } P_mv_iv_j \,\&\, \Phi_m(\sigma)_i(\sigma)_j)]\end{aligned}$$

If L contains function signs, then we must specify the value of the terms of L under assignments σ. For example, for L_{PA}, we would define a function $\mathrm{val}(\sigma,n)$ so that if n is the gödel number of a term, then $\mathrm{val}(\sigma,n)$ is the value of that term when the variables in it take the values assigned by σ. We would then define

$$\mathrm{Sat}_{\mathrm{At}}(\sigma,n) \equiv [\mathrm{Atform}(n) \,\&\, (\exists a)(\exists b)(n=a*2^5*b \,\&\, \mathrm{Val}(\sigma,a)=\mathrm{Val}(\sigma,b))].$$

In sum, an inductive definition of satisfaction for a formal language L can be formulated in a language that is adequate for talking about

(1) arbitrary finite sequences σ of members of U and their members;

(2) syntactic objects of L (by gödelization this can amount to nothing more than talking about numbers);

(3) the relations and functions that are the interpretations of the predicates and function signs of L.

Moreover, by a device similar to that of §**??**, the inductive definition can be converted into an explicit definition of Sat in a language that contains, in addition, quantification over relations between sequences of members of U and (the Gödel numbers of) formulas of L. That is, let $(\dagger_R)$ be the result of replacing, in the inductive definition of Sat above, all occurrences of "Sat" with the variable R. We can then define Sat explicitly by

$$\mathrm{Sat}(\sigma, n) \equiv (\exists R)((\dagger_R) \ \&\ R(\sigma, n))$$

or by

$$\mathrm{Sat}(\sigma, n) \equiv (\forall R)((\dagger_R) \to R(\sigma, n)).$$

Finally, given a definition of satisfaction, we can then define truth as follows

$$\mathrm{Tr}(n) \equiv (\forall \sigma)\mathrm{Sat}(\sigma, n).$$

Chapter 6

Computability

6.1 Computability

The notion of algorithm, or effective procedure, is an intuitive one that has been used in mathematics for a long time. An algorithm is simply a clerical procedure that can be applied to any of a range of inputs and will, on any input, yield an output. The basic idea is that an algorithm is a bunch of rules that can be applied mechanically; obtaining an output from any given input is just a matter of applying those rules (mindlessly, so to speak).

Before the 1930s, the notion was used in this intuitive sense. For example, as the tenth of the mathematical problems he formulated in 1900, Hilbert asked whether there is an algorithm which, if applied to any polynomial (containing any number of variables) with integer coefficients, determines whether or not there are integral values for the variables of the polynomial that give the polynomial the value 0. We ourselves have made intuitive use of the notion of algorithm. For example, in defining formal language and formal system, we said there must be an effective procedure for telling whether or not any given string of signs is a formula, and there must be an effective procedure for telling whether or not any given finite sequence of formulas is a derivation.

A number-theoretic function is said to be computable, in the intuitive sense, (or algorithmic) iff there is an algorithm for computing the function, i.e., an algorithm that yields, given any integers as inputs, the value of the function for these inputs as arguments. Our first aim in this unit is to give a precise mathematical definition of the notion of computable function.

Now we have seen a class of functions each of which is clearly computable, in the intuitive sense, namely, the primitive recursive functions. Perhaps one might think that this class exhausts the computable functions—that any function which we would intuitively call algorithmic is in fact primitive recursive. But this is not the case, as the following heuristic argument indicates.

First, the specification of the p.r. functions allows us to consider all p.r. definitions as being written in some standard symbolic form. We may then effectively list all p.r. definitions of one-place functions. Say this list is $D_1, D_2, D_3, \ldots$; and for each i let ψ_i be the p.r. function that D_i defines. (We can, for example, gödelize the p.r. definitions and then list them in increasing order of gödel number.) Now consider the following procedure: given input n, find the p.r. definition D_n. Then compute the value of ψ_n at argument n; since D_n is a p.r. definition of ψ_n, it tells us how to do this. Add one to this value of ψ_n; the result the output. What was just described is clearly an algorithm. This algorithm computes a function; call that function φ. *Then φ cannot be primitive recursive.* For suppose it were; then it would have a p.r. definition, and that definition would occur on our list, say as D_k. That is, we would have $\varphi = \psi_k$. But the value of φ at argument k is $\psi_k(k) + 1$; so φ cannot be identical with ψ_k. Thus φ is not p.r. But φ is computable.

This shows that the p.r. functions do not exhaust the computable functions. Indeed, our argument yields more: it shows that however we eventually define the notion of computable, it cannot be possible to list effectively algorithms that compute all and only the computable functions.

Our eventual definition of computable function will handle this pitfall as follows. We shall define a notion of "computing instructions", that is, a standard form of algorithm. A computable function is any function that can be computed by a computing-instruction. There will be an effective procedure for listing all the computing-instructions. But not all computing-instructions succeed in computing functions, and it will be impossible to "separate out", in an effective manner, just the computing instructions that do compute functions. This will become clearer when we see the details. In fact, we shall in fact give two explications of computability: the first uses formal systems of a particular sort, called "Herbrand-Gödel- Kleene systems" (also called the "equation calculus"); the second uses an abstract model of a computing machine, called a "Turing machine". It will turn out that these explications are equivalent.

6.2 Recursive and partial recursive functions

We shall first specify a formal language L_{HGK}. An *HGK-system* will be a formal system of a particular form in this language, and the notion of derivation in an HGK-system will be taken to correspond to the notion of computation.

Language L_{HGK}

Alphabet:	0	S	$=$	$($	$)$	$,$		
	x	y	z	x_1	y_1	z_1	$x_2 \ldots$	[formal variables]
	f	g	h	f_1	g_1	h_1	$f_2 \ldots$	[function letters]

The function letter f is called the *principal function letter*.

Formation rules:

1. 0 is a term; any formal variable is a term;
2. If t is a term then so is St;
3. If $t_1, t_2, \ldots, t_n$ are terms, $n > 0$, and δ is a function letter, then $\delta(t_1, t_2, \ldots, t_n)$ is a term;
4. Nothing else is a term.

If t and u are terms then $t = u$ is a formula. All formulas are called equations.

An HGK-system is simply a finite set of equations. We treat each HGK-system as a formal system: the notions of derivation and derivability in any such system E are determined by taking the members of E to be the axioms, and allowing just the following two rules of inference.

1. From an equation $t = u$ that contains a formal variable v may be inferred any equation that results from $t = u$ by replacing v with a formal numeral.
2. From an equation $t = u$ that contains no variables and an equation $\delta(\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r}$, where δ is a function letter, may be inferred any equation that results from $t = u$ by replacing one or more occurrences of $\delta(\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_n})$ with occurrences of $\boldsymbol{r}$.

If E is an HGK-system, we write $\vdash_E t = u$ for "the equation $t = u$ is derivable in the system E."

Definition. Let φ be an n-place function. φ is said to be *defined* by an *HGK-system* E iff, for all $p_1, p_2, \ldots, p_n$, and r,

$$\vdash_E f(\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r} \leftrightarrow \varphi(p_1, p_2, \ldots, p_n) = r.$$

The function φ is said to be *general recursive* (or, more briefly, *recursive*) iff it is defined by some HGK-system E.

The notion of general recursive function is what we shall adopt as our precise mathematical explication of the intuitive notion of computable function. Every general recursive function is computable in the intuitive sense. For suppose φ is defined by the HGK-system E. Then to compute $\varphi(p_1, \ldots, p_n)$ one simply makes an exhaustive search through the derivations in system E until one finds a derivation of an equation $f(\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r}$ for some r; r is then the value of $\varphi(p_1, \ldots, p_n)$. Since E defines φ , we are assured that there is such a derivation in system E.

Not every HGK-system defines a function. For example, suppose E contains the one equation $f(Sx) = SS0$. Then $\vdash_E f(\boldsymbol{p}) = \boldsymbol{2}$ for every $p > 0$, but no equation of the form $f(0) = r$ is derivable from E. We shall consider this phenomenon more closely two pages hence. For now, we are interested solely in HGK-systems that do define functions.

We start by investigating the extent of the general recursive functions.

Fact 6.1. *Every primitive recursive function is general recursive.*

Proof. It is a simple matter to formalize p.r. definitions by HGK-systems. For example, the system consisting of the following four equations defines multiplication: $g(x, 0) = x$, $g(x, Sy) = Sg(x, y)$, $f(x, 0) = 0$, $f(x, Sy) = g(f(x, y), x)$. □

Fact 6.2. *The class of general recursive functions is closed under composition.*

Proof. Obvious. □

Our third fact about the extent of the general recursive functions gives us something new: the possibility of specifying functions by use of the unbounded leastness operator μ.

Fact 6.3. *Let φ be an $(n+1)$-place general recursive function such that* $(\forall p_1)(\forall p_2) \ldots (\forall p_n)(\exists k)[\varphi(p_1, \ldots, p_n, k) = 0]$. *Then* $\mu k[\varphi(p_1, \ldots, p_n, k) = 0]$*, that is, the n-place function that carries $\langle p_1, \ldots, p_n \rangle$ to the least k such that $\varphi(p_1, \ldots, p_n, k) = 0$, is also general recursive.*

Proof. Let E be an HGK-system defining φ. Let E' be the result of relettering all function letters in E as letters f_i; in particular, let the principal function letter f be relettered as f_1. Let E^* be the HGK-system obtained by adding the following eight equations to E':

$$g_1(x, 0) = x$$

$$g_1(x, Sy) = Sg_1(x, y)$$

$$g_2(x, 0) = 0$$

$$g_2(x, Sy) = g_1(g_2(x, y), x)$$

$$g(x_1, \ldots, x_n, 0) = S0$$

$$g(x_1, \ldots, x_n, Sy) = g_2(f_1(x_1, \ldots, x_n, y), g(x_1, \ldots, x_n, y))$$

$$h(Sx, 0, y) = y$$

$$f(x_1, \ldots, x_n) = h(g(x_1, \ldots, x_n, y), g(x_1, \ldots, x_n, Sy), y)$$

We claim that the system E^* defines the function $\mu k[\varphi(p_1, \ldots, p_n, k) = 0]$, and hence that function is general recursive. To prove the claim it suffices to note the following:

1. $\vdash_{E^*} g_2(\boldsymbol{i}, \boldsymbol{j}) = \boldsymbol{k}$ iff $k = i \cdot j$.
2. $\vdash_{E^*} g(\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_n}, \boldsymbol{k+1}) = \boldsymbol{q}$ iff $q = \prod_{i=0}^{k} \varphi(p_1, \ldots, p_n, i)$.
3. $\vdash_{E^*} h(\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}) = \boldsymbol{m}$ iff $i \neq 0$, $j = 0$, and $k = m$.
4. $\vdash_{E^*} f(\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_n}) = \boldsymbol{k}$ iff
$$\textstyle\prod_{i=0}^{k-1} \varphi(p_1, \ldots, p_n, i) \neq 0 \text{ but } \prod_{i=0}^{k} \varphi(p_1, \ldots, p_n, i) = 0.$$

□

If the function φ has the property that $(\forall p_1)(\forall p_2) \ldots (\forall p_n)(\exists k)[\varphi(p_1, \ldots, p_n, k) = 0]$ then we say that the application of the unbounded leastness operator μ to φ is *licensed.* Thus Facts 6.1–6.3 tell us that the class of general recursive functions contains the primitive recursive functions and is closed under composition and licensed application of μ. In §**??** we shall show the converse: every general recursive function can be obtained from primitive recursive functions by composition and licensed application of μ.

As we pointed out above, not every HGK-system defines a function. For a system E to define an n-place function, there must exist for all $p_1, \ldots, p_n$ a unique r such that $\vdash_E f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r}$. This condition can fail in two ways: for some $p_1, \ldots, p_n$ there may be distinct q and r with $\vdash_E f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{q}$ and $\vdash_E f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r}$; or for some $p_1, \ldots, p_n$ there may be no r with $\vdash_E f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r}$.

The former problem may easily be avoided. We simply redefine the notion of derivability in an HGK-system as follows: we now say that an equation $f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r}$ is derivable in a system E iff, first, there is a derivation of it in E and, second, no smaller derivation is a derivation of $f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{q}$ for $q \neq r$ ("smaller" in the sense of a gödel numbering, which we assume has been fixed). This device—inspired, as should be obvious, by Rosser's proof of §**??**—makes it the case that for any $p_1, \ldots, p_n$ there is at most one integer r such that $\vdash_E f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r}$.

The latter problem, however, admits no such solution. Examples of HGK-systems in which for some $p_1, \ldots, p_n$ no equation $f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r}$ can be derived are easy to formulate. We have already seen a simple one. Here is another: let E contain just the equations $g(x, 0) = x$, $g(x, Sy) = Sg(x, y)$, $f(g(x, x)) = x$. Then $\vdash_E f(\boldsymbol{p}) = \boldsymbol{r}$ iff p is even and $r = p/2$. Thus E does not define a 1-place function. E does define something, though; namely, a function whose domain is just the even integers, and which takes each integer in its domain to one-half of that integer. Such a function is called a partial function.

Definition. An *n-place partial function* is an integer-valued function whose domain is some set of n-tuples of integers. If the domain of an n-place partial function φ is the set of all n-tuples, then φ is said to be *total.* (The domain of an n-place partial function may be all n-tuples, or it may be empty, or it may be something in between.)

We may now take it that every HGK-system E defines an n-place partial function for each $n > 0$, namely, the unique partial function φ such that for all $p_1, \ldots, p_n, r$,

$$\vdash_E f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r} \text{ iff } \varphi(p_1, \ldots, p_n) = r.$$

Thus the domain of φ is the set of n-tuples $\langle p_1, \ldots, p_n \rangle$ such that for some r $\vdash_E f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r}$. An n-place partial function φ is said to be *partial recursive* iff it is defined by some HGK-system. Thus, a general recursive function is simply a partial recursive function that is total.

We saw in the previous section that licensed application of the leastness operator μ to a general recursive function yields a general recursive function (Fact 6.3). The same proof shows that any application of μ—licensed or not—to a general recursive function yields a partial recursive function. That is, if φ is general recursive, then the function $\mu k[\varphi(p_1, \ldots, p_n, k) = 0]$, which takes $\langle p_1, \ldots, p_n\rangle$ to the least k such that $\varphi(p_1, \ldots, p_n, k) = 0$ if there is such a k and takes no value on $\langle p_1, \ldots, p_n\rangle$ if there is no such k, is partial recursive. That function will be total, and hence general recursive, just in case the application of μ is licensed.

Note: The leastness operator μ may also be applied to partial recursive functions that are not total; but here the definition must be phrased with care. Reflection on the proof of Fact 6.3 shows that, if that proof is to show $\mu k[\varphi(p_1, \ldots, p_n, k) = 0]$ to be partial recursive when φ is partial recursive, we should define

$$\mu k[\varphi(p_1, \ldots, p_n, k) = 0] = \begin{cases} \text{the least } k \text{ such that} & \\ \varphi(p_1, \ldots, p_n, k) = 0 & \text{if such a } k \text{ exists} \\ \text{and } \varphi(p_1, \ldots, p_n, j) \text{ has} & \\ \text{a valuefor each } j < k. & \\ & \\ \text{no value} & \text{otherwise.} \end{cases}$$

If μ is so defined, we have: the class of partial recursive functions is closed under application of μ. **End of Note.**

6.3 The Normal Form Theorem and the Halting Problem

First we show that all partial recursive functions can be obtained from primitive recursive functions by composition and application of μ. To do this, we gödelize the language L_{HGK}. Clearly this can be done so as to yield the following.

(I) Each HGK system is correlated with a finite set of integers, the gödel numbers of its axioms. These finite sets, in turn, can be correlated with integers, so that every integer corresponds to an HGK system and vice versa. We call the integer so correlated with an HGK-system E the index number of E.

(II) For each $n > 0$ there is a $(n + 2)$-place primitive recursive function Der_n such that $\text{Der}_n(e, p_1, \ldots, p_n, q) = 0$ if and only if q is the gödel number of a derivation in the HGK-system with index number e, and the last line of this derivation has the form $f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r}$ for some r; and $\mathrm{Der}_n(e, p_1, \ldots, p_n, q) = 1$ otherwise.

(III) There is a 1-place primitive recursive function Res such that if q is the gödel number of a sequence of equations the last of which has a formal numeral $\boldsymbol{r}$ on the right-hand side, then $\mathrm{Res}(q) = r$; and $\mathrm{Res}(q) = 0$ otherwise.

Normal Form Theorem. *Let φ be an n-place partial recursive function. Then there is a number e such that, for all $p_1, \ldots, p_n$,*

$$\varphi(p_1, \ldots, p_n) = \mathrm{Res}(\mu q[\mathrm{Der}_n(e, p_1, \ldots, p_n, q) = 0]).$$

Note that φ is total (and hence general recursive) iff the application of μ is licensed.

Proof. Since φ is partial recursive, it is defined by some HGK system E. Let e be the index number of E. Given any $p_1, \ldots, p_n$, let $k = \mu q[\mathrm{Der}_n(e, p_1, \ldots, p_n, q) = 0]$, if there is such a q. Then k is the smallest number that is the gödel number of a derivation from E whose last line has the form $f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r}$; and $\mathrm{Res}(k) = r$. But then $\vdash_E f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r}$; since E defines φ, r is the value of $\varphi(p_1, \ldots, p_n)$. If, on the other hand, there is no q such that $\mathrm{Der}_n(e, p_1, \ldots, p_n, q) = 0$, then no equation $f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r}$ is derivable in E; since E defines φ, φ takes no value on $\langle p_1, \ldots, p_n \rangle$. □

The Normal Form Theorem shows that every partial recursive function can be obtained by starting with a primitive recursive function, applying the μ-operator, and composing with a primitive recursive function. The partial function will be total, and hence general recursive, iff the application of μ is licensed.

Note: In the statement of the Normal Form Theorem and below, when we use "=" between two expressions for partial functions we mean: when both sides have values then those values are identical; and when one side takes no value, then neither does the other. **End of Note.**

As we've said, we take the notion of general recursive function to be the precise explication of the intuitive notion of computable function.

Church's Thesis—HGK Form. *A function is computable (in the intuitive sense) if and only if it is general recursive.*

Church's Thesis is not a mathematical claim. It asserts the equivalence of a mathematical notion and an intuitive one, and hence there can be no question of proof. Of course, there can be plausibility arguments (of a more or less philosophical nature). I have already argued for the direction "If general recursive then computable". For the converse, one might note the following. First, every particular function that people have ever encountered and judged on intuitive grounds to be computable has turned out to be general recursive. Second, there are in the literature other mathematical explications of the notion of computability (we'll treat one, namely, Turing-computability, in §**??**), and in each case it can be shown that the explications are equivalent, that is, each yields exactly the same class of functions. Third, if our intuitive notion of algorithm is something like that of a finite list of instructions applied to various inputs, then any precise notions of instruction and application of instructions should be gödelizable, and this will yield a result analogous to the Normal Form Theorem.

A partial recursive function is "semi-computable" in the following sense: given input $\langle p_1, \ldots, p_n \rangle$, we can systematically seek a derivation of $f(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}) = \boldsymbol{r}$ for some r; if there is such a derivation, we shall find it eventually; but if there is no such, we will go on forever. Of course, if the function is total then, no matter what input we are given, our computing will eventually stop.

It would be nicer not to have to deal with partial functions. One might hope to eliminate partial recursive functions that are not total; perhaps one could effectively weed out those HGK-systems that define nontotal functions. To do this for 1-place functions, one would need an effective procedure that would yield, given any HGK-system E, a "yes" if E defines a 1-place total function and a "no" if not. However, as we shall now see, no such effective procedure exists.

We shall speak of effective procedures whose inputs are index numbers, rather than HGK-systems. Such procedures can then be identified with general recursive functions (where we take output 1 to be "yes" and output 0 to be "no").

For any number e, we use φ_e for the 1-place partial recursive function defined by the HGK-system with index number e. For $n > 1$, we use φ_e^n for the n-place partial recursive function defined by the HGK-system with index number e.

Unsolvability of the Totality Problem. *There is no general recursive function ψ such that, for every integer e,*

$$\psi(e) = \begin{cases} 1 & \textit{if } \varphi_e \textit{ is total} \\ 0 & \textit{if } \varphi_e \textit{ is not total.} \end{cases}$$

Proof. Suppose such a ψ exists. By Facts 6.1 and 6.2, the function $\eta(e,q) = \psi(e) \cdot \mathrm{Der}_1(e,e,q)$ is general recursive. From the supposition about ψ, we have $\eta(e,q) = 0$ if either φ_e is not total or else φ_e is total and q is the gödel number of a derivation in the HGK-system with index number e of an equation $f(\boldsymbol{e}) = \boldsymbol{r}$ for some r. Hence $\forall e \exists q[\eta(e,q) = 0]$. By Fact 6.3, the function $\mu q[\eta(e,q) = 0]$ is general recursive. By Facts 6.1 and 6.2, the function $\delta(e) = \mathrm{Res}(\mu q[\eta(e,q) = 0]) + 1$ is general recursive. By the definition of Res we have, for every e,

$$\delta(e) = \begin{cases} \varphi_e(e) + 1 & \text{if } \varphi_e \text{ is total} \\ 1 & \text{if } \varphi_e \text{ is not total.} \end{cases}$$

Now δ, being general recursive, is identical to φ_{e_0} for some e_0, and φ_{e_0} is thus total. But then we have $\varphi_{e_0}(e_0) = \varphi_{e_0}(e_0) + 1$, a contradiction. (The reader should compare this proof to the heuristic proof about p.r. functions given in §**??**). □

Thus there is no effective way to weed out HGK-systems that fail to define total functions. Perhaps then, one could hope at least to "patch up" those HGK-systems that so fail. That is, if φ_e takes no value on p, why not just set the value equal to 0? But to do this, one would need an effective procedure for telling, for any e and any p, whether p is in the domain of φ_e or not. This too turns out to be impossible.

Unsolvability of the Halting Problem. *There is no 2-place general recursive function ψ such that*

$$\psi(e,p) = \begin{cases} 1 & \textit{if } \varphi_e \textit{ takes a value on } p \\ 0 & \textit{if } \varphi_e \textit{ takes no value on } p. \end{cases}$$

Proof. Suppose there were such a ψ. Then there exists a partial recursive function δ such that, for all e,

$$\delta(e) = \begin{cases} 0 & \text{if } \varphi_e(e) \text{ has no value} \\ \text{no value} & \text{if } \varphi_e(e) \text{ has a value.} \end{cases}$$

Namely, let $\delta(e) = \mu k(\psi(e,e) + k = 0)$. By the supposition, if e is not in the domain of φ_e then $\psi(e,e) = 0$, so that $\delta(e) = 0$; and if e is in the domain of φ_e then $\psi(e,e) = 1$ so that δ takes no value on e.

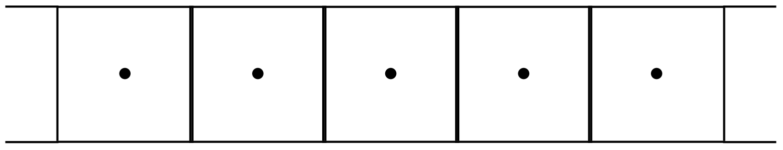
Now, since δ is partial recursive, it is φ_{e_0} for some e_0. By the specification of δ we then have: if e_0 is in the domain of φ_{e_0} then e_0 is not in the domain of δ, i.e., e_0 is not in the domain of φ_{e_0}; and if e_0 is not in the domain of φ_{e_0} then $\delta(e_0) = 0$ so that e_0 is in the domain of φ_{e_0} This is a contradiction. □

The unsolvability of the totality and halting problems show that we cannot eliminate nontotal partial recursive functions by any effective procedure. This is as we should have expected, given the heuristic argument of §**??**. The cost of capturing all computable functions by means of HGK-systems is that we cannot effectively avoid those HGK-systems that do not define total functions.

6.4 Turing Machines

In this section we give a seemingly rather different explication of the notion of computability, although as has been mentioned it will turn out to yield the same class of number-theoretic functions as the explication by the use of HGK-systems. One of the chief virtues of the Turing machine explication lies in its picturesque quality.

We think of a computing machine that operates on a two-way infinite tape. This tape is to serve as the means for getting input into the machine, the place where the machine prints its output, and also the place where the machine does its scratch-work. (The machine has no memory.) The tape is divided into cells, each of which can either be blank or else contain a symbol. Thus we may picture a tape thus:



where each dot represents either a blank or else a symbol of some sort, and the tape extends without limit in both directions from the depicted segment.

We may conceive of the machine as a mechanism that, at any moment, sits over a cell of the tape. The machine can scan the cell it sits over; it then takes the symbol (or blank) inscribed in that cell into account, and does something. The something it may do includes replacing the symbol with another and includes moving on—that is, shifting to the next cell to the right or to the next cell to the left. What the machine does at any point is determined by a finite list of instructions that we have given the machine. (Since we do not care about anything but the behavior of the machine, we may say that a machine is just its instructions.)

To be more precise, at any particular moment the machine is in one of a finite number of states. Each machine-instruction has the form: if in state i and the cell being scanned contains symbol t, then do so-and-so and go into state j. The so-and-so has two parts: the first is either erase the symbol t or replace t with t' or leave t as it is; the second is either stay put or move left one cell or move right one cell. In short, a Turing machine is specified by: first, a list of states, that is, a specification of how many states there are; second, a finite alphabet of symbols (including blank); and third, a finite list of instructions. Each instruction has the form of a quintuple

$$\langle i, t, t', X, j\rangle.$$

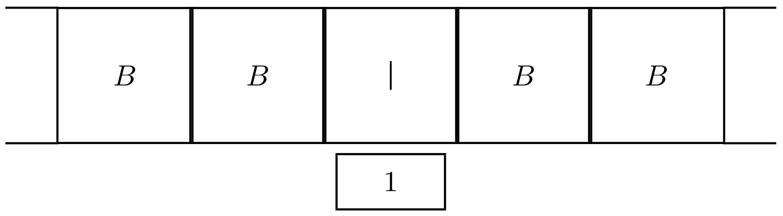
where i and j are numbers no greater than the number of states, t and t' are members of the alphabet, and X is either "D" (dont move), "L" (move left one), or "R" (move right one). The instruction may be read: if in state i and scanning a cell containing t, then replace t with t', move as X directs, and go into state j. To make the machine deterministic—that is, at each juncture there is at most one applicable instruction—we insist that no two instructions have the same first two members.

Given a Turing machine M, we may investigate the behavior of M when it is started in a particular state at a particular place on a given tape.

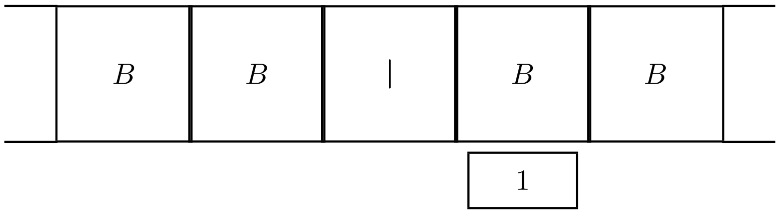
Example. Let M be the Turing machine that has 4 states, whose alphabet is just B (blank) and $|$ (stroke), and whose instructions are:

$$\langle 1, |, |, R, 1\rangle \qquad \langle 1, B, B, L, 2\rangle \qquad \langle 2, |, B, L, 3\rangle$$
$$\langle 3, B, |, D, 4\rangle \qquad \langle 3, |, |, D, 4\rangle$$

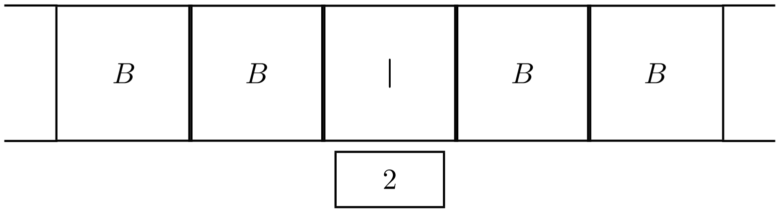
How does this machine work? Let us first consider what happens if the machine starts in state 1 situated at a cell containing a stroke and to the left and right of which are cells containing blanks. We may symbolize this initial situation thus:



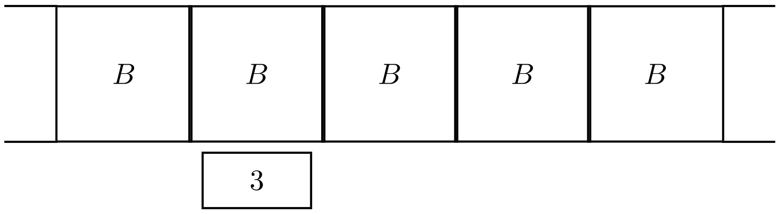
Since the machine is in state 1 and is scanning a stroke, the first instruction is applicable. Thus the machine leaves the stroke as is, moves right one cell, and remains in state 1. So heres how things look after this first move.



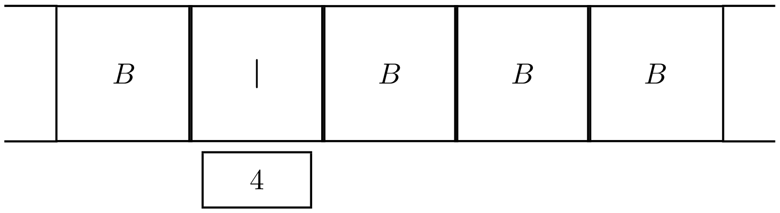
The second instruction is now applicable. So the machine moves to the left and goes into state 2. After this second move, we have:



The third instruction is now applicable. Hence the machine erases the stroke, moves left one, and goes into state 3.

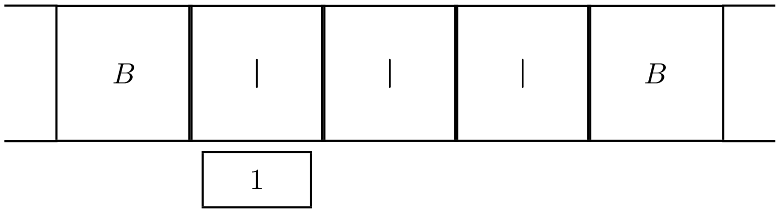


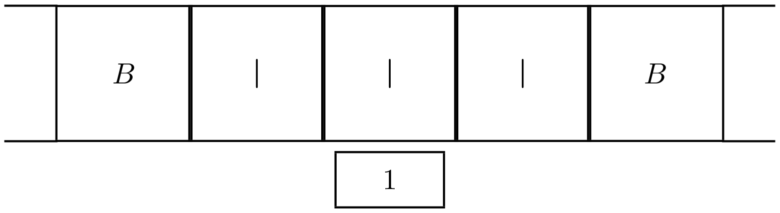
At this point the fourth instruction applies, so the machine writes a stroke in the

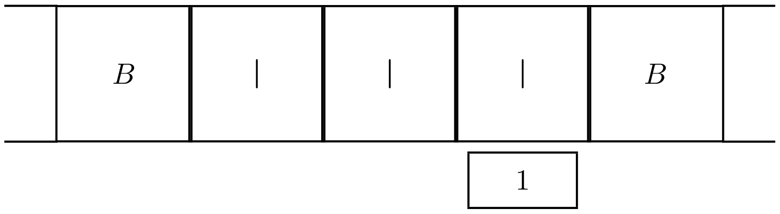


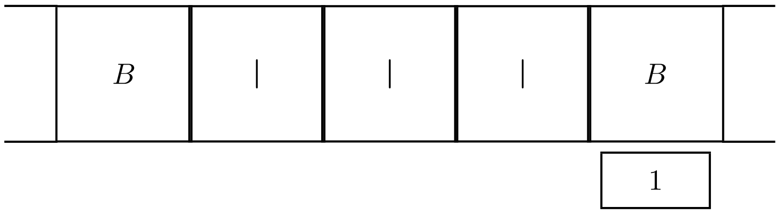
And then the machine halts; no instruction is applicable. In general, a machine halts iff it is in stage i, is scanning a cell containing t, and no machine-instruction has first two members i,t.

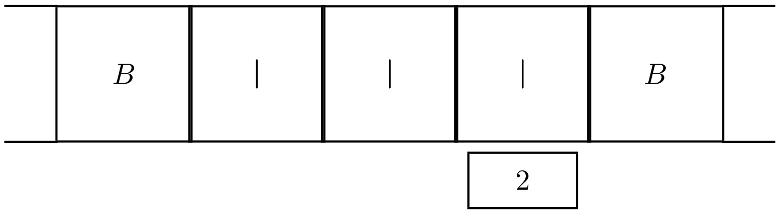
We now consider how the machine behaves if started in state 1 scanning a cell containing a stroke, to the right of which are two more cells containing strokes and then a cell containing a blank. Here are the initial situation and the ensuing ones:

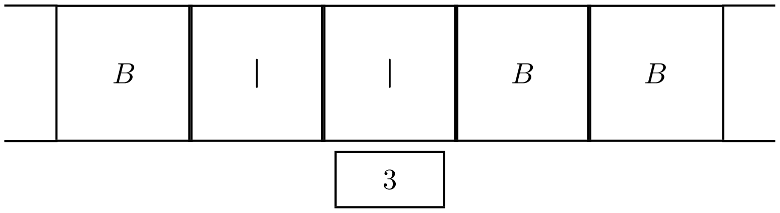


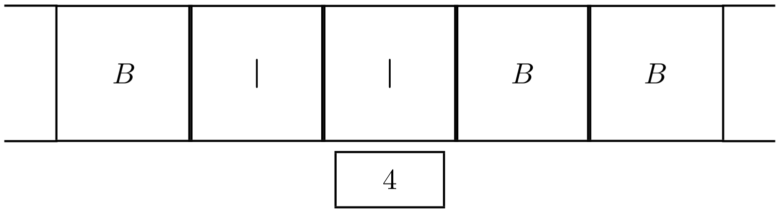












The machine halts after six steps, since it is in state 4 and scanning a cell containing a stroke.

In general, suppose M is started in state 1 scanning a cell containing a stroke to the right of which there are $n > 0$ cells containing strokes and then a cell containing a blank. The machine then moves to the right through all the cells that contain strokes until it encounters the cell that contains a blank; then it backs up, erases the last stroke, backs up once more, and halts. We may summarize its behavior more easily after we introduce some terminology.

A tape *represents* a number $n \geq 0$ iff the tape is blank but for $n+1$ consecutive cells each of which contains a stroke. To start a machine on input n is to start the machine in state 1 situated at the leftmost stroke in a tape representing n. A Turing machine yields m on input n iff when the machine is started on input n it eventually halts, and at the moment when it halts, the tape represents m.

Thus for each $n \geq 0$, the Turing machine M above yields Pred(n) on input n

(recall that Pred is the truncated predecessor function).

Here is a machine that, for each $n \geq 0$, yields $n+1$ on input n. M contains just the two instructions $\langle 1, |, |, R, 1\rangle$ and $\langle 1, B, |, D, 2\rangle$.

Definition. Let φ be a 1-place number-theoretic function. A Turing machine M *computes* φ iff, for each $n \geq 0$, M yields $\varphi(n)$ on input n. The function φ is *Turing-computable* iff some Turing machine computes φ.

It should be clear that every Turing-computable function is computable in the intuitive sense.

Now a Turing machine M may yield nothing on input n. For example, let M be the Turing machine with instructions $\langle 1, |, |, R, 2\rangle$ and $\langle 2, |, |, D, 2\rangle$. Then M yields 0 on input 0 and yields nothing (does not halt) on input n for $n > 0$.

Thus not every Turing machine computes a 1-place function. But every Turing machine does compute a partial function, if we define the notion of computing here as follows: Let φ be a 1-place partial function. Then M computes φ iff M yields $\varphi(n)$ on input n for each n in the domain of φ; and M does not halt on input n for each n not in the domain of φ. (We say M does not halt on input n iff M yields no value on input n. This slight misuse of terminology is picturesque, and promotes good intuitions.)

We may easily extend these notions to n-place functions and partial functions. Say that a tape represents an n-tuple $\langle p_1, \ldots, p_n\rangle$ of integers iff the tape is blank but for a sequence of consecutive cells, which contain the following configuration: first, $p_1 + 1$ consecutive strokes, then a blank, then $p_2 + 1$ consecutive strokes, then a blank, ..., then $p_n + 1$ consecutive strokes. The notion of yielding can then be defined as before. From this we obtain the notion of a Turing machines computing an n-place function or an n-place partial function. An n-place (partial) function is Turing-computable iff some Turing machine computes it.

By skillful programming, it can be shown that all primitive recursive functions are Turing-computable; that the Turing-computable partial functions are closed under composition and application of μ. It then follows from the Normal Form Theorem that all partial recursive functions are Turing-computable. Moreover, as we shall see in an Appendix to this section, Turing machines are specific sorts of formal systems; as such, they can be gödelized and a normal form theorem proved: every Turing-computable partial function can be obtained from primitive recursive functions by one application of μ and composition. It follows from this that all Turing-computable partial functions are partial recursive. Thus HGK-systems and Turing machines pick out the same classes of functions and partial functions.

Churchs Thesis—Turing Form. *A function is computable in the intuitive* *sense iff it is Turing-computable.*

Since the Turing-computable functions are exactly the general recursive functions, this form of Churchs Thesis is equivalent to the one on page **??**. However, some logicians—including Gödel—take the formulation in terms of Turing computability to be more directly supportable. They argue that Turing machines explicate not just the notion of computability, but also that of computation. Thus, this analysis provides the correct analysis of what we intuitively mean by "mechanical procedure".

Some evidence for this, one can argue, comes from the stability of the notion of Turing-computability. Although the kinds of instructions we allow Turing machines to have are rather limited, one can add other primitive forms of instructions, but in each case the new instructions can be simulated by instructions of the original sort. Or one can give the machines "memory", say by having any number of other tapes to work on, and instructions of the form: go to the n^{th} memory tape, and work on it. Anything such expanded machines can do, however, can be simulated by a machine of the restricted sort we have defined.

The Halting Problem. The Halting Problem for Turing machines is the problem of determining, given any Turing machine M and any integer p, whether or not M eventually halts if started on input p. Since Turing machines and HGK-systems are equivalent, it should occasion no surprise that the Halting Problem is not effectively solvable. A direct proof, completely in terms of Turing machines, is not difficult to frame.

Suppose we assign gödel numbers to Turing machines (this can be done, since machines are finite sets of quintuples). Now let us grant the following fact: for every Turing machine M there is a Turing machine N such that, for all p, if M yields 0 on input $\langle p, p\rangle$ then N yields 1 on input p, and otherwise N does not halt on input p. This fact can be shown by some reasonably simple computer-programming. It then follows quickly that:

> There is no Turing machine M such that, for all e and n, if the Turing machine numbered e halts on input n, then M yields 1 on input $\langle e, n\rangle$, and if the Turing machine numbered e does not halt on input n, then M yields 0 on input $\langle e, n\rangle$.

For suppose M were such a Turing machine. Let N be the machine provided by the fact granted above, and let d be the gödel number of N. From the specification of M we have that M yields 1 on input $\langle d, d\rangle$ iff N halts on input d; but from the definition of N we have that N halts on input d iff M yields 0 on input $\langle d, d\rangle$. Thus we obtain a contradiction, and we may conclude that no such M exists.

Appendix. Formal treatment. Lest the reader be carried away by the rather pictorial nature of the preceding section, we indicate here how Turing machines and their behavior may be defined more formally. Let S be a finite set of symbols, including "B" and "$|$", and let $q_1, q_2, \ldots$ be symbols not in S. Then a Turing machine M (on S) is simply a finite set of quintuples

$$\langle q_i, t, t', X, q_j \rangle,$$

where t and t' are in S and X is one of the symbols "D", "L", or "R", such that no two distinct quintuples have the same first two members. (The symbol q_i represents state i.)

We now seek to formalize the notion of "the situation of a machine at a given time". Note that in all our work we have been dealing with tapes that are blank in all but a finite number of cells. Thus all we need to encode is: what that finite stretch of tape that contains all the nonblank cells looks like; where the machine is (what cell it is scanning); and what state the machine is in. We can capture this by a notion of *instantaneous description* (id): an instantaneous description is any string of the form Pq_itQ, where t is a member of S and P and Q are (possibly empty) strings of symbols from S.

We now define the notion that encodes the operation of a machine M according to its instructions. Let I and J be id's. We say that I M-produces J iff either

1. There are (possibly empty) strings P and Q such that I is Pq_ktQ, J is $Pq_mt'Q$, and $\langle q_k, t, t', D, q_m \rangle$ is a quintuple in M;

or

2. There are (possibly empty) strings P and Q such that I is Pq_ktsQ, J is $Pt'q_msQ$, and $\langle q_k, t, t', R, q_m \rangle$ is a quintuple in M;

or

3. There is a (possibly empty) string P such that I is P_kt, J is $Pt'q_mB$, and $\langle q_k, t, t', R, q_m \rangle$ is a quintuple in M;

or

4. There are (possibly empty) strings P and Q such that I is Psq_ktQ, J is $Pq_mst'Q$, and $\langle q_k, t, t', L, q_m \rangle$ is a quintuple in M;

or

5. There is a (possibly empty) string Q such that I is $q_k tQ$, J is $q_m Bt'Q$, and $\langle q_k, t, t', L, q_m\rangle$ is a quintuple in M.

An M-computation is a finite sequence $I_1, \ldots, I_k$ of ids such that, for each $i < k$, I_i M-produces I_{i+1}. An M-computation is finished iff its last member is $Pq_m tQ$ for some P,Q and q_m, t are the first two members of no quintuple in M. Let $m, n \geq 0$. The Turing machine M yields p on input n iff there is a finished M-computation $I_1, \ldots, I_k$ such that

(a) I_1 is $q_1| \ldots |$, with $n + 1$ occurrences of $|$;

(b) I_k is $Pq_m Q$ for strings P, Q such that PQ contains $p + 1$ occurrences of $|$.

The notion of yielding on inputs that are n-tuples can be defined in similar fashion.

Thus we see that Turing machines can be treated as nothing more than (peculiar) types of formal systems. The point of this is, in part, simply to make clear that we may gödelize Turing machines and their behavior. That is, we may assign index numbers to Turing machines, and gödel numbers to instantaneous descriptions and to finite sequences of instantaneous descriptions, in such a way that the following holds:

1. There is, for each $n > 0$, a $(n + 2)$-place primitive recursive relation T_n such that $T_n(e, p_1, \ldots, p_n, q)$ iff e is the index number of a Turing machine M and q is the gödel number of a finished M-computation on input $\langle p_1, \ldots, p_n\rangle$.

2. There is a 1-place primitive recursive function Ans such that $\text{Ans}(q) = m$ iff q is the gödel number of a sequence of ids, the last id of which has the form $Pq_j Q$ such that PQ contains $m + 1$ strokes.

From this it then follows that for each n-place partial function φ that is computed by a Turing machine, there is an e such that

$$\varphi(p_1, ..., p_n) = \text{Ans}(\mu q[T_n(e, p_1, ..., p_n, q)]).$$

6.5 Undecidability

In this section we are concerned with applying recursive functions to the study of formal systems. One major issue is that of decidability. In §**??** we said that a formal system Σ is decidable iff there is a computational procedure for telling, given any formula of Σ, whether of not that formula is derivable in Σ. By use of Church's Thesis, we may make this a precise definition.

Definition. Let Σ be a formal system, which we assume gödelized via γ. A *decision procedure* for Σ is a recursive function φ such that, for each formula F,

$$\varphi(\gamma(F)) = \begin{cases} 1 & \text{if } \vdash_\Sigma F \\ 0 & \text{if not } \vdash_\Sigma F. \end{cases}$$

System Σ is (recursively) *decidable* iff there is a decision procedure for Σ. (**Warning:** Do not confuse the notion of decidability with that of formal decidability. The former applies to formal systems. The latter applies to formulas within formal systems. The use of the same word is merest coincidence.)

We shall be proving the undecidability of various systems. To obtain such results, we need to see how recursive functions may be numeralwise represented. Here the Normal Form Theorem is of great help.

Extended Representability Theorem. *Every recursive function is numeralwise representable in PA.*

Proof. As we showed in Chapter 4, the class of functions that are numeralwise representable in PA contains all primitive recursive functions and is closed under composition. By the Normal Form Theorem it suffices to show that this class of functions is closed under licensed application of μ, that is, if φ is an $(n+1)$-place (total) function that is numeralwise representable in PA and to which the application of μ is licensed, then $\mu k[\varphi(p_1, \ldots, p_n, k) = 0]$ is also numeralwise representable in PA.

For notational convenience we take the case $n = 2$. Let $\Phi(x, y, z, w)$ numeralwise represent φ; and let $u < v$ be the formula $u \leq v \,\centerdot\, {\sim} u = v$. Then let $F(x, y, z)$ be the formula

$$\Phi(x, y, z, 0) \,\centerdot\, \forall z'(z' < z \supset {\sim}\Phi(x, y, z, 0).$$

We claim that $F(x, y, z)$ numeralwise represents the function $\mu k[\varphi(p_1, p_2, k) = 0]$. Indeed, it is a routine matter to show that if $\mu k[\varphi(p_1, p_2, k) = 0] = q$, then

$$\vdash F(\boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{q}) \,\centerdot\, (F(\boldsymbol{p_1}, \boldsymbol{p_2}, z) \supset z = \boldsymbol{q}).$$

$\square$

Church's Theorem. If PA is consistent then PA is undecidable.

Proof. Let φ be any recursive function; we construct a formula G of L_{PA} that attests to the fact that φ is not a decision procedure for PA. By the Extended Representability Theorem, there is a formula $\Phi(x, y)$ that numeralwise represents φ in PA. By the Fixed Point Theorem there is a formula G of PA such that

$$\vdash_{\mathrm{PA}} G \equiv {\sim}\Phi(\ulcorner G \urcorner, S0).$$

Now either $\varphi(\gamma(G)) = 1$ or not. If $\varphi(\gamma(G)) = 1$ then $\vdash_{\mathrm{PA}} \Phi(\ulcorner G \urcorner, S0)$ so that $\vdash_{\mathrm{PA}} {\sim}G$; hence if PA is consistent then G is not derivable in PA. If $\varphi(\gamma(G)) \neq 1$, then $\vdash_{\mathrm{PA}} {\sim}\Phi(\ulcorner G \urcorner, S0)$, so that $\vdash_{\mathrm{PA}} G$. Hence, in either case, φ gives us the wrong answer on G. □

Note 1. The above proof should feel familiar. One could rephrase it thus: if PA were decidable, the Bew would be numeralwise representable in PA, by dint of the Extended Representability Theorem. But, by the Fixed Point Theorem, if PA is consistent, then Bew is not numeralwise expressible in PA. In other words, Gödel's work immediately tells us that there is no primitive recursive decision procedure for PA; and that work is extendible to any notion that will be representable in PA. All that was necessary after 1931, then, was to formulate the appropriate general notion of computability, and show that all computable functions were numeralwise representable.

Note 2. We have shown that for every general recursive function φ there is a formula G such that: either G is derivable and $\varphi(\gamma(G)) \neq 1$ or else ${\sim}G$ is derivable and $\varphi(\gamma(G)) = 1$. This shows that there is no way of extending PA to a system that is both consistent and decidable. PA is therefore said to be *essentially undecidable*.

The proof of Church's Theorem relies only on the fact that every recursive function is numeralwise representable in PA, and on the Fixed Point Theorem (and, of course, the Fixed Point Theorem holds provided that the function diag is numeralwise representable). Every consistent formal system in which all recursive functions are numeralwise representable is thus essentially undecidable. We now show that even systems considerability weaker than PA are essentially undecidable.

Definition. Let Q be the formal system whose language is L_{PA}, and whose axioms are like those of PA except that the axiom-schema of induction is eliminated and, in its stead, the axiom ${\sim}x = 0 \supset \exists y(x = Sy)$ is added.

System Q is often called *Robinson arithmetic*, after Raphael Robinson, who first formulated the system in 1950. Q is a very weak system, because of the absence of induction axioms. Even quite elementary truths like $\forall x(0+x = x)$ are not derivable in it. Nonetheless,

Robinson's Lemma. *Every recursive function is numeralwise representable in Q.*

Proof. The inclusion of the axiom $\sim x = 0 \supset \exists y(x = Sy)$ yields the derivability in Q of the formulas $x \leq \boldsymbol{m} \supset x = \mathbf{0} \vee x = \mathbf{1} \vee \ldots \vee x = \boldsymbol{m}$ and $x \leq \boldsymbol{m} \vee \boldsymbol{m} \leq x$ for every m. A close analysis of the proof of the Representability Theorem given in §**??** shows that these properties of $\leq$, together with the facts that $x + y$ numeralwise represents addition in Q, and $x \times y$ numeralwise represents multiplication in Q, yield the representability of all primitive recursive functions in Q. From that, the representability of all general recursive functions in Q follows by the same argument as was used above for PA. □

From Robinson's Lemma it follows that system Q is essentially undecidable. Now let Q has only finitely many nonlogical axioms (seven, in fact). From these two facts, we may prove

Church-Turing Theorem. *There is no decision procedure for quantificational validity.*

Proof. Let A be the conjunction of the universal closures of the seven non-logical axioms. By the Deduction Theorem we have:

> A formula F is derivable in Q iff the formula $(A \supset F)$ is derivable using just the logical axioms (i.e., the truth-functional, quantificational, and identity axioms).

From the undecidability of Q, we then have: there is no effective procedure that determines, given any formula F in L_{PA}, whether or not $(A \supset F)$ is derivable using just the logical axioms.

Now formulas $(A \supset F)$ are not in the language of pure quantification theory: they contain function signs (namely, "S", "$+$", and "$\times$") and constants ("0"). However, by adjoining some new predicate letters and using definite descriptions, one can effectively find a quantificational schema $(A \supset F)^*$ that is in this language and that is derivable from logical axioms iff $(A \supset F)$ is so derivable.

But then there can be no effective procedure that decides quantificational validity. For such a procedure would yield a procedure that decides, given any formula F of L_{PA}, whether or not $(A \supset F)^*$ is derivable using logical axioms only; from this, one could obtain a decision procedure for Q. □

6.6 Recursive and recursively enumerable sets

A set of integers is said to be recursive iff its characteristic function is general recursive. (Recall that the characteristic function of a set S is the function that takes n to 1 if $n \in S$ and takes n to 0 if $n \notin S$.) Thus, a set is recursive iff there is an effective procedure that determines, for any integer, whether or not that integer is in the set.

It should be clear that the complement of a recursive set is recursive, and that the union and the intersection of recursive sets are recursive. Since every general recursive function is numeralwise representable in formal system PA, every recursive set is numeralwise representable in PA; the same holds for formal system Q.

A formal system is decidable iff the set of gödel numbers of formulas derivable in the system is recursive. Thus the set of gödel numbers of formulas derivable in PA is not recursive. There is, however, a search procedure for this set, that is, an effective procedure that, applied to any integer n, eventually terminates if n is in the set but does not terminate if n is not in the set. We need merely look through the integers, consecutively, and terminate when and if we find an integer that is the gödel number of a derivation in PA of the formula with gödel number n. Sets for which there exists a search procedure are called recursively enumerable.

Definition. A set of integers is *recursively enumerable* (r.e.) iff it is the domain of some partial recursive function.

The rubric "recursively enumerable" reflects the fact that such a set can be exhaustively listed in an effective manner, that is, there exists a general recursive function g such that $g(0), g(1), g(2), \ldots$ lists all and only the members of the set (possibly with repetitions). Let us prove this.

R.e. Fact 1. *A set S is recursively enumerable iff either S is empty or else S is the range of some general recursive function.*

Proof. ($\rightarrow$) Suppose S is r.e. Thus $S = \text{domain}(\varphi_e)$ for some e. If S is empty, we are done. Otherwise, let k_0 be a member of S. We construct a general recursive function that takes an integer p to n if p is the gödel number of a computation (i.e. derivation) of a value for $\varphi_e(n)$, and takes p to k_0 otherwise. In fact, let

$$g(p) = \begin{cases} \mu n[n \leq p \ \&\ \text{Der}_1(e, n, p) = 0] & \text{if there exists such an } n \leq p \\ k_0 & \text{otherwise.} \end{cases}$$

The g is clearly recursive (indeed, p.r.), and $\text{range}(g) = \text{domain}(\varphi_e) = S$.

($\leftarrow$) If S is empty then S is the domain of the partial recursive function that is nowhere defined. If $S = \text{range}(g)$, where g is general recursive, let $\psi(n) = \mu p[g(p) = n]$. Then ψ is partial recursive, and $\psi(n)$ is defined iff $n \in \text{range}(g)$. That is, $\text{domain}(\psi) = \text{range}(g) = S$. □

R.e. Fact 2. *If a set and its complement are both r.e., then the set is recursive.*

Proof. The intuitive idea is this: suppose there are search procedures for S and for $\overline{S}$. Given n, start both search procedures on n; since either $n \in S$ or $n \in \overline{S}$, at some point one of the search procedures will terminate. If the search procedure for S terminates, we know $n \in S$; if that for $\overline{S}$ terminates, we know $n \notin S$. Thus we have a decision procedure for membership in S.

More rigorously, suppose $S = \text{domain}(\varphi_d)$ and $\overline{S} = \text{domain}(\varphi_e)$. Let $\psi(n) = \mu p[Der_1(d, n, p) \cdot Der_1(e, n, p) = 0]$. Since the application of μ is licensed, ψ is general recursive. Let $g(n) = \alpha(Der_1(d, n, \psi(n)))$. Then g is general recursive. Moreover, if $n \in S$ then $\text{Der}_1(d, n, \psi(n)) = 0$, so that $g(n) = 1$. If $n \notin S$ then $\text{Der}_1(d, n, \psi(n)) \neq 0$, so that $g(n) = 0$. Thus g is the characteristic function of S. □

R.e. Fact 2 can be extended to k-place relations for $k > 1$, if we extend our definitions in the obvious way: a k-place relation is recursive iff its characteristic function is general recursive, and is recursively enumerable iff it is the domain of some k-place partial recursive function. The following result relates r.e. sets to 2-place recursive relations. It can easily be extended to relate k-place r.e./ relations to $(k + 1)$-place recursive relations.

R.e. Fact 3 *A set is r.e. iff there exists a 2-place recursive relation R such that, for each n, $n \in S$ iff $(\exists p)R(n, p)$.*

Proof. ($\rightarrow$) Suppose $S = \text{domain}(\varphi_e)$. Then $n \in S$ iff $(\exists p)(\text{Der}_1(e, n, p) = 0)$, and, for any e, $\text{Der}_1(e, n, p) = 0$ is a recursive relation of n and p. (Indeed, it is primitive recursive.)

($\leftarrow$) Let R be a 2-place recursive relation, and let S be the set of integers n such that $(\exists p)R(n, p)$. Let χ be the characteristic function of R; thus χ is general recursive, so that the function $\psi(n) = \mu p[\chi(n, p) = 1]$ is partial recursive. Clearly $n \in S$ iff $n \in \text{domain}(\psi)$; hence S is r.e. □

Thus, a set is r.e. iff it can be obtained from a recursive relation by existential quantification.

Other r.e. Facts.

(a) *If two sets are r.e., then so are their union and their intersection.*

(b) *If ψ is any partial recursive function and k is any integer, then $\{n \mid \psi(n) = k\}$ is r.e.*

(c) *Every recursive set is r.e.*

(d) *A set is r.e. iff it is the range of some partial recursive function.*

(e) *If R is a 2-place r.e. relation, then $\{n \mid (\exists p)R(n, p)\}$ is an r.e. set.*

The proofs are left to the reader.

We now investigate relations between these recursion-theoretic notions and formal systems. Let Σ be a formal system. We presume that Σ can be gödelized and that, in particular, Der_Σ is a recursive 2-place relation that mirrors "derivation of" for system Σ.

Claim 1. *The set of gödel numbers of formulas derivable in Σ is r.e.*

Proof. n is in this set iff $(\exists m)\mathrm{Der}_\Sigma(m, n)$. The result thus follows by R.e. Fact 3. □

Claim 2. *For any formula $F(x)$, $\{n \mid \vdash_\Sigma F(\boldsymbol{n})\}$ is r.e.*

Proof. For each n, let $h(n)$ be the gödel number of $F(\boldsymbol{n})$; h is primitive recursive, and hence general recursive. By Mirroring, $\vdash_\Sigma F(\boldsymbol{n})$ iff $(\exists m)\mathrm{Der}_\Sigma(m, h(n))$. The result then follows by R.e. Fact 3. □

Claim 3. *Suppose Σ is consistent. Then every set that is numeralwise representable in Σ is recursive.*

Proof. Let $F(x)$ numeralwise represent a set S in Σ. Since Σ is consistent, $S = \{n \mid \vdash_\Sigma F(\boldsymbol{n})\}$ and $\overline{S} = \{n \mid \vdash_\Sigma {\sim}F(\boldsymbol{n})\}$. By Claim 2, S and $\overline{S}$ are both r.e. By R.e. Fact 2, S is recursive. □

Claim 2 shows that every set weakly representable in Σ is r.e. Claim 3 shows that each of the formal systems PA, SA, and Q numeralwise represent the same sets (assuming they are all consistent), to wit, the recursive sets. Moreover, assuming these systems are ω-consistent, they all weakly represent the same sets, to wit, the recursively enumerable sets.

Claim 4. *Suppose Σ contains the usual quantifier rules. If Σ is syntactically complete, then Σ is decidable.*

Proof. Let Σ be syntactically complete. We may assume Σ consistent, for if it is not then it is trivially decidable. Let A be the set of gödel numbers of sentences derivable in Σ, let B be the set of gödel numbers of sentences refutable in Σ, and let C be the set of integers that are not gödel numbers of sentences. A and B are r.e., and C is recursive. By completeness, if $n \notin A$ then $n \in B \cup C$. By consistency, if $n \in A$ then $n \notin B \cup C$. Hence $\overline{A} = B \cup C$. Thus both A and $\overline{A}$ are r.e., so A is recursive.

Since Σ contains the quantifier rules, it includes universal instantiation and generalization. Hence a formula is derivable in Σ iff its universal closure is derivable. Let u take the gödel number of any formula to the gödel number of its universal closure, and take other numbers to themselves. Then n is the gödel number of a formula derivable in Σ iff $u(n) \in A$. Consequently, the set of gödel numbers of formulas derivable in Σ is recursive; i.e., Σ is decidable. □

In §**??** we showed that if PA is consistent then it is undecidable. This, together with Claim 4, yields Rosser's strengthening of Gödel's Theorem, i.e., if PA is consistent then it is syntactically incomplete. Such a proof of incompleteness by way of undecidability, however, does not actually provide a sentence that is neither derivable nor refutable.

6.7 Recursive Function Theory

In this section we investigate the recursive and partial recursive functions more intrinsically, with an eye to showing basic undecidabilities that arise within the theory of recursive functions. We have already shown two such results, namely, the unsolvability of the Totality and Halting Problems. We shall see many more.

A basic tool in this is the Enumeration Theorem, which gives us the existence of universal Turing Machines (or universal HGK-systems), in the following sense: There is a Turing Machine M such that, on any input $\langle e, n\rangle$, M yields the same thing that the Turing Machine numbered e yields on input n. That is, we can conceive of M as having the capacity to follow these instructions: given e and n, pretend you're the Turing Machine with index number e, and calculate at input n. So M is universal, in the sense that it can do everything that any Turing Machine can do. Which is to say: a finite list of instructions suffices to capture all partial recursive functions.

One can put this into HGK-system language too: there is one HGK-system that "incorporates" all HGK-systems.

Enumeration Theorem. *For each $n > 0$ there is a universal partial recursive function U_n of $n+1$ arguments; that is, for all integers $e, p_1, \ldots, p_n$,*

$$U_n(e, p_1, \ldots, p_n) = \varphi_e^{(n)}(p_1, \ldots, p_n).$$

Special case $n = 1$: There is a 2-place partial recursive function U_1 such that, for all e and p, $U_1(e, p) = \varphi_e(p)$.

Proof. Simply let $U_n(e, p_1, \ldots, p_n) = \mathrm{Res}(\mu k[\mathrm{Der}_n(e, p_1, \ldots, p_n, k) = 0]$. □

The Enumeration Theorem allows us to obtain negative results. We exploit the Theorem to define partial recursive functions that take index numbers as arguments and, for each value of this argument, simulate the indexed function. This yields quick diagonal arguments, of which we give two.

Result 1. *There exists a one-place partial recursive function ψ such that no general recursive function agrees with ψ on all arguments at which ψ is defined.*

Proof. Define ψ thus: for each m, $\psi(m) = \varphi_m(m) + 1$ (as usual, with the convention that if the right side is undefined then so is the left). Then ψ is partial recursive, since $\psi(m) = U_1(m, m) + 1$, and thus ψ comes from the universal function U_1 by composition with the successor function.

Now let φ_e be any 1-place general recursive function. By definition of ψ, $\psi(e) = \varphi_e(e) + 1$. Since φ_e is total, $\varphi_e(e)$ is defined; thus $\psi(e)$ is defined and $\psi(e) \neq \varphi_e(e)$. Hence no general recursive function agrees with ψ at all places at which ψ is defined. (The sharp-eyed reader will have noted the similarity between this proof and that of the Unsolvability of the Totality Problem, page **??** above.) □

Result 2. *Let $K = \{e \mid \varphi_e(e)$ is defined$\}$. Then K is recursively enumerable, but not recursive.*

Proof. Let $\psi(p) = U_1(p, p)$. Then ψ is partial recursive, and its domain is precisely K. Hence K is recursively enumerable. To show that K is not recursive it suffices to show that $\overline{K}$ is not recursively enumerable. Suppose $\overline{K}$ were the domain of a partial recursive function φ_m. Then $m \in \overline{K}$ iff $\varphi_m(m)$ is defined. Thus $m \in \overline{K}$ iff $m \in K$, by the specification of K. This is a contradiction. □

The nonrecursiveness of K yields another proof of the Unsolvability of the Halting Problem. For let $K = \text{domain}(\varphi_d)$; then the Halting Problem for φ_d is undecidable. A fortiori, the Halting Problem for all partial recursive functions is undecidable.

The theorem we now present, in a sense, goes in the opposite direction from the Enumeration Theorem. The latter allows one to treat index numbers as arguments, whereas the theorem below allows one to take operations on arguments to amount to operations on index numbers.

Uniformization Theorem. *Let ψ be any 2-place partial recursive function. Then there is a general recursive function h such that, for all e and n,*

$$\varphi_{h(e)}(n) = \psi(e, n).$$

Proof. Consider the following syntactic operation on the HGK-system that defines ψ: given an integer e, first reletter the function letter f as f_k, for k large enough to avoid conflicts; then add the equation $f(x) = f_k(\boldsymbol{e}, x)$. Clearly, the resulting HGK system defines the partial function that takes each n to $\psi(e, n)$. Moreover, by gödelization, the function that takes e to the index number of the resulting HGK-system is general recursive (indeed, primitive recursive). That function is the desired h. $\square$

Note. There are forms of the Uniformization Theorem for more arguments. E.g., let ψ be a 3-place partial recursive function. Then there exists a 2-place general recursive g such that $\varphi_{g(d,e)}(n) = \psi(d, e, n)$ for all d, e, and n. In the literature, the general form of the Uniformization Theorem is called the "*s*–*n*–*m* Theorem".

The Uniformization Theorem is extremely useful in establishing the nonrecursiveness of various sets of index numbers. We use it to establish reducibilities.

Definition. Let A and B be sets of integers. A is *many-one reducible* to B (in symbols $A \leq_m B$) iff there exists a general recursive function h such that, for each n, $n \in A$ iff $h(n) \in B$.

If $A \leq_m B$. then the question of membership in A is reducible to the question of membership in B: if one knew how to decide membership in B, one would then know how to decide membership in A.

Lemma. *If $A \leq_m B$ and B is recursive, then A is recursive. If $A \leq_m B$ and B is recursively enumerable, then A is recursively enumerable.*

Proof. Let h be a general recursive function such that, for all n, $n \in A$ iff $h(n) \in B$. If B is recursive, then the characteristic function χ of B is recursive; and the characteristic function of A is the composition of the characteristic function of B and h, and so is itself recursive. Hence A is a recursive set. If B is recursively enumerable then it is the domain of some φ_e. Now the partial function that takes each n to $\varphi_e(h(n))$ is partial recursive; and A is its domain. Hence A is recursively enumerable. □

In the applications below of the Lemma, we use its contrapositive form: if $A \leq_m B$ and A is not recursive, then B is not recursive, and similarly for recursive enumerability. We often use the set K, that is, $\{e \mid \varphi_e(e) \text{ is defined}\}$, which was shown not to be recursive above. Note that, by R.e. Fact 2, $\overline{K}$ is not recursively enumerable. The needed recursive function h is obtained by means of the Uniformization Theorem.

Result 3. *None of the following sets is recursive:*

$$\{e \mid \textit{0 is in the domain of } \varphi_e\};$$
$$\{e \mid \textit{the domain of } \varphi_e \textit{ is not empty}\};$$
$$\{e \mid \textit{the domain of } \varphi_e \textit{ is infinite}\};$$
$$\{e \mid \varphi_e \textit{ is a total constant function}\}.$$

Proof. For any e and n, let $\psi(e,n) = \varphi_e(e)$. By the Enumeration Theorem, ψ is partial recursive. By the Uniformization Theorem, there exists a general recursive function h such that, for all e and n, $\varphi_{h(e)}(n) = \psi(e,n)$. Thus we have:

if $e \in K$ then $\varphi_{h(e)}$ is defined everywhere, and is constant;
if $e \notin K$ then $\varphi_{h(e)}$ is defined nowhere.

Thus, if S is any of the sets listed in the statement of the result, $e \in K$ iff $h(e) \in S$. The result follows by the Reduction Lemma. □

Result 4. $\{e \mid \varphi_e$ *has infinite range*$\}$ *is not recursive.*

Proof. Let $\psi(e,n) = \varphi_e(e)+n$. By the Enumeration Theorem, ψ is partial recursive. By the Uniformization Theorem, there exists a general recursive function g such that $\varphi_{g(e)}(n) = \psi(e,n)$. Thus if $e \in K$ then $\varphi_{g(e)}$ has infinite range;

if $e \notin K$ then $\varphi_{g(e)}$ is defined nowhere, and hence does not have infinite range.

Hence $e \notin K$ iff $\varphi_{g(e)}$ has infinite range, and we are done. $\square$

Result 5. *Let* Tot $= \{e \mid \varphi_e$ *is total*$\}$. *Neither* Tot *nor* $\overline{\text{Tot}}$ *is recursively enumerable.*

Proof. The recursive function h obtained in the proof of Result 3 has the property that $e \in K$ iff $h(e) \in \text{Tot}$. Thus $e \in \overline{K}$ iff $h(e) \in \overline{\text{Tot}}$, so that $\overline{K} \leq_m \overline{\text{Tot}}$. If $\overline{\text{Tot}}$ were r.e., then, by the Reduction Lemma, $\overline{K}$ would be r.e. But in the proof of Result 2 above, we showed that $\overline{K}$ is not r.e. Hence $\overline{\text{Tot}}$ is not r.e.

To show Tot is not r.e., we proceed in a manner similar to the proof of the Unsolvability of the Totality Problem. Suppose Tot is r.e. Thus there exists a recursive function g such that $\text{Tot} = \text{range}(g)$. Let $\psi(n) = \varphi_{g(n)}(n) + 1$. ψ is partial recursive, by the Enumeration Theorem. ψ is total, since for each n $\varphi_{g(n)}$ is total. Let d be an index number for ψ, i.e., let $\psi = \varphi_d$. Since ψ is total, $d \in \text{range}(g)$. Let e be such that $d = g(e)$. Then $\varphi_d(e) = \varphi_{g(e)}(e)$, but also $\varphi_d(e) = \psi(e) = \varphi_{g(e)}(e)+1$. This is a contradiction. $\square$

The above proof shows how to obtain, given any r.e. subset of Tot, a recursive function no index for which lies in the subset. This can be used to prove the following striking result: for any ω-consistent system there are total recursive functions that cannot be proved to be total in the system. For let Σ be an ω-consistent formal system, and let S be the set of integers e such that the formalization of $(\forall p)(\exists q)(\text{Der}_1(e, p, q) = 0)$ is derivable in Σ. By Claim 2 of §**??**, S is r.e. By the ω-consistency of Σ, if that formalization is derivable in Σ then φ_e is total. Hence S is an r.e. subset of Tot, and there exists a recursive function ψ no index for which is in S. That is, for all d, if $\psi = \varphi_d$ then $d \notin S$. Thus ψ cannot be proved in Σ to be total.