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 $f(x) = y \iff \phi(\vec{x}, y)$ and $f(x) \neq y \iff \exists z(z \neq y \land \phi(\vec{x}, z))$

we see that $f(x) = y$ is equivalent to both a $\Sigma_1$ formula $\phi(\vec{x}, y)$ and a $\Pi_1$ formula $\forall z(z \neq y \rightarrow \neg \phi(\vec{x}, z))$.

This is easily seen to imply that if $\phi^*$ is a $\Sigma_1$ formula on the language that includes $f$, then $\phi^*$ is also equivalent to a $\Sigma_1$ formula without $f$.

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

That is, the Least Number Principle for $\phi$ 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. $\Sigma_1 \vdash \text{LNP}_{\neg \phi}$ 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 $\text{LNP}_{\neg \phi}$ 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 $\Sigma_1$ induction $\forall x \Psi(x, \vec{y})$ i.e. $\neg \exists x \neg \phi(x, \vec{y})$, which is a contradiction. \qed

Theorem 9. $\Sigma_1 \vdash \text{LNP}_\phi$ 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 $\Psi(x, \vec{y}) \equiv \exists z(z < \hat{x} - x) \phi(z, \vec{y})$ where

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

Then this is a $\neg\Sigma_1$ 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. $\neg \exists z(z < \hat{x} - x_0) \phi(z, \vec{y})$; 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 $\phi$. \qed
Gödel's $\beta$ function

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

\[
\begin{align*}
\Sigma_1 \vdash \forall x, i \exists z \phi(x, i, z) \\
\mathbb{N} \models \forall x, i \phi(x, i, \beta(x, i)) \quad \text{and} \\
\Sigma_1 \vdash \forall x, y \exists \hat{x} \forall i < \beta(x, 0) \left( \beta(\hat{x}, i) = \beta(x, i) \land \right. \\
\beta(\hat{x}, \beta(x, 0)) = y \land \\
(\beta(\hat{x}, 0) = \beta(x, 0) + 1)
\end{align*}
\]

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 $y$.

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

Gödel’s original definition of $\beta$ 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 $\beta$ function, and later we will define a more efficient, polynomial time computable encoding of sequences.

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

\[
\begin{align*}
\Sigma_1 \vdash (\forall x < a) \exists! y \phi(x, y) &\rightarrow \exists w \left( \ell(w) = a \land \forall (x < a) \phi(x, \beta(w, x + 1)) \right)
\end{align*}
\]

Thus, any $\Sigma_1$-definable sequence, finite from “model’s point of view” (i.e. bounded in the model) can be encoded using the $\beta$ function.

**Proof.** From the previous theorem, using $\Sigma_1$ induction on $a$. \hfill $\square$

**Remark.** The above theorem works for arbitrary (also non-standard) element $a \in \mathcal{M} \models \Sigma_1$. 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 $\Sigma_1 \vdash \text{“}p(x, y)\text{” is a bijection between } \mathcal{M} \times \mathcal{M}\text{”}$. i.e.

\[
\begin{align*}
\Sigma_1 \vdash \forall z \exists x \exists y \left( z = p(x, y) \right) \land \forall x, y, \bar{x}, \bar{y} \\
(p(x, y) = p(\bar{x}, \bar{y}) &\rightarrow x = \bar{x} \land y = \bar{y})
\end{align*}
\]

**Proof.** $p(x, y)$ is the “Cantor snake” \hfill $\square$
Our Goal

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

$$\mathbf{I}^*\Sigma_1 \vdash \forall \vec{x} \exists y \phi(\vec{x}, y) \text{ and } \mathbb{N} \models \forall \vec{x} \phi(\vec{x}, f(\vec{x}))$$

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

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

and assume that we have shown

$$\mathbf{I}^*\Sigma_1 \vdash \forall \vec{y} \exists z \phi_g(\vec{y}, x) \quad \mathbf{I}^*\Sigma_1 \vdash \forall \vec{y}, x, z \exists w \phi_h(x, \vec{y}, z, w)$$

and

$$\mathbb{N} \models \forall \vec{y} \phi(g(\vec{y})) \quad \mathbb{N} \models \forall \vec{y}, x, z \phi(h(x, \vec{y}, z, h(x, \vec{y}, z)))$$

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

Then, using the main property of $\beta$ (i.e., extendibility of sequences) we can show by induction on $x$ that $\mathbf{I}^*\Sigma_1 \vdash \forall \vec{y} \forall x \exists! w \Psi(x, \vec{y}, w)$ and by induction on $\mathbb{N}$ that $\mathbb{N} \models \forall y, x \Psi(x, \vec{y}, f(x, \vec{y}))$. \[
\]

We now turn to the more difficult part:

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

We first present a model theoretic proof.

We can extend the language of $\mathbf{I}^*\Sigma_1$ with symbols for all primitive recursive functions and denote this theory by $\mathbf{I}^*_1$.

**Lemma 6.** If $\mathbf{I}^*\Sigma_1 \vdash \forall x \exists y \phi(x, y)$ then $\mathbf{I}^*_1 \vdash \forall x \exists y < f(x) \phi(x, y)$ for some primitive recursive function $f(x)$.

**Proof.** First we note that every recursive function is representable in $\mathbf{I}^*_1$ but might not be provably convergent. By this we mean that there exists a $\Sigma_1$ formula $\phi(\vec{x}, y)$ such that $\mathbf{I}^*\Sigma_1 \vdash \phi(\vec{h}, f(\vec{h}))$ 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

$$\mathbf{I}^*\Sigma_1 \not\vdash \forall \vec{x} \exists y \phi(\vec{x}, y)$$
To see this, we note that using coding of sequences we can encode a run of a Turing Machine as $f(n) = m \iff \exists \mathcal{C} (\text{"(c)_0 is the description of tape of length } \pm \ell(c)\text{" and } \forall i < \ell(c) \text{ "(c)_{i+1} has been obtained through a correct transition from (c)," and } \text{"the content of the tape at (c)_{\ell(c)-1} is } m\text{"})$

Denoting the last formula by

$$\exists \mathcal{C} \text{Calc}(c, x, y)(\iff f(x) = y \text{ via computation } c)$$

it is easy to see that if $f(n) = m$ then $\exists k \text{ such that } \mathbb{N} \models \text{Calc}(k, n, m)$ where $k$ codes “the real computation” on input $n$ with final value $m$.

However, there is no reason why

$$\Sigma_1 \vdash \forall x \exists z \exists \mathcal{C} \text{Calc}(c, x, y)$$

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

$$\Sigma_1 \vdash \forall x \exists y \exists z \text{Calc}_A(c, x, y, z) \quad (\text{in fact, } z = y + 1)$$

$$\Sigma_1 \vdash \forall x [\exists z \exists \mathcal{C} \text{Calc}_A(c, x + 1, 0, z) \iff \exists x \exists \mathcal{C} \text{Calc}_A(c, x, x, z)]$$

$$\Sigma_1 \vdash \forall x [\exists z \exists \mathcal{C} \text{Calc}_A(c, x + 1, y + 1, z) \iff \exists x, y, z, \mathcal{C} \text{Calc}_A(c, x, y + 1, z) \land \text{Calc}_A(c, x + 1, y, z) \land \text{Calc}_A(c, x, y + 1, z) \land \exists x, y, z]$$

$$\forall x, y, z, c, c_1, c_2, z_1, z_2 [\text{Calc}_A(c, x + 1, y + 1, z) \land \text{Calc}_A(c, x, z_1, z_2) \land \text{Calc}_A(c_1, x, z_1, z_2) \land \text{Calc}_A(c_2, x + 1, y, z_1) \rightarrow z = z_2]$$

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

$$\mathbb{N} \models \text{Calc}_A(c, n, m, k)$$

and thus

$$\Sigma_1 \vdash \exists z \text{Calc}_A(c, n, m, z)$$

\[\square\]