Math Logic 2
Σ_1-definability, Recursiveness, and Incompleteness

Prologue

The following notes contain a reworking of Enderton's Chapter 3, §§3-5. My reason for creating them is that I have trouble keeping track of what properties Enderton is describing at any particular moment. So below I concentrate on relations and functions on the natural numbers definable by formulas in three different form, called Δ_0, Σ_1, and Π_1. These will be defined below. But an intuition is this:

Δ_0 formulas are formulas whose truth depends, obviously, upon only the arithmetic properties of a known finite number of natural numbers — and thus whose truth could be checked by a (fast enough) computer.

Σ_1 formulas are of the form $\exists x_1 \exists x_2 \ldots \exists x_k \phi(\vec{x}, \vec{y})$, where ϕ is Δ_0; thus the truth of Σ_1 formula can be verified by conducting a simple search for $\vec{x}$ satisfying the ϕ, and by shouting "aha!" when you've found such $\vec{x}$. (Don't worry about the fact that you'll keep on searching forever if you never find such an $\vec{x}$.)

Π_1 formula are of the form $\forall x_1 \forall x_2 \ldots \forall x_k \phi(\vec{x}, \vec{y})$; thus, they are equivalent to the negations of Σ_1 formulas.

Δ_1-definable relations are relations which have both Π_1 and Σ_1 definitions, i.e., relations R where both R and its complement have Σ_1 definitions. We can test for the truth of such a formula by timesharing between the search described above corresponding to the Σ_1 definition of R and the search corresponding to the Σ_1 definition of $\overline{R}$. Because any tuple $\vec{y}$ is in either R or $\overline{R}$, one of these two searches has to end sooner or later. When the first one ends, we'll have our answer of whether $R(\vec{y})$ holds.

Ye Notes

1. Standard *abbreviations* when we write formulas — in addition to using usual infix order instead of prefix order when it improves readability:

Abbreviation	Abbreviates
$x \neq y$	$\neg x{=}y$
$x \leq y$	$(x{=}y \vee x < y)$
$x > y$	$y < x$
$x \geq y$	$(x{=}y \vee y < x)$
$\forall x_{<t} \phi$	$\forall x(x < t \rightarrow \phi)$
$\forall x_{\leq t} \phi$	$\forall x(x \leq t \rightarrow \phi)$
$\exists x_{<t} \phi$	$\exists x(x < t \wedge \phi)$
$\exists x_{\leq t} \phi$	$\exists x(x \leq t \wedge \phi)$
$\underline{b}$ (for $b \in \mathfrak{N}$)	$SSS\cdots S0$ (b S's)

   The last four abbreviations (for any variable x, and for any term t *not containing* x) are called *bounded quantifiers*. Note that, when interpreted over the natural numbers, they quantify only over *finite* sets of elements: $\{0, 1, \ldots, t-1\}$ or $\{0, 1, \ldots, t\}$.

2. **Reorganization:** *First* look at definability in the structure $\mathfrak{N}_E = (\mathbb{N}, 0, S, <, +, \cdot, E)$; *then* look at an axiom system $\mathbb{A}_E$.

Definability in $\mathfrak{N}_E$

3. **Notational Conventions:**
   (a) All through this section we are talking about definability in $\mathfrak{N}_E$. Generally, rather than saying that $\mathfrak{N}_E \models \phi$, we shall merely say that ϕ.
   (b) To increase readability below, instead of using our formal symbol E for exponentiation I'll use the usual superscript notation.

4. **Classes of formulas and definable sets: Δ_0, Σ_1, and Δ_1:**

Defn: A Δ_0 *formula* is a formula (of the language of $\mathfrak{N}_E$) all of whose quantifiers are bounded.

A Σ_1 *formula* is a formula (of the language of $\mathfrak{N}_E$) of the form $\exists x_1 \exists x_2 \ldots \exists x_n \phi(\vec{x}, \vec{y})$ where ϕ is Δ_0.

A Π_1 *formula* is a formula (of the language of $\mathfrak{N}_E$) of the form $\forall x_1 \forall x_2 \ldots \forall x_n \phi(\vec{x}, \vec{y})$ where ϕ is Δ_0.

Defn. (The following 3 Definitions are <u>not</u> the standard definitions, although they are provably equivalent to the standard definitions:)

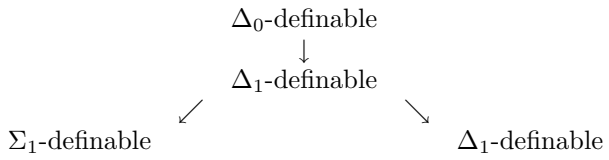
A relation R on $\mathbb{N}$ is *recursively enumerable* (*r.e.*) iff it is definable in $\mathfrak{N}_E$ by a Σ_1 formula. We also say, simply, that a recursive relation is Σ_1-definable.

A relation R on $\mathbb{N}$ is *co-r.e.* its complement is r.e. Equivalent, R is co-r.e. if it is Π_1 definable (in $\mathfrak{N}_E$).

A relation R on $\mathbb{N}$ is *recursive* if it is both r.e and co-r.e. Recursive relations are also called Δ_1-*definable relations*. But note that we do *not* define a term "Δ_1-formula".

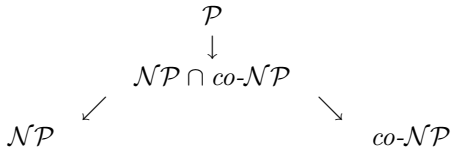
Comment upon intuition: ...

Inclusions: We represent below inclusions between the set of Δ_0-definable relations on $\mathbb{N}$, the set of recursive relations, the set of r.e.-relations, and the set of co-r.e. relations. We represent $A \subset B$ below by an arrow from A to B.



The $\subseteq$ relations above are all immediate consequences of the definitions. Inequality is more complicated, but you will see the tools to prove that the bottom 3 sets are all different in this class.

This hierarchy inspired the following hierarchy, which you have probably seen, which is the bottom end of the "polynomial time hierarchy."



But here all an arrow represents is the $\subseteq$ relation. We *assume* all the inclusions are proper, but nobody can prove it. (This is one of the major open problems in both Mathematics and Computer Science. If you have an elegant proof of the fact that they're all different, I'll be happy to be your Ph.D. advisor — at least if you promise to publish jointly!)

Defn. A *function* $f : \mathbb{N}^k \to \mathbb{N}$ is *recursive* if its graph — a set of $k+1$-tuples — is r.e.

5. **[Church's Thesis]** A function is computable by an algorithm iff it is recursive.

6. **Thrm:**

(a) Every r.e. k-ary relation R on $\mathbb{N}$ is definable in $\mathfrak{N}_E$ by a formula $\exists x \psi(x, \vec{v})$ where ϕ is Δ_0.
Proof: Since R is r.e., it is definable by a formula $\exists x_1 \exists x_2 \ldots \exists x_n \phi(\vec{x}, \vec{v})$ where ϕ is Δ_0. Hence it is also definable by

$$\exists x \exists x_{1<x} \exists x_{2<x} \ldots \exists x_{n<x} \phi(\vec{x}, \vec{v}),$$

which is of the desired form. (Note that this proof is valid, albeit a bit silly, even if $n = 0$.)

(b) The disjunction and conjunction of 2 Σ_1-definable relations are Σ_1-definable. **Proof:**

$$\begin{array}{lcl} \exists x \phi(x, \vec{y}) \vee \exists x \psi(x, \vec{y}) & \Leftrightarrow & \exists x(\phi(x, \vec{y}) \vee \psi(x, \vec{y})) \\ \exists x \phi(x, \vec{y}) \wedge \exists x \psi(x, \vec{y}) & \Leftrightarrow & \exists x(\exists x_{1<x} \phi(x_1, \vec{y}) \wedge \exists x_{2<x} \psi(x_2, \vec{y})) \end{array}$$

and the final formulas are all of the desired form.

(c) If $R(\vec{x}, y)$ is Σ_1-definable, so is the relation $S(\vec{x}) \;=_{def}\; \exists y R(\vec{x}, y)$.
Proof: This just adds one extra existential quantifier in front of the definition of R.

(d) If a relation $R(\vec{x}, v)$ is Σ_1-definable, so is the relation $S(\vec{x}, b) = \forall v_{<b} R(\vec{x}, v)$. (Remember that's vacuously true for $v = 0$.)
Proof: Suppose $R(\vec{x}, v) \;=_{def}\; \exists y \psi(\vec{x}, v, y)$, where ψ is Δ_0. So $\;S(\vec{x}, b) \;\Leftrightarrow\; \forall v_{<b} \exists y \psi(\vec{x}, v, y)$.
Now we claim: $\;\forall v_{<b} \exists y \psi(\vec{x}, v, y) \;\Leftrightarrow\; \exists z \forall v_{<b} \exists y_{<z} \psi(\vec{x}, v, y)$.
Proof of Equivalence:

$\Leftarrow$**:** Obvious.

$\Rightarrow$**:** Suppose that $\forall v_{<b} \exists y \psi(\vec{x}, v, y)$. For each such v, pick the least such y — call it y_v. Now let $z = \max(y_0, \ldots, y_{b-1})+1$. Then for this z $\forall v_{<b} \exists y_{<z} \psi(\vec{x}, v, y)$ is true.

7. **Thrm:**

(a) A function f is recursive iff its graph is recursive.
Proof:

$\Leftarrow$**:** trivial consequence of the definitions.

$\Rightarrow$**:** Suppose f is recursive, i.e., its graph is r.e. So its graph is defined by a Σ_1 formula

$$\psi(\vec{y}, z) \quad = \quad \exists x_1, \ldots, x_n \phi(\vec{x}, \vec{y}, z), \qquad \text{where } \phi \text{ is } \Delta_0.$$

We must show the complement of the graph of f is also Σ_1 definable. The important point for this is that, since f is a *function*, for each argument-sequence $\vec{y}$ there is a *unique* value for $f(\vec{y})$, a *unique* $z \in \mathbb{N}$ such that $\exists x_1, \ldots, x_n \phi(\vec{x}, \vec{y}, z)$. Thus $f(\vec{y}) \neq z$ iff $\exists z' \exists x_1, \ldots, x_n (\phi(\vec{x}, \vec{y}, z') \wedge z \not\approx z')$.

(b) A k-ary relation R is is recursive iff its *characteristic function* $\chi_R(\vec{x}) = \begin{cases} 1 & \text{if } R(\vec{x}) \\ 0 & \text{otherwise} \end{cases}$ is Σ_1-definable.

Proof: *Homework.*

8. **Thrm:** The following functions are Σ_1 definable:

(a) The successor function: by $v_2 = S v_1$.

(b) Any constant function $f(x_1, \ldots, x_m) = b$: by $v_{m+1} = \underline{b}$.

(c) The *projection functions* $I_i^m(x_1, \ldots, x_m) = x_i$: by $v_{m+1} = v_i$.

(d) Addition, multiplication, and exponentiation: by $v_3 = v_1 + v_2$, $v_3 = v_1 \cdot v_2$, and $v_3 = v_1 E v_2$.

(e) The composition of Σ_1 definable functions:
The general way to describe this is composition is to say that $h(\vec{x}) = g(f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x}), \vec{x})$. Note that this assumes that, for example, all the f_i's take the same parameters, which doesn't normally occur. The way to fix this up is to pad the f_i's with extra parameters — i.e., all of each other's parameters — and then use the projection functions to pick out only the functions we want. **Try working it out, or, if you must, see page 210; the trick is simple but clever.**
So assume $f_i(\vec{x}) = y_i \;=_{def}\; \exists \vec{v}_i \phi_i(\vec{x}, \vec{v}_i, y_i)$, and $g(\vec{y}, \vec{x}) = z \;=_{def}\; \exists \vec{w} \gamma(\vec{y}, \vec{x}, z)$.
Then $h(\vec{x}) = z$ is defined by $\exists \vec{y} (\exists \vec{v}_i \phi_i(\vec{x}, \vec{v}_i, y_i) \wedge g(\vec{y}, \vec{x}) = z)$, which, by Theorems 6b and 6c, is Σ_1-definable.

(f) *Functions* defined from Σ_1-definable functions by *minimization*: Suppose g is a Σ_1-definable function of $m+1$ arguments, called $a_1, \ldots, a_m, b$. *Furthermore,* suppose that, for some fixed integer k, it just turns out that, for each $a_1, \ldots, a_m \in \mathbb{N}$, there is at least one b where $g(a_1, \ldots, a_m, b) = k$.
Defn: The notation $\mu b[g(\vec{a}, b) = k]$ denotes the least natural number b such that $g(\vec{a}, b) = 0$. Actually, this is a *partial function*: it may be *undefined* for some arguments (if for all b's $g(\vec{a}, b) \neq 0$).
(**Note:** *A partial function may be recursive but its graph not be recursive. The previous result that if a (total) function is recursive then its graph is recursive depends critically upon the totality.*)

We assert $f(\vec{a}) = \mu b[g(\vec{a}, b) = k]$ is Σ_1-definable. For suppose $[g(\vec{a}, b) = y] \;=_{def}\; [\exists \vec{x} \gamma(\vec{a}, b, \vec{x}, y)]$. Then we have that:

$$\begin{aligned} f(\vec{a}) = z \quad &\Leftrightarrow \quad g(\vec{a}, b) = k \wedge \forall c < b g(\vec{a}, c) \not\approx k \\ &\Leftrightarrow \quad g(\vec{a}, b) = k \wedge \forall c < b \exists w (g(\vec{a}, c) = w \wedge x \not\approx k) \\ &\Leftrightarrow \quad \exists \vec{x} \gamma(\vec{a}, b, \vec{x}, k) \wedge \forall c < b \exists w (\gamma(\vec{a}, b, \vec{x}, y) \wedge w \not\approx k) \end{aligned}$$

By Theorems 6b and 6c, this last is equivalent to a Σ_1-formula.

This theorem is generally used with $k = 0$ or $k = 1$.

N.B.: *We must independently show that* for each $\vec{a}$, there is at least one b where $g(\vec{a}, b) = k$ before we can use this theorem. (There is a more general formulation, concerning what are called *partial recursive functions*, which doesn't require this assumption.)

As a corollary we get the following: Suppose $R(\vec{x}, y)$ is a recursive relation, and *furthermore* that, for each $\vec{x}$, there is at least one y such that $R(\vec{x}, y)$. Then $\mu y[R(\vec{x}, y)]$ is Σ_1-definable.

(g) Functions defined from recursive relations by *bounded minimization*: Suppose that $R(x, \vec{y})$ is a recursive relation. Define

$$f(z, \vec{y}) \ =_{def} \ \mu x < z[R(x, \vec{y})] \ =_{def} \ \begin{cases} \text{the least } x < z \text{ such that } R(x, \vec{y}) & \text{if such an x exists} \\ 0 & \text{otherwise.} \end{cases}$$

Then $\mu x < z[R(x, \vec{y})]$ is a recursive function of $\vec{y}$. (Notice that I, following Enderton, have actually used an easy form of Exercise 12f here.)

Proof: Recall that R and $\overline{R}$ can both be defined by Σ_1 formulas. Then we can get a Σ_1-definition of $\mu x < z[R(x, \vec{y})]$ by substituting those definitions into:

$$(x < z \ \wedge \ R(x, \vec{y}) \ \wedge \ \forall w < x \overline{R}(w, \vec{y})) \ \vee \ (x = 0 \ \wedge \ \forall x < z \overline{R}(x, \vec{y}))$$

and using Theorems 6b and 6d. Similarly

$$\mu x \leq z[R(x, \vec{y})] \ =_{def} \ \begin{cases} \text{the least } x \leq z \text{ such that } R(x, \vec{y}) & \text{if such an x exists} \\ 0 & \text{otherwise} \end{cases} \ = \mu x < z + 1 R(x, \vec{y})$$

is Σ_1-definable. Similarly

$$\max x < z[R(x, \vec{y})] \ =_{def} \ \begin{cases} \text{the greatest } x < z \text{ such that } R(x, \vec{y}) & \text{if such an x exists} \\ 0 & \text{otherwise.} \end{cases}$$

Proof: Modify the Σ_1 formula defining $\mu x < z[R(x, \vec{y})]$ above to read

$$(x < z \ \wedge \ R(x, \vec{y}) \ \wedge \ \forall w < z(w \leq x \ \rightarrow \ \overline{R}(w, \vec{y}))) \ \vee \ (x = 0 \ \wedge \ \forall x < z \overline{R}(x, \vec{y})).$$

Similarly

$$\max x \leq z[R(x, \vec{y})] \ =_{def} \ \begin{cases} \text{the greatest } x \leq z \text{ such that } R(x, \vec{y}) & \text{if such an x exists} \\ 0 & \text{otherwise.} \end{cases}$$

(h) **Study Hint:** I've started using earlier constructions in later constructions. Sooner or later, you're likely to lose sight of where you've been. I *strongly* recommend that as you go through this material you:

i. Go back every so often to make sure you can remember all the results. Build your own mental table of the results in a way you can keep track of. Check that you can reconstruct the proofs (formula constructions) without much review. Surprise your friends by misquoting the theorems to them and seeing whether they can find the errors.

ii. Every so often, when I give a proof using previous constructions, write the entire Σ_1 definition, tracing through the entire constructions.

9. **A Catalogue of recursive relations and functions**

(a) If R is a recursive binary relation and f, g are Σ_1-definable k-ary functions, then $S \ =_{def} \ \{\vec{x} : (f(\vec{x}), g(\vec{x})) \in R\}$ is recursive.

Proof: *Left to student.* **You need to see that this is just a fairly trivial application of the result on composition of Theorem 8e.**

(b) If $A, B \subseteq \mathbb{N}^n$ are recursive, so are $A \cap B$, $A \cup B$, and $\overline{A}$.

Proof: Since A, B are recursive, both A, B and $\overline{A}, \overline{B}$ are Σ_1-definable.

$A \cap B$: We must prove that $A \cap B$ and $\overline{A \cap B}$ are both Σ_1-definable. The first follows immediately from the assumptions by Theorem 6b. Since $\overline{A \cap B} = \overline{A} \cup \overline{B}$, the second does also.

$A \cup B$: Analogous.

$\overline{A}$: We must prove that $\overline{A}$ and $\overline{\overline{A}}$ are both Σ_1-definable. Since $\overline{\overline{A}} = A$, both parts are given in the hypotheses.

(c) If a relation $R \subseteq \mathbb{N}^n$ is recursive, so is the relation $S(\vec{x}, b) = \forall v < b R(\vec{x}, v)$.
Proof: Apply Theorem 6d to the Σ_1 definitions of R and $\overline{R}$ separately.

(d) The divisibility relation $D =_{def} \{(a, b) : a \text{ divides } b \text{ in } \mathfrak{N}_E\}$ is recursive — in fact, Δ_0-definable.
Proof: A Δ_0 definition is $\exists q \leq b(q \cdot a{=}b)$.
Notation: $a|b$ means that a is a divisor of b; $a \not| b$ means that a is not a divisor of b.

(e) The set of primes is recursive — in fact Δ_0-definable.
Proof: n is prime iff $n > 1 \wedge \forall x, y < n(x \cdot y \neq n)$.

10. **Primes and Sequence Coding:**

(a) The set of pairs of adjacent primes is recursive — in fact, Δ_0-definable.
Temporary Notation $AdjPr(m, n)$ means that m, n is an adjacent pair of primes.
Proof: m, n are adjacent primes iff

$$\text{prime}(m) \wedge \text{prime}(n) \wedge m < n \wedge \forall x < n(\text{prime}(x) \rightarrow x \leq m).$$

(b) The function mapping each natural number a to the $a+1$st prime, called p_a. (So $p_0 = 2$, $p_1 = 3$, etc.)
Proof: The underlying trick is being able to say that a natural number z is of the form $2^0 \cdot 3^1 \cdots p_a^a \cdot x$, with x not divisible by any of these primes, by a Δ_0 formula, which I temporarily call $PrSeq(a, b, z)$:

$$z > 0 \wedge 2 \not| z \wedge (\forall q_{<b} \forall r_{\leq b} AdjPr(q, r) \rightarrow \forall j_{\leq z}(q^j | z \leftrightarrow r^{j+1} | z)) \wedge b^a | z \wedge b^{a+1} \not| z).$$

Note that there always exists such b, z, and although there always exist infinitely many such z's, the b must be unique — in fact, the b must be p_a. Thus the following Σ_1 formula $athPrime(a, b)$ (another temporary name) defines the mapping from a to p_a:

$$\exists z(PrSeq(a, b, z)).$$

(c) **Definition:** We now *define* a method of coding finite sequences of natural numbers with natural numbers.
The code for the empty sequence ϵ — called $\langle\rangle$ — is 1. (This is a special case of what follows below — with $k = -1$ — under the understanding that the product of 0 numbers is 1.)
The code for a non-empty sequence $(a_0, a_1, \ldots, a_k)$ — called $\langle a_0, a_1, \ldots, a_k\rangle$ — is

$$2^{a_0+1} \cdot 3^{a_1+1} \cdots p_k^{a_k+1}.$$

Comment: There are many convenient ways to encode a finite sequence of natural numbers with a natural number. Another way is the following: Let σ_i $(0 \leq i \leq k)$ denote the base 2 representation of a_i, considered as a string of 0's and 1's. Now let the code for the sequence be the number with *base 3* representation $2\sigma_0 2\sigma_1 2\sigma_2 2 \ldots 2\sigma_k 2$.
The following theorem is expected of almost any sequence coding method:
Thrm: If $\langle a_0, a_1, \ldots, a_k\rangle = \langle b_0, b_1, \ldots, b_l\rangle$ then $k = l$ and $a_0 = b_0, a_1 = b_1, \ldots, a_k = b + k$.
Proof: This follows from the uniqueness of factorizations.
N.B.: Why are 0, 3, and 21 *not* codes for sequences?
The important point here is that, after we prove that, once we show that the encoding and decoding functions are recursive, we can refer to *finite sequences* in natural numbers when we actually quantify only over single natural numbers. (Gödel had to work a good deal harder to get this since he worked over $\mathfrak{N} = (\mathbb{N}, 0, S, <, +, \cdot)$ [see §3.7]; Enderton makes life easier for us by including exponentiation as a function in the structure.)

(d) **Thrm:** For each fixed $k \geq -1$, the function mapping $a_0, a_1, \ldots a_k$ to $\langle a_0, a_1, \ldots, a_k\rangle$ is Σ_1-definable.
Proof: In fact, it's trivially Δ_0-definable: $\phi_k(a_0, \ldots, a_k, b) =_{def} b = 2^{a_0+1} \cdot 3^{a_1+1} \cdots p_k^{a_k+1}$.

(e) **Defn:** Let $(a)_b = \max x < a[p_b^{x+1} | a]$. By Theorems 9d, 9a, 8g, and 10b, this function is Σ_1-definable.
Observe that, for $b \leq k$, $(\langle a_0, \ldots, a_b, \ldots, a_k\rangle)_b = (2^{a_0+1} \cdot 3^{a_1+1} \cdots p_k^{a_k+1})_b = a_b$, and for $b > k$, $(\langle a_0, \ldots, a_b, \ldots, a_k\rangle)_b = (2^{a_0+1} \cdot 3^{a_1+1} \cdots p_k^{a_k+1})_b = 0$, so $(a)_b$ is a decoding function.

(f) **Defn:** For any sequence code $\langle a_0, \ldots, a_m\rangle$, the *length* of $\langle a_0, \ldots, a_m\rangle$ is $m + 1$.
Thrm: There is a Σ_1-definable function lh such that, for any sequence code $a = \langle a_0, \ldots, a_m\rangle$, $\text{lh}(a) = m + 1$.
Proof: $\text{lh}(a) = \mu m(a = 0 \vee p_m \not| a)$ works.
Defn: Let lh be the above function.

(g) **Defn:** A natural number a is a *sequence number* (or *sequence code*) if there are natural numbers $m, a_0, \ldots, a_m$ such that $a = \langle a_0, \ldots, a_m \rangle$.
Thrm: The set of sequence numbers is recursive.
Proof: Exercise.
Defn: Let seq_no(a) be a Σ_1-formula asserting that a is a sequence number.

(h) **Defn:** For natural numbers $a = \langle a_0, a_1, \ldots, a_m \rangle$ and b,

$$a|\backslash b = \begin{cases} \langle a_0, a_1, \ldots, a_{b-1} \rangle & \text{if } b \leq m \\ a & \text{otherwise.} \end{cases}$$

Thrm: The function $a|\backslash b$ is a Σ_1-definable function of 2 arguments.
Proof: $a|\backslash b = \mu n[a = 0 \vee \forall j \leq b \forall k < a(p_j^k | a \rightarrow p_j^k | n)]$.

11. **Primitive Recursion:** Primitive recursion is a specific, limited, form of definition by recursion. Actually, Enderton's version is not quite the standard version — just store that in your bag of miscellaneous information for the next time you see the term "primitive recursion." Enderton's version is a sort of course-of-values primitive recursion. (See Exercise 12e.)

(a) **Defn:** For $f(a, b_1, \ldots, b_k)$ any function on the natural numbers (of at least 2 arguments), define

$$\tilde{f}(a, \vec{b}) = \langle f(0, \vec{b}), f(1, \vec{b}), \ldots, f(a-1, \vec{b}).$$

Observe that for any f, $\tilde{f}(0, \vec{b}) = 1$.
Thrm: f is Σ_1-definable iff $\tilde{f}$ is.
Proof: ($\Leftarrow$) $f(a, \vec{b}) = (\tilde{f}(S(a), \vec{b}))_a) = (\tilde{f}(a, \vec{b}))_{I_1^{k+1}(a, \vec{b})}$, the composition of 2 Σ_1 definable functions, which is Σ_1-definable by Theorem 8e.
($\Rightarrow$) Suppose f Σ_1-definable, say $f(a, \vec{b}) = c \Leftrightarrow \exists z \phi(a, \vec{b}, c)$. Then

$$\tilde{f}(a, \vec{b}) = c \Leftrightarrow \text{seq\_no}(c) \wedge \text{lh}(c) = a \wedge \forall i \leq a \phi(i, \vec{b}, (c)_i).$$

(b) **Defn:** Let g be a $k+2$-ary on $\mathbb{N}$. Define a $k+1$-ary function f as follows:

$$\begin{aligned} f(0, \vec{b}) &= g(\langle\rangle, 0, \vec{b}) \\ f(1, \vec{b}) &= g(\langle f(0) \rangle, 1, \vec{b}) \\ f(2, \vec{b}) &= g(\langle f(0), f(1) \rangle, 2, \vec{b}) \\ &\vdots \quad\quad \vdots \\ f(n, \vec{b}) &= g(\tilde{f}(n, \vec{b}), n, \vec{b}) \\ &\vdots \quad\quad \vdots \end{aligned}$$

N.B.: We can prove by induction on n that there is a unique value $f(n, \vec{b})$ meeting the above definition. Thus f is a well-defined function.
N.B.: Why is this a very specialized case of definition by recursion? We do allow defining $f(n, \vec{b})$ by using (recursively) any or all of the values $f(0, \vec{b}), f(1, \vec{b}), \ldots, f(n-1, \vec{b})$. First, we do not allow simultaneous recursion on two variables, as in $f(0,0) = 2; f(m, n) = f(m-1, n)^{f(m, n-1)}$ which does not match the above form. Second, we do not allow changing any of the other variables in the recursive call, as in $f(0, a) = 2^a; f(n, a) = 3 + f(n-1, 2a)$. We even require the recursion to be on the first argument of f, though this is really no restriction.

(c) **Thrm:** Suppose that a $k+1$ ary function f is defined from a $k+2$-ary function g by primitive recursion. If g is Σ_1-definable, so is f.
Proof: We first show that $\tilde{f}$ is Σ_1-definable. For

$$\tilde{f}(a, \vec{b}) = \mu s[seqno(s) \wedge \text{lh}(s) = a \wedge \forall i < a((s)_i = g(s|\backslash i, i, \vec{b}))].$$

Then $f(a, \vec{b}) = g(\tilde{f}(a, \vec{b}), a, \vec{b})$.
Self-test: Assuming that the Σ_1 formula $\exists \vec{x} \gamma(\vec{x}, a, \vec{b}, c)$ defines $f(a, \vec{b}) = c$, write out the Σ_1 formula defining f. This is a good check that you remember how all the above fits together.

N.B. Far more general theorems about definitions by recursion can be proved. For example, the definitions $f(0,a) = 2^a$; $f(n,a) = 3+f(n-1,2a)$ do in fact define a Σ_1-definable function, even though more than primitive recursion is used. The most famous result here is Kleene's Second Recursion Theorem; some of you may have seen this in the λ-calculus Y-combinator (as implemented, e.g., in Common LISP or Scheme). (Actually, the Y-combinator was discovered a bit earlier than Kleene's recursion theorem, I think by either Church or Kleene.)

**** Historical Note **** *The most famous (outside Russia) of the early formalisms to define computability were (1) Turing machines [Alan Turing], (2) lambda calculus [Alonzo Church, Stephen Kleene, J.B. Rosser], and (3) recursive function schemata [Stephen Kleene et. al., based upon earlier work of K. Gödel, J. Herbrand, and others]. By the third definition, a function is* recursive *if it is the smallest class of functions which (i) contain the successor, constant, and projection functions and (ii) are closed under composition, primitive recursion, and minimization. Thus we have just shown that every recursive function, by the original definition, is Σ_1-definable over $\mathfrak{N}_E$. The reverse can of course also be proved, though I don't think we'll do it. The equivalence of the three definitions above was proved, I believe, by Church and Kleene.*

(d) Suppose $f(a,\vec{b})$ is a Σ_1-definable function. Then the following functions are also Σ_1-definable:

- The product $\prod_{i<a} f(i,\vec{b})$
- The sum $\sum_{i<a} f(i,\vec{b})$.

These are easily defined by recursion. Hence the following are also defined by recursion:

$$P(a,\vec{b}) = \prod_{i<a} P(i,\vec{b}) \text{ and } S(a,\vec{b}) = \sum_{i<a} S(i,\vec{b}).$$

(e) **Defn:** For sequence numbers $a = \langle a_0, a_1, \ldots, a_m\rangle$ and $b = \langle b_0, \ldots, b_n\rangle$,

$$a * b = \langle a_0, a_1, \ldots, a_m, b_0, \ldots, b_n\rangle.$$

Thrm: The function $*$ is Σ_1 definable.
Proof:

$$a * b = \mu s[\mathrm{lh}(s) = \mathrm{lh}(a) + \mathrm{lh}(b) \ \wedge\ \forall i < \mathrm{lh}(a)((s)_i = (a)_i) \ \wedge\ \forall i < \mathrm{lh}(b)((s)_{i+\mathrm{lh}(a)} = (b)_i).$$

Just as with sums and products, we can define the concatenation as i runs from 0 to $a-1$ of the values of some function; see Enderton, page 215.

12. **HOMEWORK E:**

(a) Prove Theorem 7b.
(b) Prove Theorem 10g.
(c) Prove the case of Theorem 11c for g a binary function, as in the second part of Definition 11b.
(d) Enderton, §3.3 (page 216) # 7
(e) [Ordinary definition by primitive recursion] Show that if g, h are Σ_1-definable functions on $\mathbb{N}$, so is f defined by $f(0,\vec{b}) = g(\vec{b})$; $f(a+1,\vec{b}) = h(f(a,\vec{b}), a, \vec{b})$.
(f) **Definition by Cases:** Show that if R is a recursive relation and g, h are Σ_1 definable functions, then so is

$$f(\vec{a}) = \begin{cases} g(\vec{a}) & \text{if } \vec{a} \in R \\ h(\vec{a}) & \text{otherwise.} \end{cases}$$

(g) i. Suppose R is a recursive set (unary relation) and that $f_1, \ldots, f_n$ are Σ_1-definable functions. (The f_i's may well have different numbers of argument places; say that f_i is k_i-ary.) The *closure* of R under the f_i's is the smallest set S such that (1) $R \subseteq S$ and (2) for each f_i and each $x_1, \ldots, x_{k_i} \in S$, $f_i(x_1, \ldots, x_{k_i}) \in S$. This set S can be constructed recursively as follows:

$$\begin{aligned} S_0 &= R \\ S_{i+1} &= S_i \cup \bigcup_{1\le i\le n} \{f_i(x_1, \ldots, x_{k_i}) : x_1, \ldots, x_{k_i} \in S_i \\ S &= \bigcup_{i\in\mathbb{N}} S_i \end{aligned}$$

Prove that S is Σ_1-definable (i.e., r.e.).

ii. Suppose that, in addition, the functions $f_1, \ldots, f_n$ are *increasing*; that is, whenever $f_i(x_1, \ldots, x_{k_i}) = y$, $y > x_1$, $y > x_2$, …, and $y > x_{k_i}$. Prove that the set S above is then recursive.

Arithmetrization of Syntax

13. One of Gödel's insights was to observe that, in number theory (an object being discussed in first order logic), we could find a way to talk about the formulas of first order logic themselves and, to an extent, their satisfaction or provability. In a sense, Gödel found a way partially to bridge the object-language/meta-language gap.

    To this end we start out by assigning integers to the various symbols of the language of $\mathfrak{N}_E$ (including the "logical symbols"). Any "reasonable" assignment will do; Enderton chooses the symbols in the table below. (I added $\perp$ since it's key in Huth & Ryan's natural deduction proof system.) Now recall that every sentence is equivalent to a sentence whose only propositional connectives are $\neg$ and $\rightarrow$ (and I also allow $\perp$) and whose only quantifiers are $\forall$; thus we may, as Enderton does, treat every other formula as an *abbreviation* for a formula built up under this restriction. That's why the other connectives and $\exists$ don't appear on the table.

Symbol	:	Number	Symbol	:	Number
$\forall$	:	0	(	:	1
0	:	2	)	:	3
S	:	4	$\neg$	:	5
$<$	:	6	$\rightarrow$	:	7
$+$	:	8	$=$	:	9
$\cdot$	:	10	,	:	11
E	:	12	$\perp$	:	13
v_n	:	$14+n$			

    Such numbering, in general, is called *Gödel numbering.* Enderton calls the function given above h, i.e., $h(\forall) = 0$.

    In the defining formulas below the symbol h doesn't actually occur. When I refer to, say, $h(\forall)$, I'm really referring to the number 0; when I refer to $h(v_{17})$, I'm really referring to the number 31. When I refer to "some variable", I'm really referring to "some number ≥ 14."

    Having Gödel numbered the individual symbols, we Gödel number the set of strings $s0 \dots s_n$ of such symbols in the by-now fairly obvious fashion; recall that we are using an actual formula, not its abbreviation. (Does any remember the full rules for parenthesization any more?)

    For a string $s_0 \dots s_n$ of language symbols – expressions – (e.g., a formula):

$$\#(s_0 \dots s_n) = \langle h(s_0), h(s_1), \dots, (s_n)\rangle = 2^{h(s_0)+1} \cdot 3^{h(s_1)+1} \cdots p_n^{h(s_n)+1}.$$

    (Note that the Gödel number of a formula is a natural number.)

    For a set Φ of strings: $\#(\Phi) = \{\#(\phi) : \phi \in \Phi\}$. (Note that the Gödel number of a set of formulas is a set of natural numbers.)

    For a sequence $\phi_0, \dots, \phi_k$ of expressions:

$$\mathcal{G}(\alpha_0, \dots, \alpha_m) = \langle \#(\alpha_0), \dots, \#(\alpha_m)\rangle = 2^{\#(\alpha_0)+1} \cdot 3^{\#(\alpha_1)+1} \cdots p_m^{\#(\alpha_m)+1}.$$

    For a natural deduction proof: We don't need to store the justifications for each step; we can infer them — we just exhaustively check the current step against each legal combination of previous steps to see whether the current step is justified from those by a single application of a single proof rule. I won't go through that here — but you should *sketch* such a construction in your mind.

    We do, however, need to store nested subproof structure — sort of. To do this, just as we did in the soundness proof for natural deduction proofs, we shall consider a proof as a sequence of *sequents* $\chi_0, \chi_1, \dots, \chi_k \vdash \psi$. We may assume that, when we add hypotheses, they always get added onto the left. So, for example, $\rightarrow$-introduction is expressed as

$$\frac{\chi_0, \chi_1, \dots, \chi_k \vdash \psi}{\chi_1, \dots, \chi_k \vdash (\chi_0 \rightarrow \psi)}$$

    We shall represent such a sequent as $\mathcal{G}(\chi_0, \chi_1, \dots, \chi_k, \psi)$.

    Now that we have the additional hypotheses explicitly represented, we no longer need to represent the fact that subproofs are *consecutive* sequences of steps. (The soundness proof showed we didn't need to enforce that.) All that we need

to require is that the proof be a finite sequence of sequents, and that each sequent follows from 0 or more preceding sequents by the deduction rules.

Note that the $\forall$-introduction rules is strange in this format. For $\forall$-introduction we created a subproof with *no* additional assumptions but with a *new variable* "flagged." In this context that will translate as no additional assumption at all — just a check to make sure that the variable is "new." And "new" means just that it doesn't appear in any of the "hypotheses" of the sequent.

In order to keep the description of a proof structure relatively straightforward, we assume that only the copy rule is allowed to access a statement *outside* the current subproof — *i.e.*, a statement whose hypotheses do not exactly match the hypothesis where we are working.

Now the Gödel number for a proof P consisting of sequents $s_1, \ldots, s_k$ is $\mathcal{G}(P) = 2^{s_1+1} \cdot 3^{s_2+1} \cdots$ as usual.

14. **Thrm:** The following sets of Gödel numbers are recursive:

    (a) **The set Var of Gödel numbers of variables x:** by $x > S^{13}0$.

    (b) **The set of Gödel numbers of terms:** This is a corollary of Exercise 12g. The set of numbers of constant symbols is $\{2\}$, and every finite set is recursive. Thus $\{2\} \cup Var$ is recursive. (Which result am I quoting?)

    Now the set of terms is the closure of $\{2\} \cup Var$ under the functions creating larger terms from smaller ones:

$$\begin{aligned} f_S(\#(t)) &= h(S) * \#(t) \text{ (Recall that } * \text{ is used here for concatenation.)} \\ f_+(\#(t_1), \#(t_2)) &= h(+) * \#(t_1) * \#(t_2) \\ f_\cdot(\#(t_1), \#(t_2)) &= h(\cdot) * \#(t_1) * \#(t_2) \\ f_E(\#(t_1), \#(t_2)) &= h(E) * \#(t_1) * \#(t_2) \end{aligned}$$

    The first concatenates a fixed constant, $h(S) = 4$, onto the front of its input string, so it is Σ_1-definable by Theorem 11e. The others are similarly seen to be recursive. Moreover, it is trivial to check that all three are increasing: for example, $f_S(\#(t)) \geq h(S) \cdot \#(t)$, and since $\#(t) > 0$ (0 was not a sequence code), it follows that f_S is increasing.

    (c) **The set of Gödel numbers of atomic formulas:** Again, we'll use the fact that the concatenation operator is increasing — and if $a * b = a$ or $b * a = a$ then b must be the empty sequence. So

$$x \in AtomForm \;\Leftrightarrow\; \exists y, z < x (y \in Term \wedge z \in Term \wedge (x = h(\approx) * y * z \vee x = h(<) * y * z)).$$

    (d) **The set of Gödel numbers of formulas:** Similar to Theorem 14b.

15. **Thrm:** The following functions on Gödel numbers are Σ_1-definable.

    (a) **The function $f(n) = \#(S^n 0)$.**

$$\begin{aligned} f(0) &= \langle h(0) \rangle \\ f(n+1) &= \langle h(S) \rangle * f(n) \end{aligned}$$

    which is Σ_1-definable by Exercise 12e.

    (b) **The parsing functions which,** given the Gödel number of a term St_1, $+t_1t_2$, $\cdot t_1t_2$, or Et_1t_2 or atomic formula $\approx t_1t_2$ or $< t_1t_2$, return (i) the Gödel number of the first operation, S, $+$, $\cdot$, E, $\approx$, or $<$, (ii) the arity of that operator, (iii) the first operand (t_1 above), and (iv), if the operator is binary, the second operand.

    The parsing functions which, given the Gödel number of a formula, determine whether it is atomic, an implication $\phi \leftrightarrow \psi$, a negation $\neg\phi$, or a universal quantification $\forall x\phi$, and return ϕ and also ψ or x if appropriate.

    Proof: Homework.

    (c) **The substitution function Sb where** for $\alpha(x)$ a term or formula, t a term substitutable for x in α, and $\alpha(t)$ the result of the substitution, $Sb(\#(\alpha(x)), \#(x), \#(t)) = \#(\alpha(t))$.

    This function is defined by primitive recursion, broken into cases using the functions above. The fact that it is in fact an instance of primitive recursion depends upon the form of the definition — note that it does match the required form for primitive recursion — plus the fact that if a is a proper substring of b, then $\#(a) < \#(b)$.

$$Sb(\#(\approx t_1t_2),\#(x),\#(t)) = \begin{cases} \#(\approx tt) = \langle h(\approx)\rangle * \#(t) * \#(t) & \text{if } x = t_1 = t_2 \\ \#(\approx tt_2) = \langle h(\approx)\rangle * \#(t) * \#(t_2) & \text{if } x = t_1 \neq t_2 \\ \#(\approx t_1t) = \langle h(\approx)\rangle * \#(t_1) * \#(t) & \text{if } x = t_2 \neq t_1 \\ \#(\approx t_1t_2) = \langle h(\approx)\rangle * \#(t_1) * \#(t_2) & \text{otherwise} \end{cases}$$

$$Sb(\#(\approx t_1t_2),\#(x),\#(t)) = \begin{cases} \#(< tt) = \langle h(<)\rangle * \#(t) * \#(t) & \text{if } x = t_1 = t_2 \\ \#(< tt_2) = \langle h(<)\rangle * \#(t) * \#(t_2) & \text{if } x = t_1 \neq t_2 \\ \#(< t_1t) = \langle h(<)\rangle * \#(t_1) * \#(t) & \text{if } x = t_2 \neq t_1 \\ \#(< t_1t_2) = \langle h(<)\rangle * \#(t_1) * \#(t_2) & \text{otherwise} \end{cases}$$

$$Sb(\#(\neg\phi),\#(x),\#(t)) = \langle h(()), h(\neg)\rangle * Sb(\#(\phi),\#(x),\#(t)) * \langle h())\rangle$$

$$Sb(\#(\phi\to\psi),\#(x),\#(t)) = \langle h(()\rangle * Sb(\#(\phi),\#(x),\#(t)) * \langle h(\to)\rangle * Sb(\#(\psi),\#(x),\#(t)) * \langle h())\rangle$$

$$Sb(\#(\forall y\phi),\#(x),\#(t)) = \begin{cases} \#(\forall y\phi) & \text{if } x = y \\ \langle h(()), h(\forall), h(y), h())\rangle * Sb(\#(\phi),\#(x),\#(t)) & \text{otherwise.} \end{cases}$$

In the last clause above I used Nerode and Shore's parenthesization. Writing the actual Σ_1 formula is left for the student!

(d) **The function tail** which, given a sequence code $\langle a_0, a_1, \ldots, a_k\rangle$, returns its tail (a.k.a. CDR), that is, $\langle a_1, \ldots, a_k\rangle$: Left for students.

16. **Thrm:** The following relations are recursive:

(a) **The relation Fr such that $\langle\#(\alpha),\#(x)\rangle \in Fr$ iff x occurs free in α:** $Sb(a,b,\#(0)) \neq a$ defines Fr; $Sb(a,b,\#(0)) \neq a$ defines $\overline{Fr}$.

(b) **The set of Gödel numbers of sentences:** $\#(a)$ is a sentence iff, for all $\#(b) < \#(a)$, b doesn't not occur free in a.

(c) **The set of (*syntactically correct*) sequents** s as described above. This just checks that s is a sequence code, $\text{length}(s) \geq 1$, $s_{\text{length}(s)}$ is a formula, and for each $i < \text{length}(s)$, s_i is either a formula or 3^j for some j in Var.

(d) **For each proof rule, the set of all valid applications of that proof rule.**

We have do to this proof rule by proof rule — I'll go through a couple of examples and leave the rest for you. (But remember that we aren't using $\exists$, $\wedge$, and $\vee$ here, so we don't have to worry about those rules.)

- *Using an assumption*: $g = \mathcal{G}(\alpha_0,\alpha_1,\alpha_k)$ is a legal sequent if α_k is one of the hypotheses — one of the α_i's for $i < k$. That is, if there exists an $i < length(g)$ such that $(g)_i = (g)_{\text{length}(g)}$.
- *Copy rule*: $\mathcal{G}(\alpha_0,\alpha_1,\alpha_k)$ follows from $\mathcal{G}(\beta_0,\beta_1,\ldots,\beta_j)$ if both are sequents and, for some sequence code $t < \mathcal{G}(\beta_0,\beta_1,\ldots,\beta_j)$,
$$\mathcal{G}(\beta_0,\beta_1,\ldots,\beta_j) = \mathcal{G}(\alpha_0,\alpha_1,\alpha_k) * t.$$
- $\to$*-introduction*: $g_c = \mathcal{G}(\chi_1,\ldots,\chi_k,\chi_k\to\psi)$ follows from $g_h = \mathcal{G}(\chi_0,\chi_1,\ldots,\chi_k,\psi)$ if both are sequents and, for some $t_1,t_2,t_3 < \mathcal{G}(\chi_0,\chi_1,\ldots,\chi_k,\psi)$, $g_h = 2t_1 * t_2 * 2^{t_3}$ and $g_c = t_2 * t_1 * h(\to) * t_3$.
- $\to$*-elimination*: $g_c = \mathcal{G}(\chi_1,\ldots,\chi_k,\chi_k\to\psi)$ follows from from $g_{h_1} = \mathcal{G}(\chi_0,\chi_1,\ldots,\chi_k,\psi_1)$ and $g_{h_2} = \mathcal{G}(\chi_0,\chi_1,\ldots,\chi_k,$ if all are sequents, all have the same length, for all $i <$ that length, $(g_c)_i = (g_{h_1})_i(g_{h_2})_i$, and $(g_{h_1})_i = (g_{h_2})_i * h(\to) * (g_c)_i$.
- *Remaining cases*: Exercises. Note that you will have to worry about variable substitutions and variable occurrences for $\forall$-introduction and -elimination, so those are rather more complicated.

(e) *The set of Gödel numbers p of valid natural deduction proofs is recursive (where we assume all the assumptions/hypotheses used are explicitly included in each sequent, although the sequent may have other hypotheses that are never used).*

We need merely check that p is a sequence code, each position in p is a sequent, and each sequent follows from 0 or more preceding sequents by one of the inference rules. All those quantifications are bounded by the $length(p)$.

17. **Thrm:** For any finite (or recursive) set S of premises:

(a) $\#(Cn(S))$ is Σ_1-definable.

(b) if $Cn(S)$ is complete, then $\#(Cn(S))$ is recursive.

Proof: Exercise.

18. **Homework Exercise F:**

(a) Prove Theorem 15b.

(b) Prove Theorem 17.

(c) Above we showed that the sets of correct inferences using the copy rule and →-introduction and elimination are recursive. Prove the same for the remaining rules.

Axiom System $\mathbb{A}_E$

19. A *Small Modification of* Enderton's Axioms Set $\mathbb{A}_E$:

Axiom Set $\mathbb{A}_E$:			
[S1]	$\forall x(Sx \not\approx 0)$.	**[S2]**	$\forall x, y(Sx{=}Sy \rightarrow x{=}y)$.
[S3]	$\forall x(x \not\approx 0 \rightarrow \exists y x{=}Sy)$.		
[L1]	$\forall x, y(x < Sy \leftrightarrow x \leq y)$.	**[L2]**	$\forall x \neg(x < 0)$.
[L3]	$\forall x \neg(x < x)$.	**[L4]**	$\forall x, y(x < y \vee x{=}y \vee x > y)$.
[L5]	$\forall x, y, z(x < y \wedge y < z \rightarrow x < z)$.		
[A1]	$\forall x(x + 0{=}x)$.	**[A2]**	$\forall x, y(x + Sy{=}S(x + y))$.
[M1]	$\forall x(x \cdot 0{=}0)$.	**[M2]**	$\forall x, y(x \cdot S(y){=}x \cdot y + x)$.
[E1]	$\forall x(xE0{=}S0)$.	**[E2]**	$\forall x, y(xESy{=}xEy \cdot x)$.

Simple Lemma on *my variant of* $\mathbb{A}_E$. My $\mathbb{A}_E \supseteq \mathbb{A}_L$, and thus my $\mathbb{A}_E \models \mathbb{A}_S$.

$$\begin{aligned}\mathbb{PA}_E \quad = \quad & \mathbb{A}_E \cup \\ & \{ \quad \forall \vec{x}(\exists y \phi(\vec{x}, y) \rightarrow (\exists y(\phi(\vec{x}, y) \wedge \forall z(\phi(\vec{x}, z)) \rightarrow y \leq z))) : \\ & \qquad \phi(\vec{x}, y) \text{ is a (1-or-more-variable) formula of the language of } \mathfrak{N}_E \\ & \}\end{aligned}$$

Observe: The additional axioms in $\mathbb{PA}_E$ are just induction axioms for all 1st-order definable relations.

Historical Note: $\mathbb{PA}_E$ is essentially the axiom system called "first order Peano Arithmetic." Following (I believe) Russell and Whitehead's *Principia Mathematica*, first order Peano Arithmetic was taken as a standard of what is provable in some sort of constructive number theory. In what follows, for technical reasons which will be clear later, we shall mostly use only the finite axiom set $\mathbb{A}_E$, not the full infinite (but recursive) axiom set of first order Peano Arithmetic.

Proposition: $\mathbb{PA}_E$ is a recursive set of axioms.
We won't prove this proposition: we shall just (i) point out that $\mathbb{A}_E$ itself, since it is finite, is recursive, (ii) the programmers among you should "relatively easily" be able to write a computer program to identify the set of induction axioms, and (iii) given what we've already done — *be able to write* Σ_1 and Π_1 definitions of the set of induction axioms, and (iii) remind you that the union of two recursive sets is recursive.

Theorem: $Cn(\mathbb{PA}_E)$ is Σ_1 definable.
Proof: Follows from the above proposition plus Theorem 17.

20. **Underlying Assumption:** $\mathfrak{N}_E = \langle \mathbb{N}; 0, S, <, +, \cdot, E \rangle \models \mathbb{A}_E$.

21. **Lemmas from Enderton:**

(a) $\mathbb{A}_E \models \forall x(x \not< 0)$.

(b) For any natural number k, $\mathbb{A}_E \models \forall x(x < S^{k+1}0 \leftrightarrow (x = S^0 0 \vee S = S^1 0 \vee \cdots \vee x = S^k 0)$.
Note 1: The proof is by induction on k *in the metalanguage*.
Note 2: Can we translate that to put the "for any natural number k" into the formula, rather than in the meta language?
No: The bigger k gets above, the longer the statement gets, so if we were to say it for *all* k, the statement would get infinitely long.

(c) For every *variable free* term t (in the language of $\mathbb{A}_E$), there is a unique natural number n such that

$$\mathbb{A}_E \models t = S^n 0.$$

(Proved by induction on the length of t.)

Note *that parts (21b,21c), and the Simple Lemma on* my variant of $\mathbb{A}_E$*, together give us a picture of an arbitrary model* $\mathfrak{N}$ *of* $\mathbb{A}_E$*: the natural numbers plus a linearly-ordered group of* $\mathbb{Z}$*-chains on the top. Moreover, if there are any* $\mathbb{Z}$*-chains at all, these* $\mathbb{Z}$*-chains are densely linearly ordered without endpoints: if* $x < y$ *are in 2 different* $\mathbb{Z}$*-chains, then* $x/2$ *is in a strictly smaller* $\mathbb{Z}$*-chain,* $(x+y)/2$ *is in a* $\mathbb{Z}$*-chain strictly between them and* $2y$ *is in a* $\mathbb{Z}$*-chain strictly larger than both. (Of course, this doesn't tell us much about the* $+$ *and* $\cdot$ *relations on those chains — except, of course, that they satisfy our recursive axioms.)*

Defn: The elements in the ω-chain of any $\mathfrak{N} \models \mathbb{A}_E$ are called the *standard part* of $\mathfrak{N}$. The remaining elements are called the *nonstandard part*.

(d) For any *quantifier-free* sentence τ true in $\mathfrak{N}_E$, $\mathbb{A}_E \models \tau$. **(How would you prove this?)**

(e) For any sentence of the form $\psi = \exists x_1 \exists x_2 \cdots \exists x_k \phi$ of the language of $\mathfrak{N}_E$, where ϕ is quantifier free,

$$\text{if } \mathfrak{N}_E \models \phi \text{ then } \mathbb{A}_E \models \phi.$$

22. **Representability:**

Defn: (Generalizing property (21d) above): **Recall:**

Let $R \subseteq \mathbb{N}^m$. A formula $\rho(v_1, \ldots, v_m)$ (where only $v_1, \ldots, v_m$ occur free) *defines* R in $\mathfrak{N}_E$ if, for all $a_1, \ldots, a_m$ in $\mathbb{N}$,

$$\begin{array}{lcl} \langle a_1, \ldots, a_m \rangle \in R & \rightarrow & \mathfrak{N}_E \models \rho([a_1], \ldots, [a_m]), \text{ and} \\ \langle a_1, \ldots, a_m \rangle \notin R & \rightarrow & \mathfrak{N}_E \models \neg\rho([a_1], \ldots, [a_m]), \text{ and} \end{array}$$

We strengthen that by requiring, not only that $\rho([a_1], \ldots, [a_m])$ be *true* in $\mathfrak{N}_E$, but that in be *provable* in $\mathbb{A}_E$:

Again, let $R \subseteq \mathbb{N}^m$. A formula $\rho(v_1, \ldots, v_m)$ *represents* R *in* $\mathbb{A}_E$ if, for all $a_1, \ldots, a_m$ in $\mathbb{N}$,

$$\begin{array}{lcl} \langle a_1, \ldots, a_m \rangle \in R & \rightarrow & \mathbb{A}_E \models \rho([a_1], \ldots, [a_m]), \text{ and} \\ \langle a_1, \ldots, a_m \rangle \notin R & \rightarrow & \mathbb{A}_E \models \neg\rho([a_1], \ldots, [a_m]), \text{ and} \end{array}$$

A relation R is *representable* if there is a formula ρ representing it.
(Analogous definition for any other theory in the language of $\mathbb{A}_E$.)

Corollary to property (21d) above: If a relation R on $\mathbb{N}$ is definable by a quantifier-free formula, it is representable by the same formula.

But: There are some relation on $\mathbb{N}$ which, as we shall see later, are representable but not definable.

23. **Thrm:** For any $\mathfrak{A} \models \mathbb{A}_E$ and any $i, j, k \in \mathbb{N}$:

- $\mathfrak{A} \models \underline{i} < \underline{j} \;\Leftrightarrow\; i < j$. (**Proof:** This is similar to a related theorem for A_S.)
- $\mathfrak{A} \models \underline{i} + \underline{j} = \underline{k} \;\Leftrightarrow\; i + j = k$. (**By induction on** j**, prove** the result for all i, k. *Do it; it's easy!*)
- $\mathfrak{A} \models \underline{i} \cdot \underline{j} = \underline{k} \;\Leftrightarrow\; i \cdot j = k$. (**Proof:** Analogous.)
- $\mathfrak{A} \models \underline{i} E \underline{j} = \underline{k} \;\Leftrightarrow\; iEj = k$. (**Proof:** Analogous.)
- If $\mathfrak{N} \models \mathbb{A}_E$, the elements of the standard part of $\mathfrak{N}$ are just $\underline{0}^{\mathfrak{N}}, \underline{1}^{\mathfrak{N}}, \underline{2}^{\mathfrak{N}} \ldots$.
(**Proof:** This is just a weak form of part (21-21b))

Note: When context makes meaning clear, we usually write $\underline{n}$ to mean $\underline{n}^{\mathfrak{N}}$.

24. **Defn:** Let $\mathfrak{A} = (A, {c_1}^{\mathfrak{A}}, {c_2}^{\mathfrak{A}}, \ldots, {R_1}^{\mathfrak{A}}, {R_2}^{\mathfrak{A}}, \ldots, {f_1}^{\mathfrak{A}}, {f_2}^{\mathfrak{A}}, \ldots)$ be any structure and $B \subseteq A$ be any subset of A which (1) contains the interpretations ${c_i}^{\mathfrak{A}}$ of each constant of $\mathfrak{A}$ and (2) *closed under* the functions of $\mathfrak{A}$, i.e., for each ${f_i}^{\mathfrak{A}}$ — say f_i is k-ary — and for each $b_1, \ldots, b_k \in B$, ${f_i}^{\mathfrak{A}}(b_1, \ldots, b_k) \in B$.

    Then the *(induced) substructure* $\mathfrak{B}$ of $\mathfrak{A}$ is the subset B together with the interpretations of all the symbols of the language *restricted* to $\mathfrak{A}$. That is,

$$\mathfrak{B} = (B, {c_1}^{\mathfrak{A}}, {c_2}^{\mathfrak{A}}, \ldots, {R_1}^{\mathfrak{A}} \cap A^{j_1}, {R_2}^{\mathfrak{A}} \cap A^{j+2}, \ldots, {f_1}^{\mathfrak{A}} \cap A^{k_1+1}, {f_2}^{\mathfrak{A}} \cap A^{k_2+2}, \ldots),$$

    where each relation symbol R_i is j_i-ary and each function symbol f_i is k_i-ary (and thus ${f_i}^{\mathfrak{A}}$ is a set of ordered k_i+1-tuples).

25. **Thrm:** Let $\mathfrak{A} \models \mathbb{A}_E$ and let St be its standard part. Then:

    - St is closed under $S^{\mathfrak{A}}$, $+^{\mathfrak{A}}$, $\cdot^{\mathfrak{A}}$, and $E^{\mathfrak{A}}$,
    - the substructure $\mathcal{S}t$ of $\mathfrak{A}$ induced by St is isomorphic to $\mathfrak{N}_E$,
    - each ground term of the language evaluates to a (unique) element of St.
    - Every quantifier-free sentence τ is true in $\mathfrak{A}$ iff it is true in $\mathfrak{N}_E$, and for every $\exists_1$ formula $\phi(v_1, \ldots, v_k)$ and for every $n_1, \ldots, n_k \in \mathbb{N}$, $\mathfrak{N}_E \models \phi(n_1, \ldots, n_k) \;\Rightarrow\; \mathfrak{A} \models \phi(\underline{n_1}, \ldots, \underline{n_k})$.

    Proof: These are corollaries of the previous theorem, the Homomorphism Theorem, and Exercise 2.2.18.

26. Theorem 23 proves that the substructure $\mathcal{S}t = (St, \ldots)$ generated by the standard part of a model $\mathfrak{A}$ of $\mathbb{A}_E$ is more than just a substructure of $\mathfrak{A}$; it is what is called an *initial substructure*: if $a \in St$ and $b <^{\mathfrak{A}} a$ then $b \in St$. This property gives us a stronger version of results as in the homomorphism theorem:

    We often *identify* the standard part of $\mathfrak{N} \models \mathbb{A}_E$ with $\mathfrak{N}_E$ — thinking of $\mathfrak{N}$ as consisting of $\mathfrak{N}_E$ plus some extra elements "on the top".

27. **Theorem:**

    (a) (Modification of Homomorphism Theorem) Suppose $\phi(n_1, \ldots, n_k)$ is a Δ_0 formula and $n_1, \ldots, n_k \in \mathbb{N}$. Then, for every $\mathfrak{A} \models \mathbb{A}_E$,

$$\mathfrak{N}_E \models \phi(n_1, \ldots, n_k) \;\Leftrightarrow\; \mathfrak{A} \models \phi(\underline{n_1}, \ldots, \underline{n_k}).$$

    In Enderton's terminology (see page 197), every Δ_0 formula is *numeralwise determined*.

    Lemma 1: For any $\mathfrak{A} \models \mathbb{A}_E$ and any function s from the variable symbols into the universe A of $\mathfrak{A}$,

$$\mathfrak{A} \models \forall x_{<t} \phi(x, \vec{y}) \;\Leftrightarrow\; \text{for all } a \in A \text{ where } a <^{\mathfrak{A}} \overline{s}(t), \mathfrak{A} \models \phi(a, \vec{y}).$$

    Proof of Lemma 1: Recall that $\forall x < t \phi(x, \vec{y})$ abbreviates $\forall(x < t \rightarrow \phi(x, \vec{y}))$. So:

$$\begin{aligned}
\mathfrak{A} \models \forall(x < t \rightarrow \phi(x, \vec{y}))[s] \quad &\Leftrightarrow \quad \text{for all } a \in A \mathfrak{A} \models (x < t \rightarrow \phi(x, \vec{y}))[s(x|a)] \\
&\Leftrightarrow \quad \text{for all } a \in A, \mathfrak{A} \not\models x < t \text{ or } \mathfrak{A} \models \phi(x, \vec{y}))[s(x|a)] \\
&\Leftrightarrow \quad \text{for all } a \in A, (a, \overline{s}(t)) \not<^{\mathfrak{A}} \text{ or } \mathfrak{A} \models \phi(x, \vec{y}))[s(x|a)] \\
&\Leftrightarrow \quad \text{for all } a \in A \text{ where } (a, \overline{s}(t)) <^{\mathfrak{A}}, \mathfrak{A} \models \phi(x, \vec{y}))[s(x|a)]
\end{aligned}$$

    Lemma 2: Suppose $\phi(n_1, \ldots, n_k)$ is a Δ_0 formula and $n_1, \ldots, n_k \in \mathbb{N}$. Then, for every $\mathfrak{A} \models \mathbb{A}_E$, and for $\mathcal{S}t$ the standard part of $\mathfrak{A}$,

$$\mathcal{S}t \models \phi(n_1, \ldots, n_k) \;\Leftrightarrow\; \mathfrak{A} \models \phi(\underline{n_1}, \ldots, \underline{n_k}).$$

    Proof of Lemma 2: This proof is the much same as the proof of part (a) of the homomorphism theorem, page 91. That proof is by induction on quantifier-formulas. We need a proof by induction on Δ_0 formulas. The case atomic formulas is given in Enderton. The inductive cases for propositional connectives are routine. All we need to add is an inductive case for bound quantifiers — and just a case for bound universal quantifiers suffices since $\exists x < t \phi(x, \vec{y})$ is equivalent to $\neg \forall x < t \neg \phi(x, \vec{y})$. The only extra property of $\mathcal{S}t$ we need for that case is that $\mathcal{S}t$ is an initial substructure of $\mathfrak{A}$.

    Let $\phi(x, \vec{y})$ be Δ_0 and let s map the set of variables into $\mathcal{S}t$. The first and last equivalences below use the Lemma above; equivalence 1 uses the fact that $\mathcal{S}t$ is an initial substructure of $\mathfrak{A}$, so, for $a \in St$, $\{b \in A : b <^{\mathfrak{A}} a\} = \{b \in St : b <^{\mathcal{S}t} a\}$; and equivalence 1 follows by inductive hypothesis:

$$\begin{aligned}
\mathcal{S}t \models \forall x < t\phi(x,\vec{y})[s] \quad &\Leftrightarrow \quad \text{for all } a \in St \text{ where } a <^{\mathfrak{A}} \overline{s}(t)\mathcal{S}t \models \phi(x,\vec{y})[s(x|a)] \\
&\qquad \text{(by Lemma 1)} \\
&\Leftrightarrow \quad \text{for all } a \in A \text{ where } a <^{\mathfrak{A}} \overline{s}(t)\mathcal{S}t \models \phi(x,\vec{y})[s(x|a)] \\
&\qquad \text{(since } \mathcal{S}t \text{ is an initial substructure of } \mathfrak{A}) \\
&\Leftrightarrow \quad \text{for all } a \in A \text{ where } a <^{\mathfrak{A}} \overline{s}(t)\mathfrak{A} \models \phi(x,\vec{y})[s(x|a)] \\
&\qquad \text{(by inductive hypothesis)} \\
&\Leftrightarrow \quad \mathfrak{A} \models \forall x < t\phi(x,\vec{y}) \\
&\qquad \text{(by Lemma 1 again)}
\end{aligned}$$

Proof of Theorem: By Theorem 23, $\mathcal{S}t$ is isomorphic to $\mathfrak{N}_E$. Thus the result follows by the homomorphism theorem, parts (c-d) (that isomorphic structures satisfy the same sentences), plus Lemma 2.

(b) (Modification of Exercise 2.2.18) Suppose $\phi(n_1,\ldots,n_k)$ is a Σ_1 formula and $n_1,\ldots,n_k \in \mathbb{N}$. Then, for every $\mathfrak{A} \models \mathbb{A}_E$,

$$\mathfrak{N}_E \models \phi(n_1,\ldots,n_k) \;\Rightarrow\; \mathfrak{A} \models \phi(\underline{n_1},\ldots,\underline{n_k}).$$

Proof: Exercise.

28. **Defn:** A k-ary relation R on $\mathbb{N}$ is *representable in* $\mathbb{A}_E$ if there is a first order formula $\phi(v_1,\ldots,v_k)$ such that, for each $n1,\ldots,n_k \in \mathbb{N}$,

$$(n_1,\ldots,n_k) \in R \;\Leftrightarrow\; \mathbb{A}_E \vdash \phi(\underline{(}n_1),\ldots,\underline{(}n_k))$$

and

$$(n_1,\ldots,n_k) \notin R \;\Leftrightarrow\; \mathbb{A}_E \vdash \neg\phi(\underline{(}n_1),\ldots,\underline{(}n_k)).$$

Here we say that R is representable *by* ϕ.

29. **Thrm:** Every recursive relation $R \subseteq \mathbb{N}^k$ is representable in $\mathbb{A}_E$.

Proof: If R is recursive, both it and its complement have Σ_1 definitions in $\exists x\phi(x,\vec{y})$ as below:

$$\begin{aligned}
R(\vec{n}) \quad &\Leftrightarrow \quad \mathfrak{N}_E \models \exists x\phi(x,\underline{\vec{n}}) \\
\overline{R(\vec{n})} \quad &\Leftrightarrow \quad \mathfrak{N}_E \models \exists x'\psi(x',\underline{\vec{n}})
\end{aligned}$$

Then we have the following:

(a) For each choice of $\vec{n}$, exactly one of such an x and such an x' can exist in $\mathfrak{N}_E$.

(b) If $\exists x\phi(x,\underline{\vec{n}})$ (resp. $\exists x'\psi(x',\underline{\vec{n}})$) is true in N_E, it is true in every model $\mathfrak{A}$ of $\mathbb{A}_E$; moreover, the least x (resp. x') making the formula true in $\mathbb{A}_E$ is in the standard part of each such $\mathfrak{A}$. This follows from Theorem 27.

(c) If for some $\vec{y}$ both $\exists x\phi(x,\underline{\vec{n}})$ and $\exists x'\psi(x',\underline{\vec{n}})$ are true in some model $\mathfrak{A} \models \mathbb{A}_E$, then either every such x or every such x' must be in the non-standard part of $\mathfrak{A}$ — for if both some such x and some such x' were standard, by Theorem 27, $\mathfrak{N}_E \models\models \exists x\phi(x,\underline{\vec{n}})$ and also $\mathfrak{N}_E \models \exists x'\psi(x',\underline{\vec{n}})$, contradicting part 29a.
Thus the least such x or x' must be the "witness" for the formula which is true in $\mathfrak{N}_E$.

(d) Thus, in each $\mathfrak{A} \models \mathbb{A}_E$, for each $n_1,\ldots,n_k \in \mathbb{N}$,

$$R(\vec{n}) \;\Leftrightarrow\; \mathfrak{A} \models \exists x(\phi(x,\underline{\vec{n}}) \wedge \forall x' < x\neg\psi(x',\underline{\vec{n}})).$$

The analogous formula holds for $\overline{R}$.

Corollary: Every Σ_1-definable *function* on $\mathfrak{N}_E$ representable in $\mathbb{A}_E$. **Proof:** Recall that every Σ_1-definable function has a recursive graph.

Corollary: For every Σ_1-definable function $f(\vec{x}) = y$ there on $\mathfrak{N}_E$ there is a (possibly different) Σ_1 formula $\theta(\vec{x},y)$ where (i) θ represents f in $\mathbb{A}_E$ and (ii) for each tuple $\vec{n}$ of natural numbers, every $\mathfrak{A} \models \mathbb{A}_E$ contains a *unique* element m such that $\mathfrak{A} \models \theta(\underline{\vec{n}},m)$.

Proof: As above, θ should "say" that y is the *least such* y. This means there can be only one.

30. **Thrm: Every relation R on $\mathbb{N}$ which is representable in $\mathbb{A}_E$ is recursive.**

Proof Sketch: Since both $\mathrm{Cn}(\mathbb{A}_E)$ and $\{\alpha : \neg\alpha \in \mathrm{Cn}(\mathbb{A}_E)$ are Σ_1-definable, (by Theorem 17), and since for each $\vec{x}$ either $R(\vec{x})$ or $\overline{R}(\vec{x})$, we have the Σ_1 definitions of R and $\overline{R}$.

Incompleteness

31. In this section we continue to assume that we are working in the language of $\mathfrak{N}_E$. Generalizations are possible.

32. **Fixed Point Theorem:** For any formula β in which only v_1 occurs free, there is a sentence σ such that

$$\mathbb{A}_E \models (\sigma \leftrightarrow \beta(S^{\#(\sigma)}0)).$$

Discussion, quoted from Enderton, page 227:

> *We can think of σ as* indirectly *saying, "β is true of me." Actually σ doesn't say anything of course, it's just a string of symbols. And even when translated into English according to the intended structure $\mathfrak{N}_E$, it then talks of numbers and their successors and products and so forth. It is only by virtue of our having associated numbers with expressions that we can think of σ as referring to a formula, in this case to σ itself.*

Proof: [slightly expanded from Enderton]
Let $\theta(v_1, v_2, v_3)$ be a formula representing, as in the Corollary to Theorem 29, the function

$$sb(v_1, v_2) = \begin{cases} \#(\alpha(S^{v_2}0)) & \text{if } v_1 = \#(\alpha) \text{ for some } \alpha \text{ with (at most) 1 free variable} \\ 0 & \text{otherwise.} \end{cases}$$

The first trick is to consider the formula

$$\forall v_3(\theta(v_1, v_1, v_3) \rightarrow \beta(v_3)). \tag{1}$$

Note that we have set the first two argument positions of θ to the same variable. This means essentially that the Gödel number v_1 of a formula is being substituted into that formula, a process called "self reference." Enderton points out that "*This formula has only v_1 free. It defines in $\mathfrak{N}_E$ a set to which $\#(\alpha)$ belongs iff $\#(\alpha(S^{\#(\alpha)}0))$ is in the set defined by β .*"

Let q be the Gödel number of formula 1. The second trick is to replace v_1 in formula 1 with S^q0; note that the result is a sentence

$$\sigma \quad =_{def} \quad \forall v_3(\theta(S^q0, S^q0, v_3) \rightarrow \beta(v_3)). \tag{2}$$

Notice: Replacing v_1 with q is more self reference. Sentence σ asserts (in $\mathfrak{N}_E$) that $\#(\sigma)$ is in the set defined by β.

Finally, for this σ we shall prove our original claim that

$$\mathbb{A}_E \models (\sigma \leftrightarrow \beta(S^{\#(\sigma)}0)). \tag{3}$$

Recall that, since q is in fact a natural number, and since we chose θ as in the Corollary to Theorem 29 in every model of $\mathbb{A}_E$ there is a unique element x which satisfies $\theta(S^q(0), S^q(0), x)$, the actual Gödel number of the substituted formula. How do we create that formula? By definition of θ:

$$\begin{array}{rl} \text{take the formula with Gödel number } q\text{:} & \forall v_3(\theta(v_1, v_2, v_3) \rightarrow \beta(v_3)) \\ \text{and substitute } S^q0 \text{ for } v_1\text{:} & \forall v_3(\theta(S^q0, S^q0, v_3) \rightarrow \beta(v_3)) \end{array}$$

That formula is σ itself — the substitution operation described by the formula is just the operation we made in defining σ itself! So the Gödel number of the resultant formula is $\#(\sigma)$. Thus the only element v_3 which can satisfy $\theta(S^q, S^q, 0)$ is $\#(\sigma)$ itself, or, more precisely,

$$\mathbb{A}_E \models \forall v_3(\theta(S^q0, S^q0, v_3) \leftrightarrow v_3{=}S^{\#(\sigma)}0. \tag{4}$$

Now we prove assertion 3:

$\Rightarrow$: Trivially, from the statement of σ,

$$\sigma \models \theta(S^q0, S^q0, S^{\#(\sigma)}0) \rightarrow \beta(S^{\#(\sigma)}0).$$

By statement 4, $\mathbb{A}_E \models \theta(S^q0, S^q0, S^{\#(\sigma)}0)$. Thus $\mathbb{A}_E \cup \{\sigma\} \models \beta(S^{\#(\sigma)}0)$.

$\Leftarrow$: Statement (4) implies that

$$\beta(S^{\#(\sigma)}0) \rightarrow [\forall v_3(\theta(S^q0, S^q0, v_3) \rightarrow \beta(v_3))].$$

The part in the square brackets is just σ.

33. **Tarski's Theorem on the Undefinability of Truth:** The set $\#(Th(\mathfrak{N}_E))$ is not definable in $\mathfrak{N}_E$.

Proof: Suppose it were definable, say by some formula $\beta(x)$. By the fixed point theorem, *applied to* $\neg\beta$, there is a sentence σ such that

$$\mathfrak{N}_E \models (\sigma \leftrightarrow \neg\beta(S^{\#(\sigma)}0)).$$

But then σ is true in $\mathfrak{N}_E$ iff $\mathfrak{N}_E \models \neg\beta(S^{\#(\sigma)}0)$, contradicting the assumption about β.

Historical Note: This is just a version of the liar's paradox. If β did in fact define the set of statements true in $\mathfrak{N}_E$, σ would "say" "I am false."

34. **Corollary:** $\#(Th(\mathfrak{N}_E))$ is not r.e.

Proof: Every r.e. set of (Gödel numbers of) sentences is Σ_1 definable over $\mathfrak{N}_E$, and Tarski's theorem says that the $\#(Th(\mathfrak{N}_E))$ is not definable over $\mathfrak{N}_E$ by *any* first order sentence.

35. **Corollary: Gödel's Incompleteness Theorem [1931]:** If $A \subseteq Th(\mathfrak{N}_E)$ and A is recursive, then $Cn(A)$ is not complete.

Proof: Since $A \subseteq Th(\mathfrak{N}_E)$, if A were complete, $Cn(A) = Th(\mathfrak{N}_E)$. On the other hand, by Theorem 17, since A is recursive, $\#(Cn(A))$ is Σ_1 definable on $\mathfrak{N}_E$. This would say that $\#(Th(\mathfrak{N}_E))$ would be first order definable over $\mathfrak{N}_E$, contradiction Tarski's Theorem.

Corollary: $Th(\mathfrak{N}_E) \neq Cn(\mathbb{PA}_E)$.

Historical Note: Tracing through the proofs of the fixed-point theorem and Tarski's undefinability theorem, we can mechanically construct a sentence which is true in $\mathfrak{N}_E$ but not provable from $\mathbb{PA}_E$ (given our underlying assumption that $\mathfrak{N}_E \models \mathbb{PA}_E$).

36. **Quotation from Enderton:**

We can extract more information from the proof of Gödel's theorem. Suppose we have a particular recursive $A \subseteq Th\mathfrak{N}_E$ in mind. Then by Theorem 34A [my Theorem 17] we can find a formula β which defines $\#Cn(A)$ in $\mathfrak{N}_E$. The sentence σ produced by the proof to Tarski's Theorem is (as we noted there) a true sentence not *in $Cn(A)$. This sentence asserts that $\#(\sigma)$ does not belong to the set defined by β, i.e., it indirectly says "I am not a theorem of A." Thus $A \nvdash \sigma$, and of course $A \nvdash \neg\sigma$ as well. This way of viewing the proof is closer to Gödel's original proof, which did not involve a detour through Tarski's theorem. For that matter, Gödel's statement of the theorem did not involve $Th(\mathfrak{N}_E)$; we have taken some liberties in the labeling of theorems.*

*Next we need a lemma which says (roughly) that one can add one new axiom (*and hence finitely many new axioms) to a recursive theory without losing the property of recursiveness.*

37. **Lemma:** If $\mathrm{Cn}(\Sigma)$ is recursive then $\mathrm{Cn}(\Sigma \cup \{\gamma\})$ is recursive for any sentence γ.

Proof: $\Sigma \cup \{\gamma\} \models \alpha \;\Leftrightarrow\; \Sigma \models \gamma \rightarrow \alpha$. Thus $\#(\alpha) \in \mathrm{Cn}(\Sigma \cup \{\gamma\})$ iff $\#(\gamma \rightarrow \alpha) \in \mathrm{Cn}(\Sigma)$. Since the function mapping each $\#\alpha$ to $\#(\gamma \rightarrow \alpha)$ is recursive (by Theorem concatenation), we are done by Theorem 9a.

38. **Thrm: [Strong undecidability of $\mathbf{Cn}(\mathbb{A}_E)$]:** Let T be a theory such that $T \cup \mathbb{A}_E$ is consistent. Then $\#(T)$ is not recursive.

Proof: Suppose it were. Since $\#(T)$ is recursive, and $\mathbb{A}_E$ is finite, $\#(\mathrm{Cn}(T \cup \mathbb{A}_E))$ is also recursive (by Lemma 37). So it's complement, $\#(\overline{\mathrm{Cn}(T \cup \mathbb{A}_E)})$ is also recursive and thus is represented in $\mathrm{Cn}(\mathbb{A}_E)$ by some formula β.

By the Fixed Point Theorem, there is a sentence σ such that

$$\mathbb{A}_E \vdash [\sigma \leftrightarrow \neg\beta(S^{\#(\sigma)}0)]. \tag{5}$$

Moreover, since β represents $\#(\overline{\mathrm{Cn}(T \cup \mathbb{A}_E)})$, we have that

$$\text{either} \qquad \mathbb{A}_E \vdash \neg\beta(S^{\#(\sigma)}0) \qquad \text{or} \qquad \mathbb{A}_E \vdash \beta(S^{\#(\sigma)}0).$$

From the above 2 lines we also derive that

$$\text{either } \mathbb{A}_E \vdash \sigma \text{ or } \mathbb{A}_E \vdash \neg\sigma.$$

Now we get a contradiction by breaking into cases, asking whether $\sigma \in \mathrm{Cn}(T \cup \mathbb{A}_E)$:

$$\begin{array}{lcll} \sigma \in \mathrm{Cn}(T \cup \mathbb{A}_E) & \Rightarrow & \#(\sigma) \in \#\mathrm{Cn}(T \cup \mathbb{A}_E) & \text{(trivially)} \\ & \Rightarrow & \mathbb{A}_E \vdash \beta(S^{\#(\sigma)}) & \text{(by assumption about } \beta) \\ & \Rightarrow & \mathbb{A}_E \vdash \neg\sigma & \text{(by equation (5))} \\ & \Rightarrow & \neg\sigma \in \mathrm{Cn}(T \cup \mathbb{A}_E). & \end{array}$$

That contradicts the assumed consistency of $T \cup \mathbb{A}_E$. We get the analogous contradiction in the case $\sigma \in \mathrm{Cn}(T \cup \mathbb{A}_E)$ by negating each step in the derivation just above:

$$\begin{array}{lcll} \sigma \notin \mathrm{Cn}(T \cup \mathbb{A}_E) & \Rightarrow & \#(\sigma) \notin \#\mathrm{Cn}(T \cup \mathbb{A}_E) & \text{(trivially)} \\ & \Rightarrow & \mathbb{A}_E \vdash \neg\beta(S^{\#(\sigma)}) & \text{(by assumption about } \beta\text{)} \\ & \Rightarrow & \mathbb{A}_E \vdash \sigma & \text{(by equation (5))} \\ & \Rightarrow & \sigma \in \mathrm{Cn}(T \cup \mathbb{A}_E), & \end{array}$$

where the last is a direct violation of the assumption on the first line.

39. **Church's Theorem: The set of logically valid sentences of first order logic with equality is not recursive.** (This holds true in any language which is rich enough — by our proof, any language at least as rich as the language of $\mathfrak{N}_E$: thus any language containing at least one constant symbol, at least 4 function symbols, of which 3 are at least binary (i.e., take at least two arguments), and one relation symbol which is at least binary.)

40. **Homework Exercise G:**

    (a) Prove Theorem 27b.

    (b) Enderton, Exercise 3.5.1.

    (c) Enderton, Exercise 3.5.4.