3. THE THEOREMS OF TARSKI AND GÖDEL

Contents

One way to interpret the Completeness Theorem is that it identifies **logical truth** with **provable** or *justified truth*: if a τ-sentence χ is true under all interpretations of the constants, relation symbols and function symbols of τ in every universe of discourse, then χ is formally provable in $\mathsf{LPCI}(\tau)$. In this handout we will prove in outline the celebrated theorems of Tarski and (especially) Gödel which prohibit this sort of reduction for **arithmetical truth**: in a sense which we will make precise (and plausible), there is no useful reduction of arithmetical truth to provability in some theory.

These results are quite general, but we will keep the discussion concrete by concentrating on the (fixed for the sequel) vocabulary

$$\tau_{\mathsf{PA}} = (0, S, +, \cdot)$$

of Peano arithmetic (PA) and the fixed standard model

$$\mathbf{N} = (\mathbb{N}, 0, S, +, \cdot)$$

of it. The proofs depend on the method of **arithmetization** (or *Gödel numbering*) which assigns **number codes** to the symbols, terms, formulas and proofs of PA and most any other interesting τ_{PA}-theory T: we can then interpret the formulas of $\mathsf{LPCI}(\tau_{\mathsf{PA}})$ as making (indirectly) claims about T, and a judicious choice of *self-referential* such claims will yield the theorems.

We have put in §1 a version of what is sometimes called *Gödel's long computation*, the technical facts we need about these codings. These are most easily proved by appealing repeatedly to Theorem 3J.1 of LPCI.

§1. Arithmetization. Using standard, set-theoretic notation, we let

$$\mathbb{N}^{<\omega} = \text{the set of all finite sequences of natural numbers.}$$

Instead of the cumbersome tuple-coding of $\mathbb{N}^{<\omega}$ in terms of the β-function, we will use the following, now standard coding of tuples of numbers:

1.1. **Theorem** (Sequence coding)**.** *For each $n \geq 1$, let*

$$f_n(x_0, \dots, x_{n-1}) = p_0^{x_0+1} p_1^{x_1+1} \cdots p_{n-1}^{x_{n-1}+1}.$$

(1) *Each $f_n : \mathbb{N}^n \to \mathbb{N}$ is arithmetical and one-to-one.*

(2) *The mapping $\langle\ \rangle : \mathbb{N}^{<\omega} \to \mathbb{N}$ defined by*

$$\langle \emptyset \rangle = 1, \qquad \langle x_0, \dots, x_{n-1} \rangle = f_n(x_0, \dots, x_{n-1})$$

is one-to-one.

(3) *The unary relation*

$$\mathrm{Seq}(u) \iff u = 1 \vee (\exists n, x_0, \dots, x_{n-1})[u = f_n(x_0, \dots, x_{n-1})]$$

is arithmetical.

(4) *There is a unary arithmetical function* $\mathrm{lh}(u)$ *such that*

$$\mathrm{lh}(1) = 0, \qquad \mathrm{lh}(\langle x_0, \dots, x_{n-1} \rangle) = n.$$

(5) *There is binary, arithmetical function* $\mathrm{proj}(u, i)$ *such that*

$$\text{if } u = \langle x_0, \dots, x_{n-1} \rangle \text{ and } i < n, \text{ then } \mathrm{proj}(u, i) = x_i.$$

Proof is easy, and we will leave it for the problems. ⊣

We call $\langle x_0, \dots, x_{n-1} \rangle$ the **code** of the finite sequence $(x_0, \dots, x_{n-1})$. Notice that 0 is not a code, 1 codes the empty sequence, and the relation $\mathrm{Seq}(u)$ of "being a code" is arithmetical by (3). The function $\mathrm{lh}(u)$ gives us the length of the sequence (coded by) u, if u codes a sequence, and $\mathrm{proj}(u, i)$ gives us the i'th term of that sequence, if $i < \mathrm{lh}(u)$. To simplify notation, we will write

$$(u)_i = \mathrm{proj}(u, i), \quad (u)_{i,j} = ((u)_i)_j, \quad \dots$$

rather than $\mathrm{proj}(u, i)$, $\mathrm{proj}(\mathrm{proj}(u, i), j)$ $\dots$.

1.2. **Lemma** (Concatenation)**.** *There is a binary arithmetical function $u * v$ such that*

(1) *if u or v are not both sequence codes, then $u * v = 0$; and*

(2) *for any two finite sequences $(x_0, \dots, x_{n-1}), (y_0, \dots, y_{m-1})$,*

$$\langle x_0, \dots, x_{n-1} \rangle * \langle y_0, \dots, y_{m-1} \rangle = \langle x_0, \dots, x_{n-1}, y_0, \dots, y_{m-1} \rangle.$$

In particular, if $\mathrm{Seq}(u)$*, then $u * 1 = 1 * u = u$.*

Proof is easy, cf. Problem x3.3. ⊣

The *concatenation* operation $u * v$ is easily associative (Problem x3.3)

$$(u * v) * w = u * (v * w), \tag{1-1}$$

and so we will write $u * v * w$, $u * v * w * z$, etc., without indicating any specific grouping.

1.3. **Definition** (The coding of $\mathsf{LPCI}(\tau_{\mathsf{PA}})$)**.** We assign numbers to the *symbols* of $\mathsf{LPCI}(\tau_{\mathsf{PA}})$ by enumerating them as follows:

$\neg$	$\wedge$	$\vee$	$\to$	$\forall$	$\exists$	$=$	$($	$)$	$,$	0	S	$+$	$\cdot$	v_0	v_1	v_2	$\cdots$
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	$\cdots$

so that $\mathrm{sc}(\neg) = 0$, $\mathrm{sc}(\wedge) = 1, \dots, \mathrm{sc}(\cdot) = 13$ and $\mathrm{sc}(\mathrm{v}_i) = 14 + i$.

Using these *symbol codes*, we can assign codes to *strings of symbols* by using the coding of sequences above:

$$\#(a_0 a_2 \dots a_{n-1}) = \langle \mathrm{sc}(a_0), \dots, \mathrm{sc}(a_{n-1}) \rangle;$$

this defines codes for the terms and formulas of $\mathsf{LPCI}(\tau_{\mathsf{PA}})$, since they are sequences of symbols.

Finally, we can use the *string codes* to assign codes to sequences of strings by the same method:

$$@(\alpha_0, \dots, \alpha_{n-1}) = \langle \#(\alpha_0), \dots, \#(\alpha_{n-1}) \rangle.$$

These codings allow the easy extraction of information about the coded object from its code. For example, if $a = \#(a_0 a_2 \dots a_{n-1})$, then $\mathrm{lh}(a)$ is the length of the string α coded by a, $(a)_i$ is the code $\mathrm{sc}(a_i)$ of the i'th symbol in α for $i < \mathrm{lh}(a)$, etc. They are also immense: for example, for the simplest formula with a variable,

$$\#(\mathrm{v}_0 = 0) = \langle \mathrm{sc}(\mathrm{v}_0), \mathrm{sc}(=), \mathrm{sc}(0) \rangle = 2^{15} 3^7 5^{11}$$

This is not important for what we will do with them.

1.4. **Lemma.** *The following relations and functions are arithmetical:*

$$\begin{aligned}
\mathrm{Zero}(s) &\iff s \text{ is the code of } 0 \\
&\iff s = \mathrm{sc}(0) \\
\mathrm{Variable}(v) &\iff v \text{ is the code of a variable} \\
&\iff v \geq 14 \\
\mathrm{String}(s) &\iff s \text{ is the code of a string of symbols} \\
&\iff \mathrm{Seq}(s) \\
\mathrm{SeqStrings}(y) &\iff y \text{ is a code of a sequence of strings} \\
&\iff \mathrm{Seq}(y) \wedge (\forall i < \mathrm{lh}(y))[\mathrm{String}((y)_i)]
\end{aligned}$$

Proof is trivial using Theorem 3J.1 of LPCI. ⊣

1.5. **Definition** (Derivations of syntactic objects)**.** A *term derivation* is a finite sequence $(\alpha_0, \dots, \alpha_n)$ of strings of symbols such that for each i, one of the following conditions hold:

1. $\alpha_i \equiv 0$
2. α_i is a variable
3. There is a number $j < i$ such that $\alpha_i \equiv S(\alpha_j)$.
4. There are numbers $j, k < i$ such that $\alpha_i \equiv +(\alpha_i, \alpha_k)$.
5. There are numbers $j, k < i$ such that $\alpha_i \equiv \cdot(\alpha_i, \alpha_k)$.

A *formula derivation* is a finite sequence $(\phi_0, \dots, \phi_n)$ of strings of symbols such that for each i, one of the following conditions hold:

1. $\phi_i \equiv s = t$, where s and t are terms.
2. There is a number $j < i$ such that $\phi_i \equiv (\neg\phi_j)$.
3. There are numbers $j, k < i$ such that $\phi_i \equiv (\phi_j \wedge \phi_k)$.
4. There are numbers $j, k < i$ such that $\phi_i \equiv (\phi_j \vee \phi_k)$.
5. There are numbers $j, k < i$ such that $\phi_i \equiv (\phi_j \to \phi_k)$.
6. There is a number $j < i$ and a variable v_k such that $\phi_i \equiv \forall \mathrm{v}_k \phi_j$.
7. There is a number $j < i$ and a variable v_k such that $\phi_i \equiv \exists \mathrm{v}_k \phi_j$.

1.6. **Lemma.** (1) *A string t is a term if and only if it occurs in a term derivation; i.e., if there is a term derivation $(\alpha_0, \dots, \alpha_n)$ such that for some $i \leq n$, $t \equiv \alpha_i$.*

(2) *A string ϕ is a formula if and only if it occurs in a formula derivation.*

Proof. Both of these are easy, using structural induction for the direction ($\Longrightarrow$) and induction on the length n of the given sequence of strings for the direction ($\Longleftarrow$). ⊣

1.7. **Lemma.** *The following relations are arithmetical:*

$$\begin{aligned}
\mathrm{TermDer}(y) &\iff y \text{ is the code of a term derivation} \\
\mathrm{Term}(t) &\iff t \text{ is the code of a term} \\
&\iff (\exists y)[\mathrm{TermDer}(y) \wedge (\exists i < \mathrm{lh}(y))[t = (y)_i]]. \\
\mathrm{FormulaDer}(y) &\iff y \text{ is the code of a formula derivation} \\
\mathrm{Formula}(f) &\iff f \text{ is the code of a formula} \\
&\iff (\exists y)[\mathrm{FormulaDer}(y) \wedge (\exists i < \mathrm{lh}(y))[f = (y)_i]] \\
\mathrm{BoundOcc}(f, i, j) &\iff f \text{ is the code of a formula } \phi \\
&\qquad \text{and } \mathrm{v}_i \text{ occurs bound in position } j \text{ in } \phi \\
\mathrm{Bound}(f, i) &\iff f \text{ is the code of a formula } \phi \\
&\qquad\qquad \text{and } \mathrm{v}_i \text{ occurs bound in } \phi \\
&\iff (\exists j)\mathrm{BoundOcc}(f, i, j)
\end{aligned}$$

$$\begin{aligned}
\text{FreeOcc}(f,i,j) &\iff f \text{ is the code of a formula } \phi \\
&\qquad \text{and } \mathrm{v}_i \text{ occurs free in position } j \text{ in } \phi \\
\text{Free}(f,i) &\iff f \text{ is the code of a formula } \phi \\
&\qquad\qquad \text{and } \mathrm{v}_i \text{ occurs free in } \phi \\
&\iff (\exists j)\text{FreeOcc}(f,i,j) \\
\text{Sentence}(c) &\iff c \text{ is the code of a sentence} \\
&\iff \text{Formula}(c) \wedge (\forall i)\neg\text{Free}(c,i).
\end{aligned}$$

PROOF. The key fact is Lemma 1.6 and for most of the claims in the Lemma we have included the proof. We outline the necessary computation for the coding of terms. Set first:

$$\begin{aligned}
\text{UnDer}(a,b) &\iff \text{for some strings } \alpha \text{ and } \beta, \\
&\qquad b = \#(\beta) \text{ and } a = \#(S(\beta)) \\
&\iff \text{Seq}(b) \wedge a = \langle \text{sc}(S), \text{sc}(() \rangle * b * \langle \text{sc}()) \rangle \\
\text{BinDer}(a,b,c) &\iff \text{for some strings } \alpha,\beta,\gamma, b = \#(\beta), c = \#(\gamma) \\
&\qquad \text{and } \alpha \equiv +(\beta,\gamma) \text{ or } \alpha \equiv \cdot(\beta,\gamma) \\
&\iff \text{Seq}(b) \wedge \text{Seq}(c) \\
&\qquad \wedge\ [a = \langle \text{sc}(+) \rangle * b * \langle \text{sc}(,) \rangle * c * \langle \text{sc}()) \rangle \\
&\qquad\qquad \vee a = \langle \text{sc}(\cdot) \rangle * b * \langle \text{sc}(,) \rangle * c * \langle \text{sc}()) \rangle]
\end{aligned}$$

These relations are clearly arithmetical, and directly from the definition:

$$\begin{aligned}
\text{TermDer}(y) \iff\ & \text{Seq}(y) \wedge (\forall i < \text{lh}(y))\text{Seq}((y)_i) \\
& \wedge (\forall i < \text{lh}(y))\Big([\text{lh}((y)_i) = 1 \wedge \text{Zero}((y)_{i,0})] \\
& \qquad \vee [\text{lh}((y)_i) = 1 \wedge \text{Variable}((y)_{i,0})] \\
& \qquad \vee (\exists j < i)\text{UnDer}((y)_i, (y)_j) \\
& \qquad \vee (\exists j,k < i)\text{BinDer}((y)_i, (y)_j, (y)_k) \Big)
\end{aligned}$$

This completes the proof that $\text{Term}(t)$ is arithmetical, and the argument is similar for $\text{Formula}(f)$.

For the relation $\text{BoundOcc}(f,i,j)$, we use the fact that v_i occurs bound in the position j of the formula ϕ if $\phi \equiv \alpha\psi\beta$ where α or β may be empty, but ψ is a formula which starts with either $\exists \mathrm{v}_i$ or $\forall \mathrm{v}_i$, and the j'th position in ϕ occurs in ψ. So

$$\begin{aligned}
\text{BoundOcc}(f,i,j) \iff\ & \text{Formula}(f) \wedge (f)_j = \text{sc}(\mathrm{v}_i) \\
& \wedge (\exists a,c,b)\Big[f = a * c * b \wedge \text{Seq}(a) \wedge \text{Seq}(b) \wedge \text{Formula}(c) \\
& \qquad \wedge\ [\text{lh}(a) < j < \text{lh}(a * c)] \\
& \qquad \wedge\ [(c)_0 = \text{sc}(\exists) \vee (c)_0 = \text{sc}(\forall)] \wedge (c)_1 = \text{sc}(\mathrm{v}_i) \Big]
\end{aligned}$$

The remaining parts of the Lemma are simple, cf. Problem x3.6. ⊣

Recall the function $\Delta(n)$ which associates with each natural number n the simplest, closed term which names n in the language of $\mathsf{LPCI}(\tau_{\mathsf{PA}})$ (its *numeral*). It is defined by the simple recursion

$$\Delta(0) \equiv 0, \quad \Delta(n+1) \equiv S(\Delta(n)), \tag{1-2}$$

1.8. **Lemma.** *The function $\delta(n) = \#(\Delta(n))$ is arithmetical.*

Proof. We define $\delta(n)$ by a primitive recursion which is modeled after the recursive definition of the terms $\Delta(n)$:

$$\begin{aligned} \delta(0) &= \#(0) = \langle \mathrm{sc}(0) \rangle, \\ \delta(S(n)) &= \langle \mathrm{sc}(S), \mathrm{sc}(() \rangle * \delta(n) * \langle \mathrm{sc}()) \rangle. \end{aligned}$$ ⊣

§2. The theorems. We start with a better way of representing arithmetical relations, which utilizes the fact that every number is named by a numeral:

2.1. **Lemma.** *A unary relation $R(n)$ is arithmetical if and only if there exists a formula ϕ in which v_0 does not occur bound and no variable other than v_0 occurs free, such that for every n,*

$$R(n) \iff \mathbf{N} \models \exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(n)). \tag{2-3}$$

Proof. If ϕ is any formula and π is any assignment, then for every n,

$$\mathbf{N}, \pi\{\mathrm{v}_0 := n\} \models \phi \iff \mathbf{N}, \pi \models \exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(n)). \tag{2-4}$$

This is easily seen using the *Tarski conditions*, Theorem 3F.1 of LPCI as follows.

Suppose first that $\mathbf{N}, \pi\{\mathrm{v}_0 := n\} \models \phi$. To show that

$$\mathbf{N}, \pi \models \exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(n)),$$

we must find some number m such that

$$\mathbf{N}, \pi\{\mathrm{v}_0 := m\} \models \phi \wedge \mathrm{v}_0 = \Delta(n); \tag{2-5}$$

and $m = n$ clearly does it, since $\mathbf{N}, \pi\{\mathrm{v}_0 := n\} \models \mathrm{v}_0 = \Delta(n)$.

Conversely, if $\mathbf{N}, \pi \models \exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(n))$, then there is some m such that (2-5) holds, and this can only be n, since we cannot have

$$\mathbf{N}, \pi\{\mathrm{v}_0 := m\} \models \mathrm{v}_0 = \Delta(n)$$

unless $m = n$. Thus (2-5) holds with $m = n$, which is the left-hand-side of (2-4).

Now, a unary relation $R(n)$ is arithmetical if and only if there is a formula ϕ with at most one free variable v such that (for any π),

$$R(n) \iff \mathbf{N}, \pi\{v := n\} \models \phi.$$

By alphabetic changes of variables, we may assume that $v \equiv \mathrm{v}_0$ and that v_0 does not occur bound in ϕ, and then the Lemma follows from (2-4). ⊣

The main tool for the proofs of the Tarski and Gödel theorems is the following, simple result:

2.2. **Lemma.** *There is an arithmetical function $D(f, n)$ such that for every formula ϕ and every number n,*

$$D(\#(\phi), n) = \#(\exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(n))). \tag{2-6}$$

Proof. This is seen by a direct, explicit definition:

$$D(f, n) = \#\big(\exists \mathrm{v}_0\big) * f * \#(\wedge \mathrm{v}_0 =) * \delta(n) * \langle \mathrm{sc}()\rangle. \qquad \dashv$$

2.3. **Definition.** For any τ_{PA}-theory T, we let

$$\#(T) = \{\#(\chi) \mid \chi \in T\},$$

and we call T *arithmetical* if $\#(T)$ is an arithmetical set. We also set

$$\begin{aligned}\mathrm{Truth}(\mathbf{N}) &= \#(\mathrm{Th}(\mathbf{N}))\\ &= \{c \mid c \text{ is the code of a } \tau_{\mathsf{PA}}\text{- sentence } \chi \text{ such that } \mathbf{N} \models \chi\}.\end{aligned}$$

It is customary to refer loosely to $\mathrm{Truth}(\mathbf{N})$ as "the truth set of arithmetic", basically identifying the set of sentences $\mathrm{Th}(\mathbf{N})$ with the set $\mathrm{Truth}(\mathbf{N})$ of their codes; either way, this is the set of *arithmetical truths*.

2.4. **Theorem** (Tarski's Theorem)**.** *The set* $\mathrm{Truth}(\mathbf{N})$ *of arithmetical truths is not arithmetical.*

Proof. Suppose towards a contradiction that $\mathrm{Truth}(\mathbf{N})$ is arithmetical, and let

$$R(e) \iff D(e, e) \notin \mathrm{Truth}(\mathbf{N}).$$

Now $R(e)$ is arithmetical, and so by Lemma 2.1, there is a formula ϕ in which only v_0 occurs free, v_0 does not occur bound, and

$$R(e) \iff \mathbf{N} \models \exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(e)). \tag{2-7}$$

Let $\overline{e} = \#(\phi)$. Now $\overline{e}$ is the code of a formula, and so by Lemma 2.2,

$$D(\overline{e}, \overline{e}) = \#(\exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(\overline{e}))).$$

By the definition of $R(e)$ then,

$$R(\overline{e}) \iff D(\overline{e}, \overline{e}) \notin \mathrm{Truth}(\mathbf{N}) \iff \mathbf{N} \not\models \exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(\overline{e}));$$

while by (2-7),

$$R(\overline{e}) \iff \mathbf{N} \models \exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(\overline{e})),$$

so that $R(\overline{e}) \iff \neg R(\overline{e})$, which is absurd. ⊣

2.5. **Definition.** For any τ_{PA}-theory T, let

$$\begin{aligned}\mathrm{Proof}_T(y,c) \iff & \ c \text{ is a code of a sentence } \chi \\ & \text{and } y \text{ is the code of a proof of } \chi \text{ from } T \\ \iff & \ c \text{ is a code of a sentence } \chi \\ & \text{and there is a deduction } \phi_0,\dots,\phi_n,\chi \text{ from } T \\ & \text{with code } y = \langle \#(\phi_0),\dots,\#(\phi_n),c\rangle.\end{aligned}$$

This is the *proof relation* of the theory T.

2.6. **Lemma.** (1) *Peano arithmetic* PA *is arithmetical.*

(2) *If T is an arithmetical theory, then its proof relation* $\mathrm{Proof}_T(y,c)$ *is arithmetical.*

Proof. (1) is rather tedious but easy and (2) is proved by the same sort of computation we used in the proofs that the two relations $\mathrm{TermDer}(y)$ and $\mathrm{FormulaDer}(y)$ are arithmetical. We leave them both for problems.⊣

A τ_{PA}-theory T is *sound for* $\mathbf{N}$ if all its theorems are true in $\mathbf{N}$, i.e.,

$$T \vdash \chi \Longrightarrow \mathbf{N} \models \chi;$$

it is *complete for* $\mathbf{N}$ if it proves every τ_{PA}-sentence which is true in $\mathbf{N}$, i.e.,

$$\mathbf{N} \models \chi \Longrightarrow T \vdash \chi.$$

2.7. **Theorem** (Gödel's First Incompleteness Theorem)**.** *No arithmetical theory in the language of* PA *can be both sound and complete for* $\mathbf{N}$*; and in particular,* PA *is not complete.*

We will give two proofs of this theorem, one by contradiction and one constructive.

First proof. If T is sound and complete for $\mathbf{N}$, then

$$T \vdash \chi \iff \mathbf{N} \models \chi.$$

Thus the set of theorems T coincides with the set of sentences which are true in $\mathbf{N}$ and its code set

$$\overline{T} = \{\#(\chi) \mid \chi \text{ is a sentence and } T \vdash \chi\} = \mathrm{Th}(\mathbf{N})$$

which is not arithmetical by Tarski's Theorem. But if T is arithmetical, then $\overline{T}$ is also arithmetical since

$$c \in \overline{T} \iff (\exists y)\mathrm{Proof}_T(y,c). \qquad \dashv$$

Second proof. We assume that T is a sound, arithmetical τ_{PA}-theory, and we will construct a sentence γ_T (the *Gödel sentence of* T) which is true but not a theorem of T, i.e.,

$$\mathbf{N} \models \gamma_T \text{ but } T \nvdash \gamma_T. \tag{2-8}$$

Let

$$P(e) \iff \neg(\exists y)\mathrm{Proof}(y, D(e,e)) \tag{2-9}$$

so that $P(e)$ is an arithmetical relation and by Lemma 2.1, there is a formula ϕ with just v_0 free such that for every number e,

$$P(e) \iff \mathbf{N} \models \exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(e)). \tag{2-10}$$

Let $\overline{e} = \#(\phi)$ and set

$$\gamma_{_T} :\equiv \exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(\overline{e})) \text{ so that } \#(\gamma_{_T}) = D(\overline{e},\overline{e}). \tag{2-11}$$

By (2-9),

$$P(\overline{e}) \iff T \nvdash \gamma_{_T};$$

and by (2-10),

$$P(\overline{e}) \iff \mathbf{N} \models \gamma_{_T},$$

so that we have the equivalence

$$\mathbf{N} \models \gamma_{_T} \iff T \nvdash \gamma_{_T}, \tag{2-12}$$

and its contrapositive

$$\mathbf{N} \not\models \gamma_{_T} \iff T \vdash \gamma_{_T}. \tag{2-13}$$

We now infer that $\boxed{T \nvdash \gamma_{_T}}$: because if $T \vdash \gamma_{_T}$, then $\mathbf{N} \models \gamma_{_T}$ by soundness, contradicting (2-13). Hence, by (2-12), $\boxed{\mathbf{N} \models \gamma_{_T}}$, and the two framed propositions together give (2-8). ⊣

Notice that carefully read and "translated" into mathematical English, the *Gödel sentence* $\gamma_{_T}$ claims its own unprovability from T.

§3. Problems.

x3.1. Prove (3) Theorem 1.1, that the relation

$$\mathrm{Seq}(u) \iff u = 1 \vee (\exists n, x_0, \dots, x_{n-1})[u = f_n(x_0, \dots, x_{n-1})]$$

is arithmetical.

x3.2. Prove (5) of Theorem 1.1, that there is a binary, arithmetical function $\mathrm{proj}(u,i)$ such that

$$\text{if } u = \langle x_0, \dots, x_{n-1}\rangle \text{ and } i < n, \text{ then } \mathrm{proj}(u,i) = x_i.$$

x3.3. Prove Lemma 1.2, that the concatenation function on $\mathbb{N}^{<\omega}$ is arithmetical in the codes and also that it is associative, (1-1).

x3.4. Prove (1) of Lemma 1.6.

x3.5. Determine the free and bound occurrences of variables in the following (misspelled) formula of $\mathsf{LPCI}(\leq)$:

$$\phi \equiv \forall y(x \leq y) \wedge \forall x \exists y(x \leq y \wedge \neg(y \leq x))$$

Which are the free variables of ϕ and which are its bound variables?

x3.6. Prove the claims about Formula(f) and Sentence(c) in Lemma 1.7.

x3.7. Outline a proof of Part (1) of Lemma 2.6.

x3.8. Outline a proof of Part (2) of Lemma 2.6.