CHAPTER 4

INCOMPLETENESS AND UNDECIDABILITY

This is the main part of this (or any other) first course in logic, in which we will establish and explain the fundamental incompleteness and undecidability phenomena of first order logic due (primarily) to Gödel. There are three "waves" of results, each requiring a little more technique than the preceding one and establishing deeper and more subtle facts about first order logic.

4A. Tarski and Gödel (First Incompleteness Theorem)

The key to Gödel theory is the method of **coding** (or *arithmetization*), which makes it possible to express properties of formulas, sentences and proofs within number theory. Here we will define these codings, establish their basic properties and use them to derive the simplest (and most basic) incompleteness results.

Recall from the proof of Theorem 1J.5 the function $n \mapsto \Delta n$ from natural numbers to terms of the language of Peano Arithmetic PA which is defined by the recursion

$$\Delta 0 \equiv 0, \quad \Delta(n+1) \equiv S(\Delta n).$$

We also set $1 \equiv \Delta 1 \equiv S(0)$, to avoid the annoying notation $\Delta 1$. These numerals provide names for all numbers and allow us to reduce satisfiability of formulas to truth of sentences for the structure $\mathbf{N}$:

Lemma 4A.1. *For every full extended formula $\phi(v_1, \dots, v_n)$ in the language of arithmetic and all number $x_1, \dots, x_n$,*

$$\mathbf{N} \models \phi[x_1, \dots, x_n] \iff \mathbf{N} \models \phi(\Delta x_1, \dots, \Delta x_n). \tag{83}$$

PROOF. First we show by induction that for every full extended term $t(v_1, \dots, v_n)$ and any numbers $x_1, \dots, x_n$, in the notation introduced at the end of Section 1C,

$$\text{if } t^{\mathbf{N}}[x_1, \dots, x_n] = w, \text{ then } \mathbf{N} \models t(\Delta x_1, \dots, \Delta x_n) = \Delta w,$$

and then we prove the lemma by induction on formulas. ⊣

We will be using without comment synonymously the expressions at the two sides of (83), as it suits the purpose at hand.

Definition 4A.2 (Coding)**.** Let τ be a finite signature with k (relation, constant and function) symbols $s_1, \dots, s_k$. We assign numbers to the *symbols* of $\mathbb{FOL}(\tau)$ by enumerating them as follows,

$$\neg \ \rightarrow \ \ \& \ \ \vee \ \forall \ \exists \ = \ (\ )\ ,\ s_1 \ \dots \ s_k \ \mathrm{v}_0 \ \mathrm{v}_1 \ \dots$$

so that the code $\#\neg$ of $\neg$ is 0 and the code $\#s_1$ of the first symbol of τ is 10. The v_i are the variables, and $\#\mathrm{v}_i = 10 + k + i$. We code *strings of symbols* using the standard coding of tuples,

$$\#(a_1 a_2 \dots a_n) = \langle \#a_1, \dots, \#a_n \rangle;$$

and we code *finite sequences of strings* using the same idea.

We code terms and formulas of $\mathbb{FOL}(\tau)$ by viewing them as strings of symbols.

Lemma 4A.3 (The Substitution Lemma)**.** *Fix a finite signature τ and set*

$$\begin{aligned} \mathrm{Term}_0(a) &\iff_{\mathrm{df}} a \text{ is a code of a term } t, \\ \mathrm{Formula}_0(e) &\iff_{\mathrm{df}} e \text{ is a code of a formula } \phi, \\ \mathrm{Formula}(e,i,j) &\iff_{\mathrm{df}} e \text{ is a code of a formula } \phi \\ &\qquad \text{and } \mathrm{v}_i \text{ occurs free as the } j\text{th symbol of } \phi. \end{aligned}$$

(a) *The relations* $\mathrm{Term}_0(a)$, $\mathrm{Formula}_0(e)$ *and* $\mathrm{Formula}(e,i,j)$ *are primitive recursive.*

(b) *There is a primitive recursive function* $\mathrm{sub}(e,i,a)$*, such that if a term t is free for v_i in an extended formula $\phi(\mathrm{v}_i)$, then*

$$\mathrm{sub}(\#\phi(\mathrm{v}_i), i, \#t) = \#\phi(t). \tag{84}$$

PROOF. (a) The characteristic functions of these relations are defined by Complete Primitive Recursions, Lemma 3I.13, using the closure properties of primitive recursive relations in Lemma 3I.9 and the bounds for subsequences of the standard, power coding in Lemma 3I.12. We illustrate the method for the relations $\mathrm{Term}_0(a)$ and $\mathrm{Formula}(e,i,a)$, the former in the simple case where the signature $\tau = (0, S, +)$ has only one constant, one unary function symbol S and one binary symbol $+$.

In our special case, the Term_0 relation satisfies the equivalence

$$\begin{aligned}\mathrm{Term}_0(a) \iff & a = \langle \#0\rangle \\ & \vee (\exists u < a)[\mathrm{Term}_0(u) \ \&\ a = \langle \#S, \#(\rangle * u * \langle \#)\rangle] \\ & \vee (\exists u, v < a)[\mathrm{Term}_0(u) \ \&\ \mathrm{Term}_0(v) \\ & \qquad \&\ a = \langle \#+, \#(\rangle * u * \langle \#,\rangle * v * \langle \#)\rangle],\end{aligned}$$

which makes it clear how it can be checked for each a if we know it on all the numbers smaller than a. From this we derive an identity for the characteristic function of the Term_0 relation, of the form

$$\chi(a) = g(a, \langle \chi(0), \dots, \chi(a-1)\rangle)$$

where

$$g(a,w) = \begin{cases} 1, & \text{if } a = \langle \#0\rangle, \\ 1, & \text{ow., if } (\exists u < a)[(w)_u = 1 \ \&\ a = \langle \#S, \#(\rangle * u * \langle \#)\rangle], \\ 1, & \text{ow., if } (\exists u, v < a)[(w)_u = 1 \ \&\ (w)_v = 1 \\ & \qquad \&\ a = \langle \#+, \#(\rangle * u * \langle \#,\rangle * v * \langle \#)\rangle], \\ 0, & \text{otherwise.}\end{cases}$$

Now $g(a,w)$ is primitive recursive, and so $\chi(a)$ is primitive recursive by Lemma 3I.13. In the case of a signature with k (rather than 3) symbols, the definition of $g(a,w)$ would have $k+1$ cases.

The Formula relation satisfies

$$\begin{aligned}\mathrm{Formula}(e,i,j) \iff & \mathrm{Formula}_0(e) \ \&\ (e)_i = \#v_i \ \& \\ & \{(\forall k < \mathrm{lh(e)})[(\mathrm{e})_\mathrm{k} \neq \#\forall \ \&\ (\mathrm{e})_\mathrm{k} \neq \#\exists] \\ & \vee (\exists e' < e)[e = \langle \#\neg, \#(\rangle * e' * \langle \#)\rangle \ \&\ \mathrm{Formula}(e', i, j \dot{-} 3)] \\ & \vee (\exists e' < e)(\exists e'' < e)(\exists b < e)[(b = \#\rightarrow \vee\, b = \#\vee \vee\, b = \#\&) \\ & \qquad \&\ e = \langle \#(\rangle * e' * \langle \#), b, \#(\rangle * e'' * \langle \#)\rangle \\ & \qquad \&\ (\mathrm{Formula}(e', i, j \dot{-} 1) \vee \mathrm{Formula}(e'', i, j \dot{-} \mathrm{lh(e')} \dot{-} 4))] \\ & \vee (\exists e' < e)(\exists i' < e)[(e = \langle \#\forall, \#v_{i'}\rangle * e' \vee e = \langle \#\exists, \#v_{i'}\rangle * e') \\ & \qquad \&\ \mathrm{Formula}(e', i, j \dot{-} 2) \wedge i' \neq i]\}.\end{aligned}$$

Using this, we can define the characteristic function of Formula by complete primitive recursion in the same way we defined that of Term_0.

(b) One way to prove this is to proceed in the style of the two proofs just given. If one does this, one needs to begin to define an extension of Sub

which has the desired value $\#\phi(t)$ when $\phi(v)$ is either an extended term or an extended formula.

Another way prove (b) is to define by primitive recursion (on j) the substitution function

$$f(e, i, a, j) = \text{sub}(e\restriction j, i, a)$$

on initial segments of e, using part (a) of the Lemma to make sure that the substitutions are made in the proper places; this implies (b) since $\text{sub}(e, i, a) = f(e, i, a, \text{lh}(e))$. ⊣

This coding of syntactic quantities (terms and formulas here, proofs later) was introduced by Gödel, and so the codes of these "metamathematical" objects are also called **Gödel numbers**. We should add that there is nothing special about the specific syntactic relations proved "primitive recursive in the codes" in Lemma 4A.3, except that we will use them in what follows; in practice all natural, "effectively decidable" syntactic relations are primitive recursive in the codes, by similar arguments—and hence they are arithmetical, by Proposition 3I.4. We exploit this fact in the next, key result.

For each formula ϕ in the language of arithmetic, we let

$$\ulcorner\phi\urcorner = \Delta\#\phi = \text{the numeral of the code of } \phi; \tag{85}$$

the closed term $\ulcorner\phi\urcorner$ is a "name" by which the language of PA can refer to ϕ. In particular, if a full extended formula $\psi(v)$ defines an arithmetical relation $P(x)$, then for each sentence θ,

$$P(\#\theta) \iff \mathbf{N} \models \psi(\Delta\#\theta) \iff \mathbf{N} \models \psi(\ulcorner\theta\urcorner).$$

Theorem 4A.4 (The Semantic Fixed Point Lemma)**.** *For each full extended formula $\psi(v)$ of* PA*, there is a sentence θ such that*

$$\mathbf{N} \models \theta \leftrightarrow \psi(\ulcorner\theta\urcorner). \tag{86}$$

PROOF. Let

$$\text{Sub}(e, m) = \text{sub}(e, 0, \#\Delta m)$$

where $\text{sub}(e, i, a)$ is the substitution function of Lemma 4A.3, so that for each extended formula $\phi(\text{v}_0)$ and every number m,

$$\text{Sub}(\#\phi(\text{v}_0), m) = \#\phi(\Delta m).$$

The function $\text{Sub}(e, m)$ is primitive recursive, and hence arithmetical; so let $\mathbf{Sub}(x, y, z)$ define its graph in $\mathbf{N}$, so that

$$\text{Sub}(e, m) = z \iff \mathbf{N} \models \mathbf{Sub}(\Delta e, \Delta m, \Delta z),$$

and set

$$\phi(\mathrm{v}_0) :\equiv (\exists z)[\mathbf{Sub}(\mathrm{v}_0, \mathrm{v}_0, z)\ \&\ \psi(z)],$$

with the given $\psi(v)$, choosing a fresh variable z so that the indicated substitutions are all free. Finally, set

$$\theta :\equiv \phi(\Delta e),\ \text{where } e = \#\phi(\mathrm{v}_0).$$

By the remarks above,

$$\mathrm{Sub}(e, e) = \#\phi(\Delta e) = \#\theta.$$

To prove (86), we compute:

$$\begin{aligned}
\mathbf{N} \models \theta &\iff \mathbf{N} \models \phi(\Delta e)\\
&\iff \mathbf{N} \models \exists z[\mathbf{Sub}(\Delta e, \Delta e, z)\ \&\ \psi(z)]\\
&\iff \text{there is some } x \text{ such that } x = \mathrm{Sub}(e, e) \text{ and } \mathbf{N} \models \psi(\Delta x)\\
&\iff \mathbf{N} \models \psi(\Delta \#\theta)\\
&\iff \mathbf{N} \models \psi(\ulcorner \theta \urcorner).
\end{aligned}$$

⊣

The Semantic Fixed Point Lemma says that every unary arithmetical relation asserts of (the code of) some sentence θ of PA exactly what θ asserts about $\mathbf{N}$. As a first illustration of its power, we prove a classical non-definability result about the **truth relation** of the structure $\mathbf{N}$,

$$\mathrm{Truth}^{\mathbf{N}}(e) \iff e = \#\theta \text{ for some } \theta \text{ such that } \mathbf{N} \models \theta. \tag{87}$$

Theorem 4A.5 (Tarski's Theorem)**.** *The truth relation for the standard model of* PA *is not arithmetical.*

Proof. If the truth relation were arithmetical, then its negation would also be arithmetical, and so there would exist a full extended formula $\psi(v)$ such that for every sentence θ of PA,

$$\mathbf{N} \not\models \theta \iff \neg\mathrm{Truth}^{\mathbf{N}}(\#\theta) \iff \mathbf{N} \models \psi(\ulcorner \theta \urcorner). \tag{88}$$

By the Semantic Fixed Point Lemma, there is a sentence θ such that

$$\mathbf{N} \models \theta \iff \mathbf{N} \models \psi(\ulcorner \theta \urcorner),$$

which is absurd, since with (88) it implies that

$$\mathbf{N} \models \theta \iff \mathbf{N} \not\models \theta.$$

⊣

To derive incompleteness results about PA by this method, we need to check that the *provability relation* of PA is arithmetical. We introduce the appropriate more general notions, which we will also need in the sequel.

Definition 4A.6 (Axiomatizations)**.** Let T be a τ-theory, i.e., (by 1G.2) any set of sentences in $\mathbb{FOL}(\tau)$.

A **set of axioms** for T is any set S of sentences of $\mathbb{FOL}(\tau)$ such that for all θ,

$$S \vdash \theta \iff T \vdash \theta; \tag{89}$$

T is **finitely axiomatizable** if it has a finite set of axioms; and T is (primitive recursively) **axiomatizable** if its signature τ is finite and T has a set of axioms S which is primitive recursive (in the codes), i.e., such that the set of codes

$$\#S = \{\#\theta \mid \theta \in S\} \tag{90}$$

is primitive recursive.

Notice that if a τ-theory is axiomatizable, then (by definition) τ is a finite signature, and the codes in (90) are computed relative to some enumeration $s_1, \dots, s_k$ of the symbols in τ. Some results about axiomatizable theories depend on the selection of a specific (primitive recursive) axiomatization, and in a few cases this is important; we will make sure to indicate these instances.

Note. In some books, by "theory" they mean a set T of sentences in a language which is closed under deducibility, i.e., such that

$$T \vdash \theta \Longrightarrow \theta \in T \quad (\theta \text{ in the vocabulary of } T).$$

We have not done this here, and so we must be careful in understanding correctly results stated when this alternative usage is in effect.

Lemma 4A.7. *Every finite theory is axiomatizable;* PA *is axiomatizable; and if T_1 and T_2 are both axiomatizable, then so is their union $T_1 \cup T_2$.* (Cf. Problem 4A.11.)

Definition 4A.8 (Proof predicates)**.** We code the proofs of a theory T as sequences of strings:

$$\begin{aligned} \text{Proof}_T(e, y) \iff & e \text{ is the code of a formula } \phi \\ & \text{and } y \text{ is the code of a proof of } \phi \text{ from } T \\ \iff & \text{there exist formulas } \phi, \phi_1, \dots, \phi_{n-1} \text{ such that} \\ & e = \#\phi \text{ and } y = \langle \#\phi_1, \dots, \#\phi_{n-1}, \#\phi \rangle \\ & \text{and } \phi_1, \dots, \phi_{n-1}, \phi \text{ is a proof of } T. \end{aligned}$$

It is simpler here to use proofs in the Hilbert-style proof system for $\mathbb{FOL}$ defined in Section 1H, but we could use proofs in the Gentzen system with only a minor complication in the codings.

Lemma 4A.9. *If T is an axiomatizable theory, then its proof predicate* $\mathrm{Proof}_T(e, y)$ (with respect to any primitive recursive axiomatization) *is primitive recursive.* (Cf. Problem 4A.12.)

PROOF is tedious but basically trivial, because the axioms and the rules of inference of first order logic can be checked"primitive recursively" in the codes. ⊣

Recall from Definition 1H.9 that a τ-theory T is **complete** if for each τ-sentence θ,

$$\text{either } T \vdash \theta \text{ or } T \vdash \neg\theta;$$

so T is **incomplete** if there is a τ-sentence θ such that

$$\text{neither } T \vdash \theta \text{ nor } T \vdash \neg\theta.$$

Theorem 4A.10 (Gödel's First Incompleteness Theorem)**.** *Every axiomatizable, sound theory T in the language of arithmetic is incomplete.*

In particular, PA *is incomplete.*

PROOF. The proof predicate $\mathrm{Proof}_T(e, y)$ constructed from a primitive recursive axiomatization of T is primitive recursive, hence arithmetical, and so it is defined by some full extended formula $\mathbf{Proof}_T(e, y)$. The Semantic Fixed Point Lemma 4A.4 applied to the formula

$$\psi(\mathrm{v}_0) \equiv (\forall y)\neg\mathbf{Proof}_T(\mathrm{v}_0, y),$$

yields a sentence γ_T such that

$$\mathbf{N} \models \gamma_T \iff \mathbf{N} \models (\forall y)\neg\mathbf{Proof}_T(\ulcorner\gamma_T\urcorner, y),$$

and we can compute, using properties of the satisfaction relation:

$$\begin{aligned}
\mathbf{N} \models \gamma_T &\iff \mathbf{N} \models (\forall y)\neg\mathbf{Proof}_T(\ulcorner\gamma_T\urcorner, y)\\
&\iff \text{for every } m, \mathbf{N} \models \neg\mathbf{Proof}_T(\ulcorner\gamma_T\urcorner, \Delta m)\\
&\iff \text{for every } m, \neg\mathrm{Proof}_T(\#\gamma_T, m)\\
&\iff T \nvdash \gamma_T.
\end{aligned}$$

It follows that $\mathbf{N} \models \gamma_T$, since otherwise (by this equivalence) $T \vdash \gamma_T$—and then γ_T is true by the soundness of T, and so it cannot be that $T \vdash \neg\gamma_T$; and since $\mathbf{N} \models \gamma_T$, by the same equivalence, $T \nvdash \gamma_T$. ⊣

Notice that the Gödel sentence γ_T depends on the specific axiomatization of T chosen for the proof; but the theorem—that T is incomplete—does not refer to any specific axiomatization of T, or to any particular method of coding the syntactic objects of T. In applying the result to a specific theory, e.g., PA,

we do not even need to refer to the *possibility of coding*: we introduce a coding and check the axiomatizability of PA *as part of the proof.*

Problems for Section 4A

Problem 4A.1 (Lemma 3I.8)**.** If h is primitive recursive, then so are f and g, where:

(1) $f(x,\vec{y}) = \sum_{i<x} h(i,\vec{y})$, $(= 0$ when $x = 0)$.

(2) $g(x,\vec{y}) = \prod_{i<x} h(i,\vec{y})$, $(= 1$ when $x = 0)$.

Problem 4A.2. If P_1, P_2, g_1, g_2 and g_3 are primitive recursive, then so is f defined from them by cases:

$$f(\vec{x}) = \begin{cases} g_1(\vec{x}), & \text{if } P_1(\vec{x}), \\ g_2(\vec{x}), & \text{if } \neg P_1(\vec{x}) \ \& \ P_2(\vec{x}), \\ g_3(\vec{x}), & \text{otherwise.} \end{cases}$$

Problem 4A.3. The functions f_0, f_1 are defined by *simultaneous primitive recursion* from w_0, w_1, h_0 and h_1 if they satisfy the identities:

$$\begin{array}{rclcrcl} f_0(0) & = & w_0, & & f_1(0) & = & w_1, \\ f_0(x+1) & = & h_0(f_0(x), f_1(x), x), & & f_1(x+1) & = & h_1(f_0(x), f_1(x), x). \end{array}$$

Prove that if h_0, h_1 are primitive recursive, then so are f_0 and f_1.

Problem 4A.4. Prove that if $g(\vec{x}, y)$ and $h(\vec{x})$ are both primitive recursive, then so is the function

$$f(\vec{x}) = (\mu y < h(\vec{x}))[g(\vec{x}, y) = 0]$$
$$\text{(with } f(\vec{x}) = h(\vec{x}) \text{ if } (\forall y < h(\vec{x}))[g(\vec{x}, y) \neq 0]).$$

Problem 4A.5. A function f is defined by *nested recursion* from g, h and τ if it satisfies the following identities:

$$f(0, y) = g(y),$$
$$f(x+1, y) = h(f(x, \tau(x, y)), x, y).$$

Prove that if f is defined from primitive recursive functions by nested recursion, then it is primitive recursive.

Problem 4A.6. Prove that there is a primitive recursive, one-to-one function $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, such that

$$g(x, y) \leq (x + y + 1)^2.$$

More generally: show that for each $n \geq 2$, there is a primitive recursive. one-to-one function $g_n : \mathbb{N}^n \to \mathbb{N}$, such that

$$g_n(x_1, \ldots, x_n) \leq P_n(x_1, \ldots, x_n), \tag{91}$$

where $P_n(x_1, \ldots, x_n)$ is a polynomial of degree n.

Problem 4A.7. Prove that for every $n \geq 2$, there is no one-to-one function $g : \mathbb{N}^n \to \mathbb{N}$ which satisfies (91) with a polynomial of degree $\leq n - 1$.

Problem 4A.8. Prove that there is a primitive recursive coding of tuples in $\mathbb{N}$ such that for every n and all $x_1, \ldots, x_n$,

$$\langle x_1, \ldots, x_n \rangle \leq 2^n P_n(x_1, \ldots, x_n),$$

where the polynomial P_n has degree n.

Problem 4A.9. Prove that for every coding $\langle\ \rangle : \mathbb{N}^* \rightarrowtail \mathbb{N}$ of tuples from $\mathbb{N}$,

$$\max\{\langle x_1, \ldots, x_n \rangle \mid x_1, \ldots, x_n \leq k\} \geq 2^n \quad (k, n \geq 2)$$

Problem 4A.10 (Lemma 4A.3)**.** Prove that there is a primitive recursive function $\mathrm{sub}(e, i, a)$, such that whenever e is the code of an extended formula $\phi(\mathrm{v}_i)$ and a is the code of a term t which is free for v_i in ϕ, then $\mathrm{sub}(e, i, a)$ is the code of $\phi(t)$, i.e., the result of replacing v_i by t in all the free occurrences of v_i in $\phi(\mathrm{v}_i)$.

Problem 4A.11 (Lemma 4A.7)**.** Outline a proof that the theory PA of Peano Arithmetic is axiomatizable.

Problem 4A.12 (Lemma 4A.9)**.** Outline a proof that the proof predicate $\mathrm{Proof}_T(e, y)$ of an axiomatizable theory T is primitive recursive.

Problem 4A.13. Prove that the theory $T = \mathsf{PA} + \neg\gamma_{\mathsf{PA}}$, obtained by adding to PA the negation of its Gödel sentence is consistent, incomplete, and not sound (for the standard model $\mathbf{N}$ of PA).

4B. Numeralwise representability in Q

Gödel's First Incompleteness Theorem 4A.10 applies only to sound theories; and while this may appear to be not a serious limitation (because who would be interested in theories which prove false number-theoretic facts), its extension to (all interesting) *axiomatizable, consistent theories* has, in fact, many applications and reveals new and deeper limitations of first order axiomatic theories. To prove these results, we need to do some proof theory.

Our aim in this section is to introduce the relevant notions and to show that the axiomatic theory Q defined in 2G.1 is strong enough to prove many fundamental properties of primitive recursive functions and relations, even though it is otherwise very weak, cf. Problems 1G.3 and 4B.1. Using these facts, we will establish the Fixed Point Theorem 4B.14, a proof-theoretic version of Theorem 4A.4 which is the main tool for the stronger results of Gödel Theory.

Definition 4B.1. Let T be a theory in the language of PA. A full extended formula $\mathbf{F}(v_1, \dots, v_n, y)$ **numeralwise represents** in T an n-ary function $f : \mathbb{N}^n \to \mathbb{N}$, if for all $x_1, \dots, x_n, w \in \mathbb{N}$,

$$f(x_1, \dots, x_n) = w \implies T \vdash \mathbf{F}(\Delta x_1, \dots, \Delta x_n, \Delta w) \\ \text{and } T \vdash (\exists! y)\mathbf{F}(\Delta x_1, \dots, \Delta x_n, y).$$

A full extended formula $\mathbf{R}(v_1, \dots, v_n)$ **numeralwise expresses** in T an n-ary relation $R \subseteq \mathbb{N}^n$, if for all $x_1, \dots, x_n \in \mathbb{N}$,

$$R(x_1, \dots, x_n) \implies T \vdash \mathbf{R}(\Delta x_1, \dots, \Delta x_n),$$
$$\neg R(x_1, \dots, x_n) \implies T \vdash \neg\mathbf{R}(\Delta x_1, \dots, \Delta x_n).$$

Lemma 4B.2. (a) *If T is sound* (for the standard model $\mathbf{N}$) of PA *and $\mathbf{R}(v_1, \dots, v_n)$ is a full extended formula which numeralwise expresses R in T, then $\mathbf{R}(v_1, \dots, v_n)$ defines R in $\mathbf{N}$, and so R is arithmetical.*

(b) *If $T_1 \subseteq T_2$ and $\mathbf{R}$ numeralwise expresses R in T_1, then $\mathbf{R}$ numeralwise expresses R in T_2, and the same is true of numeralwise representability.*

(c) *If T is inconsistent, then every relation on $\mathbb{N}$ is numeralwise expressible in T and every function $f : \mathbb{N}^n \to \mathbb{N}$ is numeralwise representable in T.*

So these notions are interesting only for consistent theories.

Notice also that if $\mathbf{F}$ numeralwise represents a function f in T, then $\mathbf{F}$ numeralwise expresses in T the graph of f,

$$G_f(x_1, \dots, x_n, w) \iff f(x_1, \dots, x_n) = w,$$

but the converse is not true: numeralwise representability is stronger than the mere numeralwise expressibility of the graph, as it demands that "T knows" for each tuple of specific numbers the existence of a unique value of $f(x_1, \dots, x_n)$. On the other hand, it may be that $\mathbf{F}(v_1, \dots, v_n, y)$ numeralwise represents a function f in T without "T knowing" that the formula defines the graph of a function, which would amount to

$$T \vdash (\forall v_1, \dots, v_n)(\exists! y)\mathbf{F}(v_1, \dots, v_n, y).$$

The notions (due to Gödel) are subtle and chosen just right so that the computations go through.

Our aim in the remainder of this section is to establish Theorem 4B.13, that all primitive recursive functions are numeralwise representable in Q.

Lemma 4B.3. *The successor function, the constant functions and the projection functions are all numeralwise representable in* Q.

Lemma 4B.4. *If* $g_1(\vec{x}), \ldots, g_m(\vec{x})$ *and* $h(u_1, \ldots, u_m)$ *are all numeralwise representable in* Q*, then so is the composition*

$$f(\vec{x}) = h(g_1(\vec{x}), \ldots, g_m(\vec{x})).$$

To prove that the class of functions which are numeralwise representable in Q is also closed under primitive recursion, we need to formalize the basic constructions of Section 1E.

Definition 4B.5. We introduce the formal abbreviations

$$\begin{aligned} x \leq y &\equiv (\exists z)[z + x = y], \\ x < y &\equiv x \leq y \ \& \ \neg(x = y), \\ (\exists u \leq y)\phi &\equiv (\exists u)[u \leq y \ \& \ \phi], \\ (\forall u \leq y)\phi &\equiv (\forall u)[u \leq y \rightarrow \phi]. \end{aligned}$$

The use of $z + x$ rather than $x + z$ in the definition of $x \leq y$ is important, because (as we have shown) Q cannot prove the commutativity of addition.

Lemma 4B.6. Q *can prove all true propositional combinations of closed equalities and inequalities between terms; i.e., if* θ *is a propositional sentence in the signature* $(0, S, +, \cdot, \leq)$*, then*

$$\mathbf{N} \models \theta \iff \mathsf{Q} \vdash \theta.$$

(Cf. Problem 4B.2.)

PROOF. Check first by induction that, for each closed term t in the language of arithmetic, if value$(t) = n$, then $\mathsf{Q} \vdash t = \Delta n$, and then prove by induction that for every quantifier free sentence θ,

$$\mathbf{N} \models \theta \iff \mathsf{Q} \vdash \theta \text{ and } \mathbf{N} \models \neg\theta \iff \mathsf{Q} \vdash \neg\theta. \qquad \dashv$$

Lemma 4B.7. $\mathsf{Q} \vdash (\forall x)[x \leq \Delta m \rightarrow x = \Delta 0 \vee x = \Delta 1 \vee \cdots \vee x = \Delta m]$. *As a consequence,*

$$\mathsf{Q}, \phi(\Delta 0), \ldots, \phi(\Delta m) \vdash (\forall u \leq \Delta m)\phi(u).$$

(Q knows all the predecessors of a numeral and can quantify over the initial segment below a numeral.)

PROOF is by induction on the number m. $\dashv$

Lemma 4B.8. *The remainder function* $\mathrm{rem}(x, y)$ *is numeralwise representable in* Q*, and hence so is the Gödel* β*-function*

$$\beta(c, d, i) = \mathrm{rem}(c, 1 + (i + 1)d).$$

Lemma 4B.9. *For every* $m \in \mathbb{N}$*,*

$$\mathsf{Q} \vdash S(z) + \Delta m = z + S(\Delta m).$$

PROOF is by induction on m. ⊣

Lemma 4B.10. *For* $m \in \mathbb{N}$*,*

$$\mathsf{Q} \vdash (\forall x)[x \leq \Delta m \vee \Delta(m + 1) \leq x].$$

PROOF is by induction on m. ⊣

Lemma 4B.11. *If the graph of* $f(x_1, \ldots, x_n)$ *is numeralwise expressible in* Q*, then there exists a full extended formula* $\mathbf{F}(v_1, \ldots, v_n, y)$ *such that the following hold for all numbers* $x_1, \ldots, x_n, w$*:*

1. $f(x_1, \ldots, x_n) = w \Longrightarrow \mathsf{Q} \vdash (\mathbf{F}(\Delta x_1, \ldots, \Delta x_n, \Delta w)$.
2. $\mathsf{Q} \vdash \forall v_1 \ldots \forall v_n(\mathbf{F}(v_1, \ldots, v_n, \Delta w)\ \&\ \mathbf{F}(v_1, \ldots, v_n, y) \rightarrow y = \Delta w)$.

In particular, $\mathbf{F}(v_1, \ldots, v_n, y)$ *numeralwise represents* f *in a very strong way.*

PROOF. Let $\mathbf{F}_1(v_1, \ldots, v_n, y)$ numeralwise express the graph of f, and take

$$\mathbf{F}(v_1, \ldots, v_n, y) \equiv \mathbf{F}_1(v_1, \ldots, v_n, y)\ \&\ (\forall u < y)\neg\mathbf{F}_1(v_1, \ldots, v_n, u),$$

where $u < y$ abbreviates $u \leq y\ \&\ u \neq y$. ⊣

Lemma 4B.12. *If* f *is defined by the primitive recursion*

$$f(0, \vec{x}) = g(\vec{x}), \quad f(t + 1, \vec{x}) = h(f(t, \vec{x}), t, \vec{x})$$

and g, h *are numeralwise representable in* Q*, then so is* f*; and the same holds for primitive recursion without parameters* (5)*.*

PROOF. We start with Dedekind's analysis of the primitive recursive definition,

$$f(t, \vec{x}) = w \iff \text{there exists a sequence } (w_0, \ldots, w_t) \text{ such that}$$
$$w_0 = g(\vec{x})\ \&\ (\forall s < t)[w_{s+1} = h(w_s, s, \vec{x})]\ \&\ w = w_t;$$

we then choose formulas $\mathbf{B}(c, d, i, y)$, $\mathbf{G}(\vec{x}, u)$ and $\mathbf{H}(u, t, \vec{x}, w)$ which numeralwise represent the β-function, g and h in the strong sense of the preceding

Lemma; and we set

$$\begin{aligned}\mathbf{F}(t,\vec{x},w) \equiv [t = 0 \ \&\ \mathbf{G}(\vec{x},w)] \\ \vee(\exists c)(\exists d)[(\exists q)[\mathbf{G}(\vec{x},w) \ \&\ \mathbf{B}(c,d,0,q)] \\ \&\ (\forall i < t)(\forall u)(\forall v)[[\mathbf{B}(c,d,i,u) \ \&\ \mathbf{H}(u,\vec{x},i,v)] \\ \rightarrow \mathbf{B}(c,d,S(i),v)] \\ \&\ \mathbf{B}(c,d,i,w)].\end{aligned}$$

⊣

Theorem 4B.13. *Every primitive recursive function is numeralwise representable in* Q*; and every primitive recursive relation is numeralwise expressible in* Q*.*

PROOF. For the second assertion, let $\mathbf{F}(\vec{v},y)$ numeralwise represent the characteristic function of R and set

$$\mathbf{R}(\vec{v}) \equiv \mathbf{F}(\vec{v},1);$$

proof that this formula numeralwise expresses R follows from the assumption that for all $x_1,\dots,x_n$,

$$\mathsf{Q} \vdash (\exists!y)\mathbf{F}(\Delta x_1,\dots,\Delta x_n,y).$$

⊣

It is important for the applications, to notice that the next, basic result applies to theories in the language of PA which need not be sound for the standard model **N**.

Theorem 4B.14 (The Fixed Point Lemma)**.** *If T is a theory in the language of arithmetic which extends Robinson's system* Q*, then for each full extended formula $\psi(v)$, we can find a sentence θ such that*

$$T \vdash \theta \leftrightarrow \psi(\ulcorner\theta\urcorner). \tag{92}$$

PROOF. As in the proof of Theorem 4A.4, let

$$\mathrm{Sub}(e,m) = \mathrm{sub}(e,0,\#\Delta m)$$

where $\mathrm{sub}(e,i,a)$ is the substitution function of the Substitution Lemma 4A.3, so that for each extended formula $\phi(\mathrm{v}_0)$,

$$\mathrm{Sub}(\#\phi(\mathrm{v}_0),m) = \#\phi(\Delta m).$$

The function $\mathrm{Sub}(e,m)$ is primitive recursive; so let $\mathbf{Sub}(x,y,z)$ numeralwise represent it in T (and such that v_0 does not occur in it), and set

$$\phi(\mathrm{v}_0) :\equiv (\exists z)[\mathbf{Sub}(\mathrm{v}_0,\mathrm{v}_0,z) \ \&\ \psi(z)],$$

choosing again a fresh variable z, so that the indicated substitutions are all free. Set

$$\theta :\equiv \phi(\Delta e), \text{ where } e = \#\phi(\mathrm{v}_0),$$

so that by the remark above,

$$\mathrm{Sub}(e, e) = \#\phi(\Delta e) = \#\theta.$$

From the definition of numeralwise representability (and the definition of $\ulcorner\theta\urcorner = \Delta\#\theta$), we have

$$T \vdash \mathbf{Sub}(\Delta e, \Delta e, \ulcorner\theta\urcorner), \tag{93}$$

$$\text{and } T \vdash (\forall z)[\mathbf{Sub}(\Delta e, \Delta e, z) \to z = \ulcorner\theta\urcorner]. \tag{94}$$

To prove (92), argue in T as follows: if $\psi(\ulcorner\theta\urcorner)$, we have

$$\mathbf{Sub}(\Delta e, \Delta e, \ulcorner\theta\urcorner) \;\&\; \psi(\ulcorner\theta\urcorner)$$

from (93), which yields

$$(\exists z)[\mathbf{Sub}(\Delta e, \Delta e, z) \;\&\; \psi(z)],$$

i.e., $\phi(\Delta e)$, i.e., θ; and if θ, we have

$$(\exists z)[\mathbf{Sub}(\Delta e, \Delta e, z) \;\&\; \psi(z)],$$

which with (94) yields $\psi(\ulcorner\theta\urcorner)$, and completes the proof. ⊣

Problems for Section 4B

Problem 4B.1. Show that Q does not prove that addition is associative, i.e.,

$$\mathsf{Q} \nvdash x + (y + z) = (x + y) + z.$$

Problem 4B.2 (Lemma 4B.6)**.** Show that Q can prove all true propositional combinations of closed equalities and inequalities between terms; i.e., if θ is a propositional sentence in the signature $(0, S, +, \cdot, \leq)$, then

$$\mathbf{N} \models \theta \iff \mathsf{Q} \vdash \theta.$$

Problem 4B.3. Prove that if a relation $R(y, \vec{x})$ is numeralwise expressible in Q, then so is the relation

$$P(z, \vec{x}) \iff (\exists y \leq z) R(y, \vec{x}).$$

4C. Rosser, more Gödel and Löb

If T_1, T_2 are theories in the same language $\mathbb{FOL}(\tau)$, we (naturally) say that

T_1 is weaker than T_2 and T_2 is stronger than T_1

$$\iff \text{ for all } \tau\text{-sentences } \theta, T_1 \vdash \theta \Longrightarrow T_2 \vdash \theta;$$

the next definition extends this idea in a natural way to theories in different languages.

4C.1. Interpretations. Let T_1, T_2 be theories, in two (possibly different) languages $\mathbb{FOL}(\tau_1)$, $\mathbb{FOL}(\tau_2)$ of finite signatures. A (propositionally faithful, minimal) **interpretation** of T_1 in T_2 is a primitive recursive function π from the *sentences* of T_1 to sentences of T_2 such that the following hold.

(1) $T_1 \vdash \theta \Longrightarrow T_2 \vdash \pi(\theta)$.
(2) $T_2 \vdash \pi(\neg\theta) \leftrightarrow \neg\pi(\theta)$.
(3) $T_2 \vdash \pi(\phi \ \& \ \psi) \leftrightarrow \pi(\phi) \ \& \ \pi(\psi)$.

Here we call $\pi : \text{Sentences}(T_1) \to \text{Sentences}(T_2)$ *primitive recursive* if there is a primitive recursive function $\pi^* : \mathbb{N} \to \mathbb{N}$ such that

$$\#\pi(\phi) = \pi^*(\#\phi) \quad (\phi \text{ any sentence of } T_1),$$

where $\#$ denotes the coding function of $\mathbb{FOL}(\tau_2)$ on the left and the coding function of $\mathbb{FOL}(\tau_1)$ on the right. Notice that (2) and (3) together imply that an interpretation preserves the propositional structure of sentences, for example

$$T_2 \vdash \pi(\phi \to \psi) \leftrightarrow (\pi(\phi) \to \pi(\psi)).$$

Directly from the definition, we get:

Lemma 4C.2. *If T_1' is weaker than T_1, T_2 is weaker than T_2', and π interprets T_1 in T_2, then π interprets T_1' in T_2'.*

When $T_2 = T_1$ or the language of T_2 is the same (or an expansion) of the language of T_1 and T_2 has more axioms, then the identity function interprets T_1 in T_2. For an example of an interpretation between entirely different languages, we note (without proof) the classical interpretation of Peano arithmetic in the axiomatic set theories specified in Definition 1G.12, which is constructed by "defining" th natural numbers within set theory:

Proposition 4C.3. *Peano arithmetic* PA *is interpretable in Zermelo's set theory without choice* Z*, and hence every subtheory of* PA *is interpretable in all the* (stronger) *axiomatic set theories listed in Definition 1G.12.*

Much stronger notions of interpretation exist and are often useful, but this is all we need now; and for the theorems we will prove, the weaker the notion of interpretation employed, the better.

Theorem 4C.4 (Rosser's form of Gödel's First Theorem)**.** *If T is a consistent, axiomatizable theory and Q is interpretable in T, then T is incomplete.*

PROOF. Fix an interpretation π of Q in T, set

$$\begin{aligned}\text{Proof}_{\pi,T}(e,y) &\iff e \text{ codes a sentence } \phi \text{ of } \mathsf{PA}\\ &\qquad \text{and } y \text{ codes a proof in } T \text{ of the translation } \pi(\phi),\\ \text{Refute}_{\pi,T}(e,y) &\iff e \text{ codes a sentence } \phi \text{ of } \mathsf{PA}\\ &\qquad \text{and } y \text{ codes a proof in } T \text{ of the translation } \pi(\neg\phi),\end{aligned}$$

and let $\mathbf{Proof}_{\pi,T}(e,y)$, $\mathbf{Refute}_{\pi,T}(e,y)$ be formulas of number theory which numeralwise express in Q these primitive recursive relations. By the Fixed Point Lemma for Q, we can construct a sentence

$$\rho = \rho(T,\pi)$$

in the language of PA, such that

$$\mathsf{Q} \vdash \rho \leftrightarrow (\forall y)[\mathbf{Proof}_{\pi,T}(\ulcorner\rho\urcorner, y) \to (\exists u \leq y)\mathbf{Refute}_{\pi,T}(\ulcorner\rho\urcorner, u)]. \tag{95}$$

The *Rosser sentence* ρ expresses the unprovability of its translation in T, but in a round-about way: it asserts that "for each one of my proofs, there is a shorter (not longer) proof of my negation".

(a) Suppose towards a contradiction that there is a proof of $\pi\rho$ in T, with code m, so by the hypotheses,

$$\mathsf{Q} \vdash \mathbf{Proof}_{\pi,T}(\ulcorner\rho\urcorner, \Delta m).$$

Taking $y = \Delta m$ and appealing to the consistency of T and basic facts about Q, we get that

$$\mathsf{Q} \vdash (\exists y)[\mathbf{Proof}_{\pi,T}(\ulcorner\rho\urcorner, y) \;\&\; (\forall u \leq y)\neg\mathbf{Refute}_{\pi,T}(\ulcorner\rho\urcorner, u)];$$

thus with (95), $\mathsf{Q} \vdash \neg\rho$, hence $T \vdash \pi(\neg\rho)$, i.e., $T \vdash \neg\pi(\rho)$, contradicting the assumed consistency of T.

(b) Suppose now that there is a proof in T of $\neg\pi(\rho)$, hence a proof of $\pi(\neg\rho)$, and let m be its code. We know that

$$\mathsf{Q} \vdash \mathbf{Refute}_{\pi,T}(\ulcorner\rho\urcorner, \Delta m),$$

among other things. To prove

$$(\forall y)[\mathbf{Proof}_{\pi,T}(\ulcorner\rho\urcorner, y) \to (\exists u \leq y)\mathbf{Refute}_{\pi T}(\ulcorner\rho\urcorner, u)] \tag{96}$$

in Q, we take cases (by Lemma 4B.10) on whether

$$y \leq \Delta m \vee \Delta(m+1) \leq y;$$

in the first of these cases we know (by Lemma 4B.7, in Q) that $y = i$ for some $i \leq m$, and it is trivial to verify that $\neg\mathbf{Proof}_{\pi,T}(\ulcorner\rho\urcorner, y)$, since this sentence is true and Q knows such true assertions about the values of $\mathrm{Proof}_{\pi,T}$ by Lemma 4B.6. In the second case, Q knows $\Delta m \leq y$, in which case the conclusion of the implication in (96) follows immediately. So we have proved (96) which is equivalent in Q to ρ by (95), contradicting (a). ⊣

Notice (again) that the Rosser sentence ρ we constructed depends on a specific axiomatization of T chosen for the proof, as well as a specific interpretation of Q into T; but the result—the incompleteness of T—is independent of these parameters, and for specific theories T, we can incorporate the verification of axiomatizability in the proof and derive a result which is entirely independent of any particular coding. This is certainly the case for the axiomatic theories of Definition 1G.12, for which the result is very clean, e.g., *if* ZFC *is consistent, then it is incomplete.*

4C.5. Remarks. Rosser's form of Gödel's Theorem 4C.4 is much more general than Gödel's First Incompleteness Theorem 4A.10, as it does not make any *soundness* assumptions of T: it applies, for example, to the theory $\mathsf{PA} + \neg\gamma_{\mathsf{PA}}$, which is consistent but certainly not sound, cf. Problem 4A.13. It also applies to axiomatic set theories, for which it is easy to establish that they interpret Q, but it is not clear exactly in what sense they are sound, and (in some cases) it is not even completely clear that they are consistent!

Next we identify a specific, especially interesting fact that sufficiently strong, consistent theories can express but cannot prove:

Definition 4C.6. For each axiomatizable theory T, let

$$\mathbf{Consis}_T \equiv \neg(\exists e)(\exists u)(\exists v)[\mathbf{Proof}_T(e,u) \;\&\; \mathbf{Refute}_T(e,v)];$$

this is the sentence of number theory which expresses formally the consistency of T—with respect, again, to a specific axiomatization of T.

Lemma 4C.7. *If T is axiomatizable and consistent, π is an interpretation of Q in T, and ρ_T is the Rosser sentence of T for π, then*

$$\mathsf{PA} \vdash \mathbf{Consis}_T \to \rho_T.$$

Proof. The proof of (a) Theorem 4C.4 is elementary, and it can be formalized in Peano Arithmetic; thus

$$\mathsf{PA} \vdash \mathbf{Consis}_T \to (\forall y)\neg\mathbf{Proof}(\ulcorner\rho_T\urcorner, y).$$

On the other hand, ρ_T is implied by its unprovability, i.e.,

$$\mathsf{PA} \vdash (\forall y)\neg\mathbf{Proof}\,(\ulcorner\rho_T\urcorner, y) \to \rho_T;$$

and these two claims, together, yield the Lemma. ⊣

Theorem 4C.8 (Gödel's Second Incompleteness Theorem)**.** *If T is an axiomatizable, consistent theory such that Q is interpretable in it by some function π, then $\pi(\mathbf{Consis}_T)$ is not a theorem of T.*

In particular, PA *cannot prove its own consistency, unless it is inconsistent.*

PROOF is immediate from the Lemma, since $T \nvdash \pi\rho_T$. ⊣

The Second Incompleteness Theorem can also be deduced using sentences gotten from the Fixed Point Lemma as was γ_{PA}. Indeed, such sentences are more closely related to Second Incompleteness than are the ρ_T.

Lemma 4C.9. *Let T and π be as in the statement of Lemma 4C.7. Let $\gamma \equiv \gamma(T, \pi)$ be the sentence gotten by applying the Fixed Point Lemma to the formula $(\forall y)\neg\mathbf{Proof}_T(v, y)$.*

(1) $Q \vdash \gamma \leftrightarrow \neg\mathbf{Proof}_T(\ulcorner\rho\urcorner)$.

(2) $\mathsf{PA} \vdash \gamma \leftrightarrow \mathbf{Consis}_T$.

PROOF. (1) is immediate from the Fixed Point Lemma.

(2) If m is the code of a proof from T of $\pi(\gamma)$, then $Q \vdash \mathbf{Proof}_T(\ulcorner\gamma\urcorner, \Delta(m))$. Thus $Q \vdash \neg\gamma$. Hence $Q \vdash \mathbf{Refute}_T(\ulcorner\gamma\urcorner, \Delta(n))$ for n the code of a proof. As in the proof of Lemma 4C.7, this proof can be formalized in PA, so $\mathsf{PA} \vdash \mathbf{Consis}_T \to \gamma$.

If T is inconsistent then there is an m such that $Q \vdash \mathbf{Proof}_T(\ulcorner\gamma\urcorner, \Delta(m))$, and so $Q \vdash\!\!\!\neq \gamma$. This argument can also be formalized in PA, so $\mathsf{PA} \vdash \neg\mathbf{Consis}_T \to \neg\gamma$. ⊣

For any sentence θ in the language of PA, clearly

$$\mathbf{N} \models (\exists y)\mathbf{Proof}_{\mathsf{PA}}(\ulcorner\theta\urcorner, y) \to \theta; \tag{97}$$

this is just a formal expression of the soundness of PA. It should be that PA can prove this basic principle—recognize that it is sound—but it cannot, except when it is trivial:

Theorem 4C.10 (Löb's Theorem)**.** *For each sentence θ of number theory,*

$$\textit{if } \mathsf{PA} \vdash (\exists y)\mathbf{Proof}_{\mathsf{PA}}(\ulcorner\theta\urcorner, y) \to \theta, \textit{ then } \mathsf{PA} \vdash \theta.$$

PROOF. Towards a contradiction, we assume that

$$\mathsf{PA} \vdash (\exists y)\mathbf{Proof}_{\mathsf{PA}}(\ulcorner\theta\urcorner, y) \to \theta \quad \text{but} \quad \mathsf{PA} \nvdash \theta$$

for some θ, so that the theory

$$T = \mathsf{PA} \cup \{\neg\theta\}$$

is consistent. We now argue (in outline) that some metamathematical arguments can be formalized in PA, to infer that $T \vdash \mathbf{Consis}_T$, contrary to the Second Incompleteness Theorem for T.

From the hypothesis,

$$\mathsf{PA} \vdash \neg\theta \to \neg(\exists y)\mathbf{Proof}_{\mathsf{PA}}(\ulcorner\theta\urcorner, y),$$

so that

$$T \vdash \neg(\exists y)\mathbf{Proof}_{\mathsf{PA}}(\ulcorner\theta\urcorner, y).$$

Next we claim that

$$\mathsf{PA} \vdash (\exists y)\mathbf{Proof}_T(\ulcorner 0 = 1\urcorner, y) \leftrightarrow (\exists y)\mathbf{Proof}_{\mathsf{PA}}(\ulcorner\neg\theta \to (0 = 1)\urcorner, y),$$

i.e., that PA recognizes (in effect) the Deduction Theorem; and also that

$$\mathsf{PA} \vdash (\exists y)\mathbf{Proof}_{\mathsf{PA}}(\ulcorner\neg\theta \to (0 = 1)\urcorner, y) \leftrightarrow (\exists y)\mathbf{Proof}_{\mathsf{PA}}(\ulcorner\theta\urcorner, y),$$

i.e., that PA recognizes that it can do proofs by contradiction. Replacing PA by the stronger T and combining these two equivalences, we get

$$T \vdash \neg(\exists y)\mathbf{Proof}_T(\ulcorner 0 = 1\urcorner, y) \leftrightarrow \neg(\exists y)\mathbf{Proof}_{\mathsf{PA}}(\ulcorner\theta\urcorner, y);$$

and since T proves the right-hand-side of this equivalence, it also proves the left-hand-side, which implies (in PA) $\mathbf{Consis}_T$. ⊣

It should be clear that the appropriate version of Löb's Theorem holds for any consistent, axiomatizable theory T in which PA can be interpreted, cf. Problem 4C.3.

4C.11. Provability theory. It is convenient to introduce a notation for the sentences which express formally provability, and so for a fixed axiomatizable T in the language of PA and any sentence θ, we set:

$$\Box_T(\theta) :\equiv (\exists y)\mathbf{Proof}_T(\ulcorner\theta\urcorner, y). \tag{98}$$

This sentence depends, of course, on the specific axiomatization of T we choose to define the proof predicate.

With this notation, Löb's Theorem takes the simple form

$$\text{if } \mathsf{PA} \vdash \Box_{\mathsf{PA}}(\theta) \to \theta, \text{ then } \mathsf{PA} \vdash \theta,$$

and the question arises whether its formal version can be proved in PA, i.e., whether the following holds for each sentence θ:

$$\mathsf{PA} \vdash \Box_{\mathsf{PA}}(\Box_{\mathsf{PA}}(\theta) \to \theta) \to \Box_{\mathsf{PA}}(\theta).$$

This is indeed true, and has interesting consequences. To show it, we must look a little more carefully at how various informal (mathematical) claims can be *formalized and proved* in axiomatized theories, a topic which is generally referred to as *provability theory*. We will confine ourselves here to just a few, basic facts.

4C.12. Bounded and Σ_1 formulas. A formula ϕ in the language of PA is **bounded** if it contains only bounded quantifiers as in Definition 4B.5, i.e., more precisely, if it belongs to the smallest set of formulas which contains all the prime formulas of the form $s = t$ and is closed under the propositional connectives and bounded quantification, i.e., the formation rules

$$\psi \mapsto (\exists \mathrm{v}_i \leq \mathrm{v}_j)\psi,\ \psi \mapsto (\forall \mathrm{v}_i \leq \mathrm{v}_j)\psi.$$

A formula ϕ is Σ_1 if

$$\phi \equiv \exists x_1 \cdots \exists x_n \psi$$

where ψ is a bounded formula.

For any theory T in the language of PA, a formula ϕ is T-**bounded** or T-Σ_1 if there is a bounded or Σ_1 formula ϕ^* such that

$$T \vdash \phi \leftrightarrow \phi^*;$$

and a formula ϕ is T-Δ_1, if both ϕ and $\neg\phi$ are T-Σ_1.

Proposition 4C.13. *Suppose T is an extension of* PA*, in the language of* PA*.*

(1) *The class of T-Σ_1 formulas includes all prime formulas and is closed under the positive propositional connectives & and $\vee$, bounded quantification of both kinds, and unbounded existential quantification.*

(2) *For each primitive recursive function $f(\vec{x})$, there is a T-Δ_1 formula $\phi(\vec{v}, w)$ which numeralwise represents $f(\vec{x})$ in T.*

(3) *Each primitive recursive relation is numeralwise expressible in T by a T-Δ_1 formula.*

PROOF of these propositions can be extracted by reading with some care the proofs in Section 4B, and formalizing in the given T some easy, informal arguments. For example, to show for the proof of (1) that the class of T-Σ_1 formulas is closed under universal bounded quantification, it is enough to show that for any extended formula $\phi(x, y, z)$,

$$T \vdash (\forall x \leq y)(\exists z)\phi(x, y, z) \leftrightarrow (\exists w)(\forall x \leq y)(\exists z \leq w)\phi(x, y, z);$$

the equivalence expresses an obvious fact about numbers, which can be easily proved by induction on y—and this induction can certainly be formalized in PA.

(2) and (3) can be read-off the proof of the basic Theorem 4B.13, by computing the forms of all the formulas used in that proof and appealing to (1) of this Proposition. ⊣

Proposition 4C.14. *Suppose T is an axiomatizable extension of* PA*, in the language of* PA*.*

(1) *The proof predicate* $\text{Proof}_T(e,y)$ *of T is numeralwise expressible by a T-Δ_1 formula* $\mathbf{Proof}_T(e,y)$*; and hence, for each sentence θ, the provability assertion $\Box_T(\theta)$ is a T-Σ_1 sentence.*

(2) *For every T-Σ_1 sentence ϕ,*

$$T \vdash \phi \to \Box_T(\phi).$$

(3) *For every sentence θ,*

$$T \vdash \Box_T(\theta) \to \Box_T(\Box_T(\theta)).$$

Proof. (1) The formula $\mathbf{Proof}_T(e,y)$ is T-Δ_1 by (3) of the preceding theorem, and so $\Box_T(\theta)$ is T-Σ_1 by its definition (98).

(2) Notice that this is not a trivial claim, because it does not, in general, hold for sentences which are not T-Σ_1: if, for example, T is sound and γ_T is its Gödel sentence, then $\gamma_T \to \Box_T(\gamma_T)$ is not true, and so it is not a theorem of T. To prove the claim, let

$$\begin{aligned}\text{Proof}^n_T(e,x_1,\dots,x_n,y) \iff\ & e \text{ is the code of}\\ & \text{a full extended formula } \phi(v_1,\dots,v_n)\\ & \text{and } y \text{ is the code of a proof of } \phi(\Delta x_1,\dots,\Delta x_n) \text{ from } T.\end{aligned}$$

This is a generalization of the proof relation $\text{Proof}_T(e,y)$, so that, in fact

$$\text{Proof}_T(e,y) \iff \text{Proof}^0_T(e,y),$$

and it is also primitive recursive. Let $\mathbf{Proof}^n_T(x_0,x_1,\dots,x_n,y)$ be a formula which numeralwise expresses $\text{Proof}^n_T(e,x_1,\dots,x_n,y)$ in T. The heart of the proof is to show that *for every full extended bounded formula $\psi(v_1,\dots,v_n)$,*

$$T \vdash \psi(x_1,\dots,x_n) \to (\exists y)\mathbf{Proof}^n_T(\ulcorner\psi(x_1,\dots,x_n)\urcorner, x_1,\dots,x_n,y). \tag{99}$$

This is verified by induction on the construction of bounded formulas, i.e., it is shown first for prime formulas, and then it is shown that it persists under the positive propositional connectives and bounded quantification. Notice,

again, that a detailed (complete) proof would involve a good deal of work: for example, to show (part of) the basic case

$$T \vdash u + v = w \to (\exists y)\mathbf{Proof}^3_T(\ulcorner u + v = w \urcorner, u, v, w, y), \tag{100}$$

we must formalize in T the informal claim

$$\text{if } u + v = w, \text{ then } T \vdash \Delta u + \Delta v = \Delta w;$$

the proof of the informal claim is by induction on v—and so the formal proof of (100) requires induction within T. This is why we assume in the theorem that T extends PA—there is no way to show this result for weak theories like Q. On the other hand, PA is a powerful theory in which we can formalize inductive proofs, and so (99) is plausible, and can be verified with some computation.

To complete the proof of (2), suppose

$$\phi \equiv (\exists x_1) \cdots (\exists x_n)\psi(x_1, \dots, x_n)$$

is a Σ_1 sentence with $\psi(x_1, \dots, x_n)$ bounded and having no free variables other than the indicated $x_1, \dots, x_n$, and argue in T. Assume $\psi(x_1, \dots, x_n)$, and infer

$$(\exists y)\mathbf{Proof}^n_T(\ulcorner \psi(x_1, \dots, x_n) \urcorner, x_1, \dots, x_n, y)$$

by (99). From this, by trivial properties of the provability relations (which can be formally established in PA), infer that

$$(\exists y)\mathbf{Proof}\,(\ulcorner (\exists x_1) \cdots (\exists x_n)\psi(x_1, \dots, x_n) \urcorner, y).$$

So we have shown that

$$T \vdash \psi(x_1, \dots, x_n) \to (\exists y)\mathbf{Proof}\,(\ulcorner (\exists x_1) \cdots (\exists x_n)\psi(x_1, \dots, x_n) \urcorner, y),$$

from which we get immediately the required

$$T \vdash (\exists x_1) \cdots (\exists x_n)\psi(x_1, \dots, x_n) \to (\exists y)\mathbf{Proof}\,(\ulcorner (\exists x_1) \cdots (\exists x_n)\psi(x_1, \dots, x_n) \urcorner, y).$$

Finally, (3) follows from (1) and (2). ⊣

By methods like these, we can show that the statement and proof of Löb's Theorem for any axiomatizable extension of PA, can also be formalized in PA:

Theorem 4C.15. *For any axiomatizable extension T of PA, and every sentence θ,*

$$\mathsf{PA} \vdash \Box_T(\Box_T(\theta) \to \theta) \to \Box_T(\theta).$$

The most complex part of the argument is the formalization in PA of the proof of the Second Incompleteness Theorem of Gödel 4C.8.

And, finally, to prove the following considerably deeper result, we also need to formalize in PA the Gentzen Hauptsatz:

Theorem 4C.16. PA *is not finitely axiomatizable.*

This is somewhat different from the preceding is that its *statement* (as opposed to its proof) does not depend on any particular coding of formulas, proofs, etc.: the result simple asserts that no finite set of sentences in the language of PA has exactly the same consequences as PA.

Problems for Section 4C

Problem 4C.1. Suppose T is an axiomatizable theory, π is an interpretation of Q into T, and ρ is the Rosser sentence for T (relative to some axiomatization and π): is ρ true or false?

Problem 4C.2. Prove that the theory ZFC (Zermelo-Fraenkel set theory with choice) defined in Definition 1G.12 is incomplete, unless it is inconsistent. (This requires knowing some set theory.)

Problem 4C.3 (Abstract Löb Theorem)**.** Suppose T is a consistent, axiomatizable theory into which PA can be interpreted. Prove that for any sentence θ in the language of T,

$$\text{if } T \vdash \pi\Big(\exists y \mathbf{Proof}_{\pi,T}(\ulcorner\theta\urcorner, y)\Big) \to \theta, \text{ then } T \vdash \theta,$$

where $\mathbf{Proof}_{\pi,T}(e, y)$ is defined in the proof of Theorem 4C.4.

Problem 4C.4. (A corrected version of #2 in the Fall 1998 Logic Qual.) For each of the following assertions, determine whether the assertion is true for every formula θ and prove your answers by reference to appropriate theorems where necessary.

(a) $\mathsf{PA} \vdash \mathbf{Provable}_{\mathsf{PA}+\neg\theta}(\Delta\#\theta) \to \mathbf{Provable}_{\mathsf{PA}}(\Delta\#\theta)$.
(b) $\mathsf{PA} \vdash \mathbf{Provable}_{\mathsf{PA}}(\Delta\#\theta) \to \neg\mathbf{Provable}_{\mathsf{PA}}(\Delta\#\neg\theta)$.

Problem 4C.5. (#3 in the Fall 2002 Qual.) Let $\mathbf{Provable}_{\mathsf{PA}}(e) \equiv \exists y \mathbf{Proof}_{\mathsf{PA}}(e, y)$. Consider the following four sentences which can be constructed from an arbitrary sentence θ:

(a) $\theta \to \mathbf{Provable}_{\mathsf{PA}}(\ulcorner\theta\urcorner)$
(b) $\mathbf{Provable}_{\mathsf{PA}}(\ulcorner\theta\urcorner) \to \theta$
(c) $\mathbf{Provable}_{\mathsf{PA}}(\ulcorner\theta\urcorner) \to \mathbf{Provable}_{\mathsf{PA}}(\ulcorner\mathbf{Provable}_{\mathsf{PA}}(\ulcorner\theta\urcorner)\urcorner)$

(d) $\mathbf{Provable}_{\mathsf{PA}}(\ulcorner\mathbf{Provable}_{\mathsf{PA}}(\ulcorner\theta\urcorner)\urcorner) \to \mathbf{Provable}_{\mathsf{PA}}(\ulcorner\theta\urcorner)$

Determine which of these four sentences are provable in PA (for every choice of θ), and justify your answers by appealing, if necessary, to standard theorems which are proved in 220.

Problem 4C.6. (#8 in the Fall 2003 Qual.) Let σ be gotten from the Fixed Point Lemma applied to $\forall v_2 \neg \mathbf{Proof}_{\mathsf{PA}}(v_1, v_2)$. In other words, let σ be a sentence such that

$$\mathsf{PA} \vdash (\sigma \leftrightarrow \forall v_2 \neg \mathbf{Prov}(\Delta k, v_2),$$

where $k = \#\sigma$. Let T be the theory gotten from PA by adding $\neg\sigma$ as an axiom. Show that T is ω-inconsistent: that is, there is a formula $\psi(v_1)$ such that $T \vdash \exists v_1 \psi(v_1)$ and $T \vdash \neg\psi(\Delta n)$ for each numeral Δn.

Problem 4C.7. True or false: if T is an inconsistent theory, then every theory is interpretable in T.

Problem 4C.8. (#3 in the Fall 2004 Qual.) A *sound interpretation* of Peano arithmetic into a theory T (in any language with finite signature) is a primitive recursive function $\theta \mapsto \theta^*$ on the sentences of PA to the sentences of T which satisfies the following properties, for every sentence θ in the language of PA:

(1) If $\mathsf{PA} \vdash \theta$, then $T \vdash \theta^*$.
(2) If $T \vdash \theta^*$, then θ is true.
(3) $\Big(\neg\theta\Big)^* \equiv \neg\theta^*$.

Prove that if T is axiomatizable and there exists a sound interpretation of PA into T, then T is incomplete.

Hint. Use the Fixed Point Lemma in Peano Arithmetic.

Problem 4C.9. (#8 in the Winter 2005 Qual.) A sentence in the language of PA is Π_1 if it is of the form

$$\phi \equiv (\forall x_1) \cdots (\forall x_n)\theta$$

where θ has only bounded quantifiers of the form

$$(\exists x \le y), \qquad (\forall x \le y).$$

Let PA be Peano arithmetic and prove that for every Π_1 sentence ϕ,

$$\mathsf{PA}, \mathrm{Con}_P(\ulcorner\phi\urcorner) \vdash \phi,$$

where $\mathrm{Con}_P(\ulcorner\phi\urcorner)$ expresses in a natural way the consistency of ϕ with Peano arithmetic, in other words it is the sentence $\neg(\exists y)\mathbf{Proof}\,(\ulcorner\neg\phi\urcorner, y)$.

4D. Computability and undecidability

Is it possible to determine "effectively" whether an arbitrary sentence of arithmetic is true? Gödel's First Incompleteness Theorem shows that this cannot be done by the classical axiomatic method, i.e., by singling out some "obvious" arithmetical truths and then (formally) proving all the others, but it may be argued that this exhibits only a fundamental *incompleteness of the axiomatic method*: it may be that some other method (unrelated to logic) might "identify" effectively all arithmetical truths, without necessarily justifying them (by reducing them to some few, obvious axioms). We will show in this and the next two sections that this cannot be done, by proving that *there is no general method which can decide effectively whether a given sentence of the language of arithmetic is true.* The methods we will use (due to Turing, Church and Kleene) will yield a host of related *undecidability results* which are among the most fundamental applications of logic.

For each set Λ of "symbols", Λ^* is the set of *strings* (words, finite sequences) from Λ, including the empty string ϵ. For example, the sets of terms and formulas of $\mathbb{FOL}(\tau)$ for a specific finite signature τ are sets of strings from the alphabet

$$\neg \ \rightarrow \ \ \& \ \ \vee \ \forall \ \exists \ = \ (\ )\ ,\ s_1 \ \ldots \ s_k \ \mathrm{v}_0 \ \mathrm{v}_1 \ \ldots$$

of $\mathbb{FOL}(\tau)$. We can replace this by the finite alphabet

$$V_\tau = \{\neg, \rightarrow, \ \& \ , \vee, \forall, \exists, =, (,), \ \ , , s_1, \ldots s_k, \mathrm{v}, |\}$$

where v (for "variable") and the tally | are new symbols and we identify the variable v_i with the string of v followed by $i+1$ tallies,

$$\mathrm{v}_0 \equiv \mathrm{v}|, \mathrm{v}_1 \equiv \mathrm{v}||, \mathrm{v}_2 \equiv \mathrm{v}|||, \ldots$$

If we further think of *proofs* in $\mathbb{FOL}(\tau)$ as sequences of formulas separated by commas, then proofs are also words in this finite alphabet V_τ, i.e., members of ${V_\tau}^*$. Thus the notion that we need to make precise is that of a *computable function*

$$f : \Lambda^* \rightarrow \Lambda^*$$

for an arbitrary finite Λ; a set of words $A \subseteq \Lambda^*$ will be *decidable* if its characteristic function

$$\chi_A(\alpha) = \begin{cases} T & \text{if } \alpha \in A \\ F & \text{otherwise,} \end{cases}$$

where T and F are any two, specific, distinct strings standing for truth and falsity.

Alan Turing had (in 1936) the fundamental intuition that a string function is computable if its values can be computed by some "mechanical device" (machine) which has access to the string argument of the function and an unbounded amount of "scratch paper" for each computation. Turing's abstract, mathematical model of "machine" was introduced before actual computers had been built, but it has proved very robust and (in all interesting aspects other than efficiency, which does not concern us here) equivalent to the electronic computers we use today.

4D.1. Turing machines. A Turing machine is a structure

$$M = (S, Q_0, \Sigma, \sqcup, \text{Table}),$$

where the following hold.

(1) S is a finite set, the set of (internal) *states* of M, and $Q_0 \in S$ is a specified *initial state*.

(2) Σ is a finite set, the set of *symbols* (alphabet) of M, and $\sqcup \in \Sigma$ is a specified member of Σ standing for "the blank symbol" (empty space).

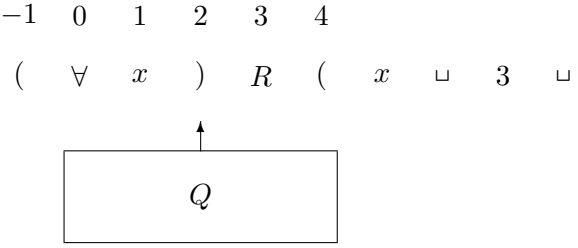
(3) The Table of M is a finite set of *transitions*, i.,e., quintuples of the form

$$Q, X \mapsto X', Q', m \tag{101}$$

where Q and Q' are states; X and X' are symbols; and the *move* of the transition $m \in \{0, -1, +1\}$. We say that the pair (Q, X) *activates* the transition.

A machine M is *deterministic* if for each state Q and each symbol X there is at most one transition which is activated by the pair (Q, X), otherwise it is *non-deterministic*.

Turing's image is that the machine is situated in front of a two-way infinite *tape* which has a finite number of symbols from the alphabet placed on it; the machine can only see the symbol on the *cell* just in front of it—it cannot see any other symbols and it cannot see the coordinate of that cell, i.e., it does not "know" where it is on the tape.

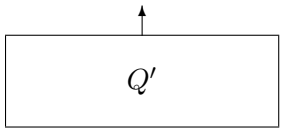


If the machine is in state Q and the *visible symbol* is X, then each transition (101) in the machine's Table which is activated by the pair (Q, X) produces a change in this *situation*, overwriting the symbol X by the *new symbol* X', changing from the *current state* Q to the *new state* Q' and *moving* one-cell-to-the-left if the move $m = -1$, not-at-all if $m = 0$, and one-cell-to-the-right if $m = 1$. For example, the transition

$$Q,) \mapsto (, Q', +1$$

will change the situation in the picture above to the new situation:

-1	0	1	2	3	4				
$($	$\forall$	x	$($	R	$($	x	$\sqcup$	3	$\sqcup$



Finally, a *computation* of M is a sequence of successive situations produced by transitions of M in this way, starting with an *initial situation* involving the initial state Q_0.

Without further explanation of this simple idea, we proceed to the precise definitions of the notions italicized in these remarks.

Definition 4D.2. For a fixed Turing machine $M = (S, Q_0, \Sigma, \sqcup, \text{Table})$, we define:

(1) A *tape* (description) is any function $\tau : \mathbb{Z} \to \Sigma$, which assigns a symbol of M to each rational integer

$$i \in \mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\},$$

such that for all but finitely many i, $\tau(i) = \sqcup$.

(2) A *situation* of M is any triple

$$s = (Q, \tau, i),$$

where Q is a state, τ is a tape and $i \in \mathbb{Z}$. The *state* of M in this situation is Q; the *place* of M in s is the integer i; and the *visible symbol* in s is $\tau(i)$. We call s *initial* if Q is the initial state Q_0 of M and $i = 0$; and we call s *terminal* if either there is no transition in the table of M activated by the pair $(Q, \tau(i))$ or the only transition activated by this pair is a "stand-pat" transition

$$Q, X \mapsto X, Q, 0.$$

(3) A situation $s' = (Q', \tau', i')$ is a *next situation* to $s = (Q, \tau, i)$ if s is not terminal, if

$$j \neq i \Longrightarrow \tau(j) = \tau'(j),$$

and if the Table of M contains the transition

$$Q, \tau(i) \mapsto \tau'(i), Q', i' - i.$$

Notice that this implies $|i' - i| \leq 1$, since $i' - i$ is a move; that (by the definition) there is no s' next to s if s is terminal; and that if M is deterministic, then there is at most one s' next to s, since at most one transition can be activated by the given pair $(Q, \tau(i))$.

(4) A *computation* of M is any (finite or infinite) sequence of situations

$$s_0, s_1, \dots,$$

such that s_0 is initial and each s_{i+1} is next to s_i, diagrammatically

$$s_0 \mapsto s_1 \mapsto s_2 \mapsto \cdots$$

A computation is *maximal* if no extension of it is a computation, and a maximal, finite computation is called *convergent*. For each initial situation s_0, we set

(102) $M : s_0\!\downarrow$

$$\iff \text{there exists a convergent computation } s_0 \mapsto \cdots \mapsto s_m$$

and we read "$M : s_0\!\downarrow$" as "M *halts*" (or *converges*) *on* s_0.

It follows easily that if M is deterministic, then for each initial situation $s_0 = (Q_0, \tau, 0)$ there is exactly one maximal computation which starts with s_0, and that a maximal computation is either finite, ending with a terminal situation, or infinite (and with no terminal situations in it). We picture these possibilities in the following, simple examples of Turing machines.

4D.3. Example. The machine with just one state Q_0 on the alphabet $\{1, \sqcup\}$ and just two transitions

$$Q_0, 1 \mapsto 1, Q_0, +1$$
$$Q_0, \sqcup \mapsto 1, Q_0, +1$$

is deterministic, and starting from any initial situation, it moves to the right forever, printing a 1 on every cell to the right of the origin which does not already have a 1 in it.

4D.4. Example. On the same alphabet $\{1, \sqcup\}$, consider the machine with the following transitions (and the states which occur in these transitions):

(a) $$Q_0, 1 \mapsto 1, Q_0, +1$$

(b) $$Q_0, \sqcup \mapsto 1, Q_1, 0$$

(c) $$Q_1, 1 \mapsto 1, Q_1, -1$$

(d) $$Q_1, \sqcup \mapsto \sqcup, Q_2, +1$$

For each number x, let

$$\mathrm{in}(x) = \cdots \sqcup\sqcup \underbrace{11\ldots1}_{x+1}, \sqcup\sqcup \ldots$$

be the tape with $x+1$ 1s on and to the right of the origin and no other symbols but blanks, and consider the computation of this deterministic machine starting with the initial situation $(Q_0, \mathrm{in}(x), 0)$. It will start with $x+1$ executions of the transition (a), as long at is sees a 1, and then execute (b) just once, to write a 1 on the first blank cell on the right; it will then execute (c) $x+3$ times, until it is back to the left of the origin, where it finds the first blank on the left, and finally execute (d) just once to move to the origin and stop, in the situation $(Q_2, \mathrm{in}(x+1), 0)$.

For each string $\alpha \equiv \alpha_0\alpha_1\cdots\alpha_{n-1} \in \Lambda^*$, let

$$\mathrm{in}(\alpha)(i) = \begin{cases} \alpha_i, & \text{if } 0 \le i < n, \\ \sqcup, & \text{otherwise;} \end{cases}$$

this is the tape that we use to represent a string α an an *input* to a computation by a Turing machine whose alphabet includes Λ. Similarly, for each tape τ, let

$$\begin{aligned} \mathrm{out}(\tau) =& \tau(0)\tau(1)\cdots\tau(m-1) \\ & \text{for the least } m \in \mathbb{N} \text{ such that } \tau(m) = \sqcup, \end{aligned}$$

so that if $\tau(0) = \sqcup$, then $\mathrm{out}(\tau) = \epsilon$ (the empty string), and if $\tau(0) = N$, $\tau(1) = O$ and $\tau(2) = \sqcup$, then $\mathrm{out}(\tau) = NO$.

Definition 4D.5 (Turing computable functions)**.** A Turing machine

$$M = (S, Q_0, \Sigma, \sqcup, \mathrm{Table})$$

computes a function

$$f : \Lambda^* \to \Lambda^*$$

if $\Lambda \subseteq \Sigma$; $\sqcup \notin \Lambda$; and for all strings α, $\beta \in \Lambda^*$,

$$f(\alpha) = \beta \iff \text{there exists a convergent computation } s_0, s_1, \dots s_m \text{ of } M \text{ such that } s_0 = (Q_0, \text{in}(\alpha), 0) \text{ and } s_m = (Q, \tau', i), \text{ with out}(\tau') = \beta.$$

Similarly, and representing each tuple $x_1, \dots, x_n$ of numbers by the string of 1s and blanks

$$\text{in}(x_1, \dots, x_n) = \dots \sqcup\sqcup \underbrace{11\dots1}_{x_1+1} \sqcup \underbrace{11\dots1}_{x_2+1} \dots \sqcup \underbrace{11\dots1}_{x_n+1} \sqcup\sqcup \dots$$

(with $\text{in}(x_1, \dots, x_n)(0) =$ the first 1), a Turing machine M as above computes a function

$$f : \mathbb{N}^n \to \mathbb{N}$$

if the alphabet of M includes the symbol 1 and for all $x_1, \dots, x_n$, w,

$$f(x_1, \dots, x_n) = w \iff \text{there exists a convergent computation } s_0, s_1, \dots s_m \text{ of } M \text{such that } s_0 = (Q_0, \text{in}(x_1, \dots, x_n), 0) \text{ and } s_m = (Q, \tau', i), \text{ with out}(\tau') = \underbrace{11\dots1}_{w+1}.$$

Note that in both situations, if the machine M is deterministic, then for each input, there will be exactly one convergent computation ("the computation") of M which computes the value of the function.

A string or number-theoretic function is **Turing computable** if it is computed by a deterministic Turing machine.

After giving these definitions, Turing claimed that his simple, restricted machines can actually compute all functions on strings which are "intuitively computable", so that his precise definition can be used to prove rigorously that specific functions *are not computable in any way whatsoever*, by showing that they cannot be computed by a Turing machine. Alonzo Church had made a similar proposal for another, precisely defined class of functions (subsequently proved to coincide with the class of Turing computable functions), so that the next, fundamental claim carries now both their names:

4D.6. The Church-Turing Thesis. *A string function $f : \Lambda^* \to \Lambda^*$* (on a finite alphabet Λ) *is computable exactly when it is Turing computable*; *and a set of strings $A \subseteq \Lambda^*$ is decidable exactly when its characteristic function is decidable, taking* (for concreteness) $T = \text{in}(1)$ *and* $F = \text{in}(0)$.

Since the operations

$$x_1, \dots, x_n \mapsto \mathrm{in}(x_1, \dots, x_n) \quad \text{and} \quad \mathrm{in}(w) \mapsto w$$

which code and decode numbers by strings of 1s are evidently computable (in a very basic, intuitive sense), the Church-Turing Thesis implies its version for functions on the natural numbers:

4D.7. The Church-Turing Thesis for functions on $\mathbb{N}$. *A number-theoretic function $f : \mathbb{N}^n \to \mathbb{N}$ is computable exactly when it is Turing computable; and a relation $R \subseteq \mathbb{N}^n$ is decidable exactly when its characteristic function is Turing computable.*

4D.8. Remarks. The Church-Turing Thesis is not a theorem and cannot be rigorously proved, as it identifies the prematheматical, intuitive notion of "computability" with a precisely defined (set-theoretic) notion of "computability by a Turing machine". At the same time, the Thesis is not a "definition by stipulation", in the sense that when we adopt it we simply *decide* (arbitrarily and for convenience) to call a function "computable" exactly when it is Turing computable—it would not be useful if that were all it is. Its status is similar to the "definitions" of *area* and *volume* in Geometry or *work* in Physics, which within a rigorous development of mathematics are treated as arbitrary, stipulative definitions, but whose significance for applications derives from the fact that they are not-at-all arbitrary: when we prove that the volume of a ball of radius r is $(4/3)\pi r^3$ using the "definition" of volume via an integral, we make a claim that the physical approximations to ideal balls we meet in our world will exhibit this relationship between their radius and their volume—and experimentation verifies this. In the same way, when we prove that a certain function $f : \mathbb{N} \to \mathbb{N}$ is not Turing computable, we claim (through the Church-Turing Thesis) that nobody, ever will devise an "algorithm" which (effectively and uniformly) will compute each value $f(m)$ from the argument m, and this claim is subject to experimentation and verification.

The main arguments supporting the truth of the Church-Turing Thesis are

(1) Turing's original analysis of the notion of "machine computability", strengthened immensely by our current, much better understanding of *symbolic computation* gained from our experience with actual computers;
(2) the great wealth of Turing computable functions, and the very strong closure properties of the class of Turing computable functions; and
(3) the experience of more than seventy years, which has failed to produce plausible counterexamples.

We will not elaborate on any of these here, except that the main evidence for (2) will be detailed in the subsequent sections on *computability theory*.

The main applications of the Church-Turing Thesis are *negative*, in proofs which establish that certain functions are not computable by proving (rigorously) that they are not Turing computable and appealing to the Thesis. It is customary to claim sometimes that "f is Turing computable, since we have given intuitive instructions for computing it", but what is always meant by this is "f is Turing computable, but I do not want to take the time to prove this in detail because it is boring and routine (to someone who has understood the justification of the Church-Turing thesis)".

4E. Computable partial functions

Not every deterministic Turing machine computes a string function, because the computation from any given string α may fail to terminate, as in Example 4D.3, where the computation *on every input* is infinite and fails to return a value; however, every Turing machine computes a "partial string function", where these objects are defined as follows:

Definition 4E.1. A **partial function**

$$f : X \rightharpoonup Y$$

on a set X to some set Y is any (ordinary, total) function

$$f : X_0 \to Y,$$

where X_0 is any subset of X. We call X_0 the *domain of convergence* of f, and set

$$\begin{aligned} f(x)\!\downarrow &\iff x \in X_0 \qquad (f(x) \textit{ converges} \text{ or } \textit{is defined}) \\ f(x)\!\uparrow &\iff x \in X \setminus X_0 (f(x) \textit{ diverges}). \end{aligned}$$

Notice the special notation $\rightharpoonup$ which indicates that f is a partial function. Notice also that, by the definition, every total $f : X \to Y$ is a partial function (taking $X_0 = X$), and (at the other extreme), taking $X_0 = \emptyset$, we have the *totally undefined* partial function $f : X \rightharpoonup Y$ for which $f(x)\!\uparrow$, for every $x \in X$.

Turing computability for string and number-theoretic partial functions is defined (almost) exactly like the corresponding notion for total functions in 4D.5, except that we insist that the machine computation "converges" (is finite) exactly when the partial function converges. We repeat the definition to make precise this additional condition.

4E.2. Turing computable partial functions. A Turing machine

$$M = (S, Q_0, \Sigma, \sqcup, \text{Table})$$

computes a partial function

$$f : \Lambda^* \rightharpoonup \Lambda^*$$

if $\Lambda \subseteq \Sigma$; $\sqcup \notin \Lambda$; and for all strings α, $\beta \in \Lambda^*$,

$$\begin{aligned} f(\alpha)\downarrow &\iff M : (Q_0, \text{in}(\alpha), 0)\downarrow, \\ f(\alpha) = \beta &\iff \text{there exists a convergent computation } s_0, s_1, \dots s_m \text{ of } M \\ &\qquad \text{such that } s_0 = (Q_0, \text{in}(\alpha), 0) \text{ and } s_m = (Q, \tau', i), \\ &\qquad \text{with } \text{out}(\tau') = \beta; \end{aligned}$$

similarly, a Turing machine M as above computes a partial function $f : \mathbb{N}^n \rightharpoonup \mathbb{N}$ if the alphabet of M includes the symbol 1, and for all $x_1, \dots, x_n$, w,

$$\begin{aligned} f(x_1, \dots, x_n)\downarrow &\iff M : (Q_0, \text{in}(x_1, \dots, x_n), 0)\downarrow, \\ f(x_1, \dots, x_n) = w &\iff \text{there exists a convergent computation } s_0, s_1, \dots s_m \\ &\qquad \text{of } M \text{ such that } s_0 = (Q_0, \text{in}(x_1, \dots, x_n), 0) \\ &\qquad \text{and } s_m = (Q, \tau', i), \text{ with } \text{out}(\tau') = \underbrace{11\dots1}_{w+1}. \end{aligned}$$

A string or number-theoretic partial function is **Turing computable** if it is computed by a deterministic Turing machine.

4E.3. The Church-Turing Thesis for partial functions. *A string partial function $f : \Lambda^* \rightharpoonup \Lambda^*$ is computable exactly when it is Turing computable*; and *a number-theoretic partial function $f : \mathbb{N}^n \rightharpoonup \mathbb{N}$ is computable exactly when it is Turing computable.*

Number theoretic partial functions arise very naturally through the application of the following (unbounded) *minimalization* operator, which, on the surface, is unrelated to Turing computability.

Definition 4E.4 (Unbounded minimalization)**.** With each partial function

$$g : \mathbb{N}^{n+1} \rightharpoonup \mathbb{N},$$

we associate a new, n-ary partial function

$$\begin{aligned} f(\vec{x}) =& \mu y[g(\vec{x}, y) = 0] \\ &= \text{the least number } y \text{ such that} \\ &\quad (\forall u < y)(\exists w)[g(\vec{x}, u) = w + 1] \ \&\ g(\vec{x}, y) = 0, \end{aligned}$$

with the obvious domain of convergence,

$$\mu y[g(\vec{x},y)=0]\downarrow \iff (\exists y)\Big[(\forall u<y)(\exists w)[g(\vec{x},u)=w+1]\ \&\ g(\vec{x},y)=0\Big];$$

we say that f is defined from g by **minimalization**.

Note that if g is a total function such that for all $\vec{x}$ there is at least one y such that $g(\vec{x},y)=0$, then this is *exactly* minimalization,

$$\mu y[g(\vec{x},y)=0] = \text{the least number } y \text{ such that } g(\vec{x},y)=0;$$

but if (for example) $g(x,0)\uparrow$ and $g(x,1)=0$, then $\mu y[g(x,y)=0]\uparrow$.

We also use the minimalization operation on relations, in the obvious way:

$$\mu y R(\vec{x},y) = \mu y[1 \dot{-} \chi_R(\vec{x},y)=0].$$

Definition 4E.5 (μ-recursion)**.** A **μ-recursive derivation** is a sequence of partial functions on $\mathbb{N}$

$$f_0, f_1, \dots, f_k,$$

where each f_i is S, or a constant C^n_q or a projection P^n_i, or is defined by composition, primitive recursion or minimalization from functions before it in the sequence; and a partial function $f:\mathbb{N}^n \rightharpoonup \mathbb{N}$ is **μ-recursive** if it occurs in a μ-recursive derivation.

In interpreting this definition, we must understand the operations of *composition* and *primitive recursion* correctly for partial functions, for example

$$f(g(\vec{x}),h(\vec{x}))=w \iff (\exists u)(\exists v)[g(\vec{x}=u\ \&\ h(\vec{x})=v)\ \&\ f(u,v)=w].$$

It is clear that every primitive recursive function is μ-recursive, and that the class of μ-recursive partial functions is closed under composition, primitive recursion and minimalization.

The next result is proved by a sequence of tedious constructions of deterministic Turing machines ("Turing machine programming") which we will omit:

Theorem 4E.6. *Every μ-recursive partial function $f:\mathbb{N}^n \rightharpoonup \mathbb{N}$ is Turing computable.*

4E.7. Coding. The converse—and much of the elementary theory of Turing computable functions—is derived by coding the theory of a fixed (possibly non-deterministic) Turing machine $M=(S,Q_0,\Sigma,\sqcup,\text{Table})$ as follows.

If $S=\{Q_0,\dots,Q_a\}$, let $[Q_i]=i$, so that 0 is the code of the initial state and the relation

$$\begin{aligned}\text{State}_M(i) &\iff i \text{ is the code of a state of } M\\ &\iff i\le a\end{aligned}$$

is primitive recursive. Similarly, if $\Sigma = \{\sqcup, R_1, \dots, R_b\}$, let $[R_j] = j$, so that 0 is the code of $\sqcup$ and the relation

$$\begin{aligned}\text{Symbol}_M(j) &\iff j \text{ is the code of a symbol of } M\\ &\iff j \le b\end{aligned}$$

is also primitive recursive.

The coding of tapes is messier, because we have to deal with negative numbers and tapes are "infinite", albeit with only finitely many symbols on them. It is convenient to allow many codes for the same tape. We let

$$\text{Tape}_M(t) \iff \text{Seq}(t) \;\&\; (\forall i < \text{lh}(t))[\text{Symbol}_M((t)_{i,0}) \;\&\; \text{Symbol}_M((t)_{i,1})],$$

and with each t such that $\text{Tape}_M(t)$ we associate the tape

$$\tau_t(i) = \begin{cases} R_{(t)_{i,0}} & \text{if } 0 \le i < \text{lh}(t)\\ R_{(t)_{-i,1}} & \text{if } i < 0 \text{ and } -i < \text{lh}(t)\\ \sqcup & \text{otherwise,}\end{cases}$$

where we have used the notation $(u)_{i,j}$ for the j'th component of the i'th component of the sequence code u

$$(u)_{i,j} = ((u)_i)_j. \tag{103}$$

It is that clear the tape relation is primitive recursive, that every tape gets many codes by this definition, and that "decoding" the tape from any of its codes is "primitive recursive".

Situations are coded as triples of codes, as usual:

$$\text{Sit}_M(s) \iff \text{Seq}(u) \;\&\; \text{lh}(s) = 3 \;\&\; \text{State}_M((s)_0) \;\&\; \text{Tape}_M((s)_1);$$

here, $(s)_2$ codes $i \in \mathbb{Z}$ in some fixed way, e.g., $\text{place}(s) = (s)_{2,0}$ if $(s)_{2,1} = 0$, and $\text{place}(s) = -(s)_{2,0}$ if $(s)_{2,1} > 0$.

With these definitions it is not hard to verify that the relation

$$\begin{aligned}\text{Next}_M(s, s') \iff\ & s \text{ codes a situation } \overline{s}\\ & \;\&\; s' \text{ codes a situation } \overline{s}'\\ & \;\&\; \overline{s}' \text{ is a next situation to } \overline{s}\end{aligned}$$

is primitive recursive, and, using it,

$$\text{Terminal}_M(s) \iff s \text{ is a code of a terminal situation}$$

is primitive recursive too.

Theorem 4E.8. *For each Turing machine* $M = (S, Q_0, \Sigma, \sqcup, \text{Table})$*:*

(1) *The relation*

$$\begin{aligned}\text{Comp}_M(y) &\iff y \text{ is a code of a convergent computation of } M\\ &\iff \text{Seq}(y)\\ &\qquad \& \ (\forall i < \text{lh}(y))[i+1 < \text{lh}(y) \Longrightarrow \text{Next}_M((y)_i, (y)_{i+1})]\\ &\qquad \& \ \text{Terminal}_M((y)_{\text{lh}(y) \dot{-} 1})\end{aligned}$$

is primitive recursive.

(2) *For each* n*, there is a primitive recursive function* $\text{input}_n : \mathbb{N}^n \to \mathbb{N}$ *such that for each tuple* $\vec{x} = (x_1, \dots, x_n)$*,* $\text{input}_n(\vec{x})$ *is a code of the initial situation* $(Q_0, \text{in}(\vec{x}), 0)$*.*

(3) *There is a primitive recursive function* $\text{output}(s)$*, such that if* s *is a code of a terminal situation* (Q, τ', j) *and* $\text{out}(\tau') = \underbrace{11\dots1}_{w+1}$*, then* $\text{output}(s) = w$*.*

(4) *If a partial function* $f : \mathbb{N}^n \rightharpoonup \mathbb{N}$ *is computed by a* (possibly nondeterministic) *Turing machine, then it is* μ*-recursive.*

In particular: *every Turing computable partial function is* μ*-recursive.*

PROOF. (1) is immediate and (2) and (3) are verified by simple (if messy) explicit constructions. For (4), we note that, by the definitions,

$$\begin{aligned}&f(\vec{x}) = w\\ &\quad \iff (\exists y)[\text{Comp}_M(y) \ \& \ (y)_0 = \text{input}_n(\vec{x}) \ \& \ \text{output}((y)_{\text{lh}(y) \dot{-} 1}) = w],\end{aligned}$$

so that the graph of f satisfies an equivalence of the form

$$f(\vec{x}) = w \iff (\exists y) R(\vec{x}, w, y)$$

with a primitive recursive relation R; but then

$$f(\vec{x}) = \Big(\mu y R(\vec{x}, (y)_0, (y)_1)\Big)_0,$$

and $f(\vec{x})$ is μ-recursive. ⊣

We introduce one more, proof-theoretic notion of computability for partial functions (due to Gödel), and a useful variation.

Definition 4E.9 (Reckonability)**.** Suppose $f : \mathbb{N}^n \rightharpoonup \mathbb{N}$, $\mathbf{F}(v_1, \dots, v_n, y)$ is a full extended formula in the language of PA, and T is a theory in the language of PA. We say that $\mathbf{F}(v_1, \dots, v_n, y)$ **reckons** f **in** T if for all $\vec{x}, w$,

$$f(\vec{x}) = w \iff T \vdash \mathbf{F}(\Delta x_1, \dots, \Delta x_n, \Delta w);$$

$\mathbf{F}(v_1, \dots, v_n, y)$ **soundly reckons** f **in** T if for all $\vec{x}, w$, the following two conditions hold:

$$f(\vec{x}) = w \implies T \vdash \mathbf{F}(\Delta x_1, \dots, \Delta x_n, \Delta w),$$
$$\mathbf{N} \models \mathbf{F}(\Delta x_1, \dots, \Delta x_n, \Delta w) \implies f(\vec{x}) = w.$$

It is clear that if T is sound and f is soundly reckonable in T, then f is reckonable in T, but otherwise these two notions are not easily related.

Theorem 4E.10. *For a partial function $f : \mathbb{N}^n \rightharpoonup \mathbb{N}$, the following are equivalent:*

(1) *f is μ-recursive.*

(2) *f is soundly reckonable in Q.*

(3) *f is reckonable in Q.*

(4) *f is reckonable in some axiomatizable theory T in the language of PA.*

(5) *The graph of f satisfies an equivalence of the form*

$$f(\vec{x}) = w \iff (\exists y) R(\vec{x}, w, y), \tag{104}$$

with some primitive recursive relation $R(\vec{x}, w, y)$.

PROOF. (1) $\implies$ (2). It is enough to show that the class of partial functions which are soundly reckonable in Q contains the basic S, C_q^n and P_i^n and is closed under composition, primitive recursion and minimalization. We outline the argument for the last case, the others being similar (and a bit simpler).

So suppose that

$$f(x) = \mu y[g(x, y) = 0]$$

(taking a function of one variable for simplicity) and $\mathbf{G}(v_1, v_2, w)$ soundly reckons $g(x, y)$ in Q, and set

$$\mathbf{F}(v_1, y) \equiv \mathbf{G}(v_1, y, 0) \ \&\ (\forall z < y)\exists w \mathbf{G}(v_1, z, S(w)).$$

To prove that this formula reckons f soundly in Q, assume first that $f(x) = y$, so that

$$\begin{aligned} \mathsf{Q} \vdash \quad & \mathbf{G}(\Delta x, \Delta 0, \Delta w_0) \\ \& \ & \mathbf{G}(\Delta x, \Delta 1, \Delta w_1) \\ & \vdots \\ \& \ & \mathbf{G}(\Delta x, \Delta(y-1), \Delta w_{y-1}) \\ \& \ & \mathbf{G}(\Delta x, \Delta y, \Delta 0) \end{aligned}$$

with suitable numbers $w_z \neq 0$ for $z < y$. The required conclusion, that $\mathsf{Q} \vdash \mathbf{F}(\Delta x, \Delta y)$ follows easily, by appealing to basic properties of Q.

To verify the second condition required of sound reckonability, suppose $\mathbf{N} \models \mathbf{F}(\Delta x, \Delta y)$, so that there are numbers $w_0, \ldots, w_{y-1}$ all > 0, such that

$$\mathbf{N} \models \mathbf{G}(\Delta x, \Delta y, 0), \mathbf{G}(\Delta x, 0, \Delta w_0), \ldots, \mathbf{G}(\Delta x, \Delta(y-1), \Delta w_{y-1};$$

now the hypothesis about g easily implies that $f(x) = y$.

(2) $\Longrightarrow$ (3) follows immediately from the soundness of Q, and (3) $\Longrightarrow$ (4) is trivial, taking $T = \mathsf{Q}$.

(4) $\Longrightarrow$ (5) The hypothesis implies that

$$f(\vec{x}) = w \iff (\exists y)\mathrm{Proof}_T(\#\mathbf{F}(\Delta x_1, \ldots, \Delta x_n, \Delta w), y)$$

so that f satisfies (104) with

$$R(\vec{x}, w, y) \iff \mathrm{Proof}_T(\#\mathbf{F}(\Delta x_1, \ldots, \Delta x_n, \Delta w), y),$$

which is primitive recursive.

(5) $\Longrightarrow$ (1) If f satisfies (104) with a primitive recursive R, then as in the proof of (4) of Theorem 4E.8,

$$f(\vec{x}) = \Big(\mu t R(\vec{x}, (t)_0, (t)_1)\Big)_0,$$

so that f is μ-recursive. ⊣

Thus for any partial function $f : \mathbb{N}^n \rightharpoonup \mathbb{N}$,

$$\begin{aligned} f \text{ is Turing computable} &\iff f \text{ is } \mu\text{-recursive} \\ &\iff f \text{ is reckonable in } \mathsf{Q} \\ &\iff f \text{ is reckonable in some axiomatizable } T \end{aligned}$$

and these equivalences are part of the evidence for the Church-Turing Thesis.

Definition 4E.11. (Recursive partial functions and relations) From now on we will call **computable** or **recursive** the number-theoretic partial functions which are "μ-recursive" (equivalently: "Turing computable", etc.); and we will call **decidable** or **recursive** the relations on $\mathbb{N}$ whose characteristic function is recursive. The term "recursive" is the most common appellation for this class of partial functions and relations, and so we wii tend to use it most often; it derives not so much from μ-recursiveness but from another, fundamental characterization of computability which we will not introduce just yet.

Here is a corollary of Theorem 4E.10 that appeals to condition (5):

Corollary 4E.12. *Recursive functions and recursive relations on* $\mathbb{N}$ *are arithmetical, and, in particular, the truth relation* $\text{Truth}^{\mathbf{N}}(e)$ *for* $\mathbf{N}$ *is not recursive.*

PROOF. Every recursive function is reckonable, and so its graph satisfies an equivalence (104) with primitive recursive—and hence arithmetical—R, so it is arithmetical. The second claim follows from this and Tarski's Theorem 4A.5.⊣

Corollary 4E.13. *Let (5′) be the result of replacing "primitive recursive" by recursive in condition (5) of Theorem 4E.10. A partial function* $f : \mathbb{N}^n \rightharpoonup \mathbb{N}$ *is recursive if and only if it satisfies (5′).*

PROOF. Trivially (5) implies (5′), and the proof that (5) implies (1) also shows that (5′) implies (1). ⊣

Corollary 4E.14 (Definition by cases)**.** *If* $P(\vec{x})$ *is a recursive relation,* g_1 *and* g_2 *are recursive partial functions, and*

$$f(\vec{x}) = \begin{cases} g_1(\vec{x}), & \text{if } P(\vec{x}), \\ g_2(\vec{x}), & \text{otherwise,} \end{cases} \tag{105}$$

then f *is recursive.*

PROOF. Given representations of g_1 and g_2 of the form (104) with respective primitive recursive relations $R_1(\vec{x}, w, y)$ and $R_2(\vec{x}, w, y)$, we verify easily that

$$f(\vec{x}) = w \iff (\exists y)\Big[(P(\vec{x}) \ \&\ R_1(\vec{x}, w, y)) \vee (\neg P(\vec{x}) \ \&\ R_2(\vec{x}, w, y))\Big].$$

It is not hard to show that the relation within the brackets is recursive (e.g., that it is reckonable in Q) and so f is recursive by (5′). ⊣

4F. The basic undecidability results

The results in the preceding section add up to the following basic theorem, which is the key tool for proving undecidability theorems:

Theorem 4F.1 (Kleene's Normal form and Enumeration Theorem)**.** *Let*

$$U(y) = (y)_0$$

(to agree with classical notation), *and for each* n, *let*

$$\begin{aligned} T_n(e, x_1, \dots, x_n, y) \iff\ & e \text{ is the code of a} \\ & \text{full extended formula } \psi(\mathrm{v}_0, \dots, \mathrm{v}_n) \\ & \text{in which } \mathrm{v}_0, \dots, \mathrm{v}_n \text{ actually occur (free)} \\ & \text{and } (y)_1 \text{ is the code of a proof in } \mathsf{Q} \\ & \text{of the sentence } \psi(\Delta x_1, \dots, \Delta x_n, \Delta(y)_0), \\ \varphi_e^n(x_1, \dots, x_n) \quad = \quad & U(\mu y T_n(e, x_1, \dots, x_n, y)), \end{aligned}$$

(1) *The function* $U(y)$ *and each relation* $T_n(e, \vec{x}, y)$ *are primitive recursive.*

(2) *Each* $\varphi_e^n(\vec{x})$ *is a recursive partial function, and so is the partial function which "enumerates" all these,*

$$\varphi^n(e, \vec{x}) = \varphi_e^n(\vec{x}).$$

(3) *For each recursive partial function* $f(x_1, \dots, x_n)$ *of* n *arguments, there exists some* e (a code of f) *such that*

$$f(\vec{x}) = \varphi_e^n(\vec{x}) = U(\mu y T_n(e, x_1, \dots, x_n, y)), \tag{106}$$

so that for each n, *the sequence*

$$\varphi_0^n, \varphi_1^n, \varphi_2^n, \dots$$

enumerates all n*-ary recursive partial functions.*

PROOF. Only (3) needs to be proved, and for that we let e be the code of some formula $\psi(\mathrm{v}_0, \dots, \mathrm{v}_n)$ which reckons f in Q by Theorem 4E.10 and which is (easily) adjusted so that $\mathrm{v}_0, \dots, \mathrm{v}_n$ are the first $n+1$ individual variables and they all actually occur free in it; the verification of (131) is immediate. ⊣

Note: The technical requirement on the free variables of $\psi(\mathrm{v}_0, \dots, \mathrm{v}_n)$ is not needed for this proof; it will be useful in the proof of Theorem 5A.1 further on, and it is just convenient to include it in the definition of the T-predicate now.

Theorem 4F.2 (Undecidability of the Halting problem, Turing)**.** *The relation*

$$H(e, x) \iff \varphi_e^1(x)\!\downarrow \quad (\iff (\exists y) T_1(e, x, y))$$

is undecidable.

PROOF. If $H(e, x)$ were a recursive relation, then the total function

$$f(x) = \begin{cases} \varphi_x^1(x) + 1 & \text{if } \varphi_x^1(x)\!\downarrow \\ 0 & \text{otherwise} \end{cases}$$

would be recursive, and so for some e and all x we would have

$$\varphi^1_e(x) = f(x) = \varphi^1_x(x) + 1;$$

but this is absurd for $x = e$. ⊣

The proof uses the undecidability of the "diagonal" relation

$$K(e) \iff (\exists y)T_1(e, e, y)$$

which is often useful in getting undecidability results. In fact most (elementary) undecidability results are shown by proving an equivalence of the form

$$P(\vec{x}) \iff R(f(\vec{x})),$$

where $f(\vec{x})$ is a recursive function and $P(\vec{x})$ a known, undecidable relation, often $H(e, x)$ or $K(e)$; this is called a **reduction** of $P(\vec{x})$ to $R(u)$, and it implies immediately that $R(u))$ cannot be recursive, else $P(\vec{x})$ would be too. Some of these applications appeal also to the following, trivial

Lemma 4F.3. *If* $\mathbf{T}(v_1, v_2, v_3)$ *is a formula which numeralwise expresses the primitive recusive relation* $T_1(e, x, y)$ *in* Q*, then*

$$\begin{aligned} H(e, x) \iff \varphi^1_e(x)\downarrow &\iff (\exists y)T_1(e, x, y) \\ &\iff \mathsf{Q} \vdash (\exists y)\mathbf{T}(\Delta e, \Delta x, y) \\ &\iff \mathbf{N} \models (\exists y)\mathbf{T}(\Delta e, \Delta x, y). \end{aligned}$$

Definition 4F.4. A theory T in a finite signature τ is **decidable**, if (the characteristic function of) the set (of codes of) its theorems

$$\#T = \{\#\theta \mid \theta \text{ is a } \tau\text{-sentence and } T \vdash \theta\}$$

is decidable , otherwise T is **undecidable**.

The next result extends considerably Corollary 4E.12:

Theorem 4F.5. *If* T *is a sound extension of* Q *in the language of* PA*, then* T *is undecidable.*

In particular, Q *and* PA *are undecidable.*

PROOF. By Lemma 4F.3 and the hypothesis, for any $e, x \in \mathbb{N}$,

$$H(e, x) \Longrightarrow \mathsf{Q} \vdash \exists y\mathbf{T}(\Delta e, \Delta x, y) \Longrightarrow T \vdash \exists y\mathbf{T}(\Delta e, \Delta x, y);$$

and, conversely, by the assumed soundness of T,

$$T \vdash \exists y\mathbf{T}(\Delta e, \Delta x, y) \Longrightarrow \mathbf{N} \models \exists y\mathbf{T}(\Delta e, \Delta x, y) \Longrightarrow H(e, x),$$

again by Lemma 4F.3. Thus

$$H(e, x) \iff T \vdash \exists y\mathbf{T}(\Delta e, \Delta x, y),$$

and so if T were decidable so would $H(e,x)$ be decidable, which it is not. ⊣

The undecidability of Q also yields the undecidability of logical provability (i.e., logical truth):

Theorem 4F.6 (Church's Theorem)**.** *For some finite signature τ, the relation*

$$\mathrm{Th}_\tau(e) \iff e \text{ is the code of a sentence } \theta \text{ of } \mathbb{FOL}(\tau) \text{ and } \vdash \theta$$

is undecidable.

PROOF. We take the signature τ of the language of arithmetic, and notice that if α_{Q} is the conjunction of the (finitely many) axioms of Robinson's Q, then for an arbitrary θ in this language,

$$\mathsf{Q} \vdash \theta \iff \vdash \alpha_{\mathsf{Q}} \to \theta,$$

and so by Lemma 4F.3,

$$H(e,x) \iff \vdash \alpha_{\mathsf{Q}} \to (\exists y)\mathbf{T}(\Delta e, \Delta x, y);$$

but the function

$$g(e,x) = \#(\alpha_{\mathsf{Q}} \to (\exists y)\mathbf{T}(\Delta e, \Delta x, y))$$

is primitive recursive, and so

$$H(e,x) \iff \mathrm{Th}(g(e,x))$$

and $\mathrm{Th}(e)$ cannot be recursive, since $H(e,x)$ is not. ⊣

To extend Theorem 4F.5 to consistent theories in languages richer than he language of PA and not necessarily sound, we need the following simple extension of the undecidability of the Halting Problem:

Theorem 4F.7. *There is a recursive partial function $u : \mathbb{N} \rightharpoonup \{0,1\}$ which has no recursive, total extension.*

PROOF. We let

$$u(t) = 1 \dot{-} \varphi_t(t) = 1 \dot{-} U(\mu y T_1(t,t,y)). \tag{107}$$

This is evidently μ-recursive. Suppose, towards a contradiction, that $f : \mathbb{N} \to \mathbb{N}$ is a total, recursive function which extends u, i.e., such that

$$u(t)\!\downarrow \implies u(t) = f(t),$$

and let e be a code of f, so that $f = \varphi_e$. Now

$$u(e) = 1 \dot{-} \varphi_e(e) = f(e) = \varphi_e(e),$$

which is absurd when $\varphi_e(e)\!\downarrow$. ⊣

Theorem 4F.8. *If T is a consistent theory in a language $\mathbb{FOL}(\tau)$ with finite τ and Q is interpretable in T, then T is undecidable.*

PROOF. Let $u : \mathbb{N} \rightharpoonup \{0,1\}$ be a recursive partial function which has no total, recursive extension, by Theorem 4F.7, and let $\phi(v,y)$ be a full extended formula which numeralwise represents u in Q, so that

$$\text{if } u(t) = w, \text{ then } \mathsf{Q} \vdash \phi(\Delta t, \Delta w) \text{ and } \mathsf{Q} \vdash \exists! y \phi(\Delta t, y).$$

In particular,

$$u(t) = 0 \Longrightarrow \mathsf{Q} \vdash \phi(\Delta t, 0). \tag{108}$$

We claim that also

$$u(t) = 1 \Longrightarrow \mathsf{Q} \vdash \neg\phi(\Delta, 0); \tag{109}$$

this is because if $u(t) = 1$, then (writing 1 for $\Delta 1$),

$$\mathsf{Q} \vdash \phi(\Delta t, 1) \ \& \ \exists! y \phi(\Delta t, y),$$

from which we get immediately get that $\mathsf{Q} \vdash \neg\phi(\Delta t, 0)$, since $\mathsf{Q} \vdash 0 \neq 1$. If π is the assumed interpretation of Q in T, then (108), (109) and one of the basic properties of interpretations yield that

$$u(t) = 0 \Longrightarrow T \vdash \pi\phi(\Delta t, 0), \quad u(t) = 1 \Longrightarrow T \vdash \neg\pi\phi(\Delta t, 0). \tag{110}$$

Now let

$$f(t) = \begin{cases} 0, & \text{if } T \vdash \pi\phi(\Delta t, 0), \\ 1, & \text{otherwise.} \end{cases}$$

This is a total function, and if T is a decidable theory, it is recursive. Clearly

$$u(t) = 0 \Longrightarrow f(t) = 0 = u(t);$$

and since T is consistent and so cannot prove both $\pi\phi(\Delta t, 0)$ and $\neg\pi\phi(\Delta t, 0)$, (110) implies that

$$u(t) = 1 \Longrightarrow f(t) = 1 = u(t).$$

Thus f is a total, recursive extension of u, which is a contradiction. ⊣

Notice that Theorem 4F.8 is a direct generalization of Rosser's Theorem 4C.4, because of Problem 5A.2.