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 in 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 forbid 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 of the language of (Peano) arithmetic PA

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

and the fixed standard model

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

of it. The proofs depend on the method of *arithmetization* (or *coding*, or *Gödel numbering*) which assigns number codes to the symbols, terms, formulas and proofs of PA and most any other interesting object of 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 such *self-referential* claims will yield the theorems.

Accordingly, much of the technical work required for these proofs involves the construction of *codings*.

§1. Arithmetization. The arithmetical coding of sequences of numbers by the β function is cumbersome and we will abandon it in favor of the following, much simpler construction. We let

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

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

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

where p_i = the i'th prime number with $p_0 = 2$.

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

(2) *The mapping $\langle\ \rangle : \mathbb{N}^* \to \mathbb{N}$ 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 ease the use of this arithmetical coding of sequences, we will write

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

rather than the most formal but cumbersome $\mathrm{proj}(u, i)$, $\mathrm{proj}(\mathrm{proj}(u, i), j)$.

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

(1) *if u or v are not 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 in Problem x3.2. ⊣

The *concatenation* operation $u * v$ is easily associative,

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

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:

$\neg$	$\wedge$	$\vee$	$\rightarrow$	$\forall$	$\exists$	$=$	$($	$)$	$,$	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

$$\mathrm{sc}[\neg] = 0,\ \mathrm{sc}[\wedge] = 1, \ldots, \mathrm{sc}[\cdot] = 13,$$

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

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

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:

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

Notice that these codings allow the easy extraction of information about the coded object from its code. For example, if $a = \#(a_0 a_2 \ldots a_{n-1})$, then $\mathrm{lh}(a)$ is the length of the string α coded by a, if $i < \mathrm{lh}(a)$, then $(a)_i = \mathrm{sc}[a_i]$ is the code of the i'th symbol in α, etc.

The codes produced in this way are immense: for example, considering the simplest formula $0 = 0$,

$$\#(0 = 0) = \langle \mathrm{sc}[0], \mathrm{sc}[=], \mathrm{sc}[0] \rangle = 2^{11} 3^7 5^{11}.$$

However, size is not the relevant complexity measure: the important fact is that the associated *decoding* relations and functions are arithmetical.

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

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

1.5. **Definition** (Derivations of syntactic objects)**.** A *term derivation* is a finite sequence of strings of symbols

$$\alpha_0, \ldots, \alpha_n$$

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 $\alpha_i \equiv +(\alpha_j, \alpha_k)$.
5. There are numbers $j, k < i$ such that $\alpha_i \equiv \cdot(\alpha_j, \alpha_k)$.

A *formula derivation* is a sequence of strings

$$\alpha_0, \dots, \alpha_n$$

such that for each i, one of the following conditions hold:

1. $\alpha_i \equiv s = t$, where s and t are terms.
2. There is a number $j < i$ such that $\alpha_i \equiv (\neg\alpha_j)$.
3. There are numbers $j, k < i$ such that $\alpha_i \equiv (\alpha_j \wedge \alpha_k)$.
4. There are numbers $j, k < i$ such that $\alpha_i \equiv (\alpha_j \vee \alpha_k)$.
5. There are numbers $j, k < i$ such that $\alpha_i \equiv (\alpha_j \to \alpha_k)$.
6. There is a number $j < i$ and a variable v such that $\alpha_i \equiv \forall v\alpha_j$.
7. There is a number $j < i$ and a variable v such that $\alpha_i \equiv \exists v\alpha_j$.

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 $i \le n$, $t \equiv \alpha_i$.*

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

PROOF is left for Problem x3.3. ⊣

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

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

PROOF. The key fact is Lemma 1.6. We outline the necessary computation for the coding of terms. Set first:

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

These relations are clearly arithmetical and directly from the definition:

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

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

For the relation $\mathrm{BoundOcc}(f,i,j)$, we use the fact that v_i occurs bound in the position j of the formula ϕ, if it is possible to break the string ϕ into three parts

$$\phi \equiv \alpha\psi\beta$$

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. This is expressed by the following:

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

The remaining parts of the Lemma are simple. ⊣

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 $\Delta(n)$:

$$\delta(0) = \#(0) = \langle \mathrm{sc}[0] \rangle,$$
$$\delta(S(n)) = \langle \mathrm{sc}[S], \mathrm{sc}[(] \rangle * \delta(n) * \langle \mathrm{sc}[)] \rangle. \qquad \dashv$$

§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,

$$R(n) \iff \mathbf{N} \models \exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(n)). \tag{2-1}$$

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

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

This is easily seen directly from the clause for the existential quantifier in the definition of satisfaction, 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

$$\mathbf{N}, \pi\{\mathrm{v}_0 := m\} \models \phi \wedge \mathrm{v}_0 = \Delta(n); \tag{2-3}$$

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 (2-3) 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 (2-3) holds with $m = n$, which is the left-hand-side of (2-2).

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 (2-2). $\dashv$

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 each formula* ϕ *and each number* n,

$$D(\#(\phi), n) = \#(\exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(n))). \tag{2-4}$$

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

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

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.

In particular,

$$\text{(2-5)} \quad \begin{aligned} \mathrm{Truth} &= \#(\mathrm{Th}(\mathbf{N})) \\ &= \{c \mid c \text{ is the code of a } \tau_{\mathsf{PA}}\text{- sentence } \chi \text{ and } \mathbf{N} \models \chi\} \end{aligned}$$

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

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

PROOF. Suppose towards a contradiction that Truth is arithmetical and let

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

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

$$\text{(2-6)} \qquad R(e) \iff \mathbf{N} \models \exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(e)).$$

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,

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

while by (2-6),

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

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

The idea of the proof is that it there were a formula ϕ which satisfies (2-6) and if $\overline{e}$ is the code of ϕ, then the sentence $\exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(\overline{e})$ expresses its own falsity, so it cannot be either true or false. Tarski's proof uses a precise, first-order version of the ancient Liar's Paradox which has a liar utter the sentence "*I always lie*" and locates the contradiction in the assumption that arithmetical truth can be defined in a first-order language.

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.

Recall that a theory T is arithmetical if its code set $\#(T)$ is arithmetical.

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. ⊣

A τ_{PA}-theory T is *sound* (for the standard model) if all its axioms are true in $\mathbf{N}$,

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

it is *complete* if for every τ_{PA}-sentence χ,

$$\text{either } T \vdash \chi \text{ or } T \vdash \neg\chi.$$

2.7. **Theorem** (Gödel's First Incompleteness Theorem)**.** *No arithmetical theory in the language of* PA *can be both sound and complete*; *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, then

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

and if T is both sound and complete, then

$$\begin{aligned}\mathbf{N} \models \chi &\Longrightarrow T \nvdash \neg\chi \quad \text{(by soundness)} \\ &\Longrightarrow T \vdash \chi \quad \text{(by completeness).}\end{aligned}$$

Thus the set of theorems of a sound and complete T coincides with the set of true sentences and so its code set

$$T^* = \{\#(\chi) \mid \chi \text{ is a sentence and } T \vdash \chi\}$$

cannot be arithmetical by Tarski's Theorem. But if T is arithmetical, then T^* is also arithmetical since

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

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 decided by T, i.e.,

$$\mathbf{N} \models \gamma_T, \quad T \not\vdash \gamma_T, \quad T \not\vdash \neg\gamma_T. \tag{2-7}$$

Let

$$P(e) \iff \neg(\exists y)\text{Proof}(y, D(e,e)) \tag{2-8}$$

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,

$$P(e) \iff \mathbf{N} \models \exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(e)). \tag{2-9}$$

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

$$\gamma_T :\equiv \exists \mathrm{v}_0(\phi \wedge \mathrm{v}_0 = \Delta(\overline{e})) \text{ so that } \#(\gamma_T) = D(\overline{e}, \overline{e}). \tag{2-10}$$

By (2-8),

$$P(\overline{e}) \iff T \not\vdash \gamma_T;$$

and by (2-9),

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

so that we have the equivalence

$$\mathbf{N} \models \gamma_T \iff T \not\vdash \gamma_T, \tag{2-11}$$

and its contrapositive

$$\mathbf{N} \not\models \gamma_T \iff T \vdash \gamma_T. \tag{2-12}$$

We now infer that $T \not\vdash \gamma_T$: because if $T \vdash \gamma_T$, then $\mathbf{N} \models \gamma_T$ by soundness, contradicting (2-12). Hence, by (2-11), $\mathbf{N} \models \gamma_T$, i.e., γ_T is true; and hence $T \not\vdash \neg\gamma_T$, since T is sound and does not prove any false sentences.

This completes the proof of (2-7). ⊣

Notice that carefully read, γ_T claims its own unprovability from T.

§3. Problems.

x3.1. Prove Theorem 1.1.

x3.2. Prove Lemma 1.2, that the concatenation function is arithmetical.

x3.3. Prove Lemma 1.6.

x3.4. Prove that the relation Formula(f) defined in Lemma 1.7 is arithmetical.

x3.5. Prove (2) of Lemma 2.6.