Remark. If $T \vdash \forall \vec{x}\, \exists! y\ \phi(\vec{x}, y)$ with $\phi \in \Sigma_1$, then we can expand the language of T with a new functional symbol $f(\vec{x})$ and add the axiom $\phi(\vec{x}, f(\vec{x}))$.

Since $\quad f(x) = y \leftrightarrow \phi(\vec{x}, y) \qquad$ and

$$f(x) \neq y \leftrightarrow \exists z (z \neq y \wedge \phi(\vec{x}, z))$$

we see that $f(x) = y$ is equivalent to both a Σ_1 formula $\phi(\vec{x}, y)$ *and* a Π_1 formula $\forall z\ (z \neq y \rightarrow \neg\phi(\vec{x}, z))$.

This is easily seen to imply that if ϕ^* is a Σ_1 formula on the language that includes f, then ϕ^* is also equivalent to a Σ_1 formula without f.

Definition 5. *If $\phi(x, \vec{y})$ is a formula then LNP_ϕ (Least Number Principle) is the formula $\forall \vec{y}\ (\exists x\ \phi(x, \vec{y}) \rightarrow \exists x\ (\phi(x, \vec{y}) \wedge \forall z < x\ \neg\phi(z, \vec{y}))$.*

That is, the Least Number Principle for ϕ is a formula that states that: for any $\vec{y}$ for which $\phi(x, \vec{y})$ can be satisfied, there is a least such x satisfying $\phi(x, \vec{y})$.

Theorem 8. $\mathrm{I}\Sigma_1 \vdash LNP_{\neg\phi}$ *for every* $\phi \in \Sigma_1$.

Proof. Consider the formula $\Psi(x, \vec{y}) \equiv \forall z < x\ \phi(z, \vec{y})$. Clearly $\Psi(0, \vec{y})$. Assume $LNP_{\neg\phi}$ fails. Then

$$\Psi(x, \vec{y}) \rightarrow \Psi(x + 1, \vec{y})$$

since otherwise $x + 1$ would be the least element such that $\neg\phi(x + 1, \vec{y})$ holds. Thus, by Σ_1 induction $\forall x\ \Psi(x, \vec{y})$ i.e. $\neg\exists x\ \neg\phi(x, \vec{y})$, which is a contradiction. □

Theorem 9. $\mathrm{I}\Sigma_1 \vdash LNP_\phi$ *for all* $\phi \in \Sigma_1$.

Proof. Assume $\exists x\ \ \phi(x, \vec{y})$. Pick an arbitrary $\hat{x}$ such that $\phi(\hat{x}, y)$. Consider $\Psi(x, \vec{y}) \equiv \neg\exists(z < \hat{x} \dot{-} x)\ \phi(z, \vec{y})$ where

$$x \dot{-} y = \begin{cases} z & \text{such that } y + z = x \text{ if } y \leq x \\ 0 & \text{if } y > x \end{cases}$$

Then this is a $\neg$-Σ_1 formula and thus it satisfies the LNP.

Clearly $\Psi(\hat{x}, \vec{y})$ holds and thus there exists the least element x_0 that satisfies $\Psi(x_0, \vec{y})$; i.e. $\neg\exists(z < \hat{x} - x_0)\ \phi(z, \vec{y})$; i.e.

$$\forall(z < \hat{x} - x_0)\ \neg\phi(z, \vec{y}) \text{ and } \phi(\hat{x} - x_0)$$

i.e. $\hat{x} - x_0$ is the least number satisfying ϕ. □

Gödel's β function

Theorem 10. *There exists a primitive recursive function $\beta(x, i)$ such that for some $\phi \in \Sigma_1$*

$$\begin{aligned} \mathrm{I}\Sigma_1 &\vdash \forall x, i\ \exists z\ \phi(x, i, z) \\ \mathbb{N} &\vDash \forall x, i\ \phi(x, i, \beta(x, i)) \quad \textit{and} \end{aligned}$$

$$\begin{aligned} \mathrm{I}\Sigma_1 \vdash \forall x, y\ \exists \hat{x}\ \forall i < \beta(x, 0)\ (&\beta(\hat{x}, i) = \beta(x, i) \wedge \\ &\beta(\hat{x}, \beta(x, 0)) = y \wedge \\ &\beta(\hat{x}, 0) = \beta(x, 0) + 1) \end{aligned}$$

The idea is that x encodes a sequence of elements of length $\beta(x, 0)$, and given any y, x can be extended to a code $\hat{x}$ of a sequence that has one extra element y.

$$\begin{aligned} &\beta(x, 0) = \text{length}(x) = \ell \\ &\beta(x, i + 1) = (x)_i \quad \text{for all } 0 \leq i < \ell \\ &x = \overline{\langle (x)_0, \ldots, (x)_{\ell-1} \rangle} \end{aligned}$$

Gödel's original definition of β was based on the Chinese remainder theorem: Given an arbitrary sequence $a_0, \ldots, a_n$ and a sequence of relatively prime numbers $b_0, \ldots, b_n$ there exists a such that $a \equiv a_i \pmod{b_i}$ for all i.

However, such a coding function, while primitive recursive, is not suitable for us because it is not P-time computable.

For that reason we will simply assume the existence of Gödel's β function, and later we will define a more efficient, polynomial time computable encoding of sequences.

Theorem 11 (Main theorem for β function)**.** *Let $\phi \in \Sigma_1$. Then*

$$\mathrm{I}\Sigma_1 \vdash (\forall x < a)\ \exists! y\ \phi(x, y) \rightarrow \exists w\ (\ell(w) = a \wedge \forall (x < a)\ \phi(x, \beta(w, x + 1))$$

Thus, any Σ_1-definable sequence, finite from "model's point of view" (i.e. bounded in the model) can be encoded using the β function.

Proof. From the previous theorem, using Σ_1 induction on a. □

Remark. The above theorem works for arbitrary (also non-standard) element $a \in \mathcal{M} \vDash \mathrm{I}\Sigma_1$. For "honest-to-god" finite sequences a much simpler encoding can be defined by iterating the following pairing function:

Theorem 12. *Let $p(x, y) = \frac{1}{2}(x + y)(x + y + 1) + x$. Then $\mathrm{I}\Sigma_1 \vdash$ "$p(x, y)$ is a bijection between $\mathcal{M} \times \mathcal{M}$ and $\mathcal{M}$". i.e.*

$$\begin{aligned} \mathrm{I}\Sigma_1 \vdash \forall z \exists x \exists y\ (z = p(x, y))\ &\wedge \forall x, y, \bar{x}, \bar{y} \\ &(p(x, y) = p(\bar{x}, \bar{y}) \rightarrow x = \bar{x} \wedge y = \bar{y}) \end{aligned}$$

Proof. $p(x, y)$ is the "Cantor snake" □

Our Goal

Theorem 13. *All primitive recursive functions are provably total in* $\mathrm{I}\Sigma_1$*. i.e., for every* $f \in \mathrm{PR}$ *there exists a* Σ_1 *formula* ϕ_f *such that*

$$\mathrm{I}\Sigma_1 \vdash \forall \vec{x}\ \exists! y\ \phi(\vec{x}, y) \text{ and } \mathbb{N} \vDash \forall \vec{x}\ \phi(\vec{x}, f(\vec{x}))$$

Proof. The proof proceeds by induction on the complexity of f. Assume that

$$\begin{aligned} f(0, \vec{y}) &= g(\vec{y}) \\ f(x+1, \vec{y}) &= h(x, \vec{y}, f(x, \vec{y})) \end{aligned}$$

and assume that we have shown

$$\begin{aligned} &\mathrm{I}\Sigma_1 \vdash \forall \vec{y}\ \exists! z\ \phi_g(\vec{y}, x) \\ &\mathrm{I}\Sigma_1 \vdash \forall \vec{y}, x, z\ \exists w\ \phi_h(x, \vec{y}, z, w), \text{ and} \\ &\mathbb{N} \vDash \forall \vec{y}\ \phi_g(\vec{y}, g(\vec{y})) \\ &\mathbb{N} \vDash \forall \vec{y}, x, z\ \phi_h(x, \vec{y}, z, h(x, \vec{y}, z)) \end{aligned}$$

Let $\Psi(x, \vec{y}, w) \equiv \exists c\ (\ell(c) = x + 1 \wedge \phi_g(\vec{y}, (c)_0) \wedge$
$\forall i < x\ \phi_h(i+1, \vec{y}, (c)_i, (c)_{i+1}) \wedge$
$(c)_x = w)$

Then, using the main property of β (i.e., extendibility of sequences) we can show by induction on x that $\mathrm{I}\Sigma_1 \vdash \forall \vec{y}\, \forall x\ \exists! w\ \Psi(x, \vec{y}, w)$ and by induction on $\mathbb{N}$ that $\mathbb{N} \vDash \forall y, x\ \Psi(x, \vec{y}, f(x, \vec{y}))$. □

We now turn to the more difficult part:

Theorem 14. *If* $\mathrm{I}\Sigma_1 \vdash \forall \vec{x}\ \exists! y\ \phi(\vec{x}, y)$ *then* $\mathbb{N} \vDash \forall \vec{x}\ \phi(\vec{x}, f(\vec{x}))$ *for a primitive recursive function* $f(\vec{x})$*.*

We first present a model theoretic proof.

We can extend the language of $\mathrm{I}\Sigma_1$ with symbols for all primitive recursive functions and denote this theory by $\mathrm{I}\Sigma_1^*$.

Lemma 6. *If* $\mathrm{I}\Sigma_1 \vdash \forall x\ \exists! y\ \phi(x, y)$ *then* $\mathrm{I}\Sigma_1^* \vdash \forall x\ \exists y < f(x)\ \phi(x, y)$ *for some primitive recursive function* $f(x)$*.*

Proof. First we note that *every recursive function* is representable in $\mathrm{I}\Sigma_1$ but might not be provably convergent. By this we mean that there exists a Σ_1 formula $\phi(\vec{x}, y)$ such that $\mathrm{I}\Sigma_1 \vdash \phi(\vec{h}, f(\vec{h}))$ whenever and only if $f(\vec{h})$ converges. (This is called "numeral-wise representable".) However, even if $f(\vec{h})$ is a total function (defined for all inputs $\vec{n}$, still it might happen that

$$\mathrm{I}\Sigma_1 \nvdash \forall \vec{x}\ \exists y!\ \phi(\vec{x}, y)$$

To see this, we note that using coding of sequences we can encode a run of a Turing Machine as $f(\underline{n}) = \underline{m} \leftrightarrow \exists c$ ("$(c)_0$ is the description of tape of length $\pm\ell(c)$" and $\forall i < \ell(c)$ "$(c)_{i+1}$ has been obtained through a correct transition from $(c)_i$" and "the content of the tape at $(c)_{\ell(c)-1}$ is $\underline{m}$").
Denoting the last formula by

$$\exists c\, Calc(c, x, y)(\leftrightarrow \text{"}f(x) = y \text{ via computation } c\text{"})$$

it is easy to see that if $f(\underline{n}) = \underline{m}$ then $\exists k$ such that $\mathbb{N} \vdash Calc(k, n, m)$ where k codes "the real computation" on input n with final value m.
However, there is no reason why

$$\mathrm{I}\Sigma_1 \vdash \forall x \exists y \exists c\, Calc(c, x, y)$$

For example we will see that $\mathrm{I}\Sigma_1 \nvdash$ "Ackermann function is total".
However, by encoding either the general TM or a derivation in equational calculus, we can come up with a Σ_1-formula $Calc_A$ such that:

$$\mathrm{I}\Sigma_1 \vdash \forall y \exists y \exists c\, Calc_A(c, 0, y, z) \qquad \text{(in fact, } z = y + 1)$$
$$\mathrm{I}\Sigma_1 \vdash \forall x[\exists z \exists c\, Calc_A(c, x + 1, 0, z) \leftrightarrow \exists \overline{z} \exists \overline{c}\, Calc_A(\overline{c}, x, 1, \overline{z})]$$
$$\mathrm{I}\Sigma_1 \vdash \forall x[\exists z \exists c\, Calc_A(c, x + 1, y + 1, z) \leftrightarrow \exists z_1, z_2, c_1, c_2\, [Calc_A(c_1, x + 1, y, z_1) \wedge Calc_A(c_2, x, z_1, z_2)] \wedge$$
$$\forall x, y, z, c, c_1, c_2, z_1, z_2\, [Calc_A(c, x + 1, y + 1, z) \wedge Calc_A(c_1, x, z_1, z_2) \wedge Calc_A(c_2, x + 1, y, z_1) \rightarrow z = z_2)$$

However, we cannot prove in $\mathrm{I}\Sigma_1$, $\forall x \forall y \exists c \exists z\ Calc_A(c, x, y, z)$, even though for all naturals m, n there exists c, k such that

$$\mathbb{N} \vDash\ Calc_A(c, \underline{n}, \underline{m}, \underline{k})$$

and thus

$$\mathrm{I}\Sigma_1 \vdash \exists z\, Calc_A(c, \underline{n}, \underline{m}, z)$$

□