Math Logic, Week#VIII

Moon's Day

Structure $\mathfrak{N}_S$ and axiom set $\mathbb{A}_S$:

$\mathfrak{N}_S = (\mathbb{N}; 0, S)$, where S is the successor function.

$$\mathbb{A}_S = \left\{ \begin{array}{ll} S1. & \forall x(S(x) \neq 0) \\ S2. & \forall x \forall y(S(x) = S(y) \ \rightarrow \ x = y) \\ S3. & \forall y(y \neq 0 \ \rightarrow \ \exists x(y = S(x))) \\ S4.1. & \forall x(S(x) \neq x) \\ S4.2. & \forall x(S(S(x)) \neq x) \\ \vdots & \vdots \\ S4.n & \forall x(S^n(x) \neq x) \\ \vdots & \vdots \end{array} \right\}$$

Proposition: $\mathfrak{N}_S \models \mathbb{A}_S$ — thus $Cn(\mathbb{A}_S) \subseteq Th(\mathfrak{N}_S)$.

Elimination of Quantifiers:

1. *A theory — or set of sentences — T admits elimination of quantifiers iff for every formula φ* (of the language of T) *there is a quantifier-free formula ψ* (of the language of T) *such that*

$$T \models (\varphi \leftrightarrow \psi.$$

Vocabulary: We say that *φ is equivalent to psi in $Th(\mathfrak{N}_S)$.*

(And, of course, we are interested where we have an algorithm that, given φ, produces ψ.)

2. **Lemma:** *Suppose that, whenever each of* α_0, α_1, $\ldots$, α_n *is an atomic formula or a negated atomic formula, there is a quantifier-free formula* ψ *where*

$$T \models (\psi \leftrightarrow \exists x(\alpha_0 \wedge \alpha_1 \wedge \cdots \wedge \alpha_n).$$

Then T *admits elimination of quantifiers.*

Proof: By induction on formulas: [and just expanded from Enderton]
We note that every formula is logically equivalent to one whose only connectives are $\wedge$, $\vee$, and $\neg$, and whose only quantifier is $\exists$.

If φ **is atomic:** Trivial: $T \models (\varphi \leftrightarrow \varphi)$.

If φ **is** $(\neg\theta)$ where the result holds for θ**:** Also trivial. **(¿Why?)**

If φ **is** $(\theta \vee \chi)$ **or** $(\theta \wedge \chi)$ where the result holds for θ and χ**:**
Similarly trivial.

If φ is $\exists x\theta$ where the result holds for θ:

- By inductive hypothesis, θ is equivalent to a quantifier-free formula.

- And that formula can be put into DNF (disjunctive normal form) — say

  $$T \models \theta \leftrightarrow ((\alpha_1 \wedge \cdots \wedge \alpha_m) \vee (\beta_1 \wedge \cdots \wedge \beta_n) \vee \cdots \vee (\xi_1 \wedge \cdots \wedge \xi_t)).$$

- So, from T we can prove

  $$\exists x\theta \;\leftrightarrow\; \exists x((\alpha_1 \wedge \cdots \wedge \alpha_m) \vee \cdots \vee (\xi_1 \wedge \cdots \wedge \xi_t))$$

  and so also

  $$\exists x\theta \;\leftrightarrow\; \exists x(\alpha_1 \wedge \cdots \wedge \alpha_m) \vee \cdots \vee \exists x(\xi_1 \wedge \cdots \wedge \xi_t)$$

  (¿Why?)

- Now θ was, by assumption, quantifier-free, and the conversion to DNF doesn't add quantifiers.
  So, by definition of DNF (in the context of first-order formulas), each of $\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_n, \xi_1, \ldots \xi_t$ is either atomic or negated atomic.

- Accordingly, by assumption of the lemma, each of $\exists x(\alpha_1 \wedge \cdots \wedge \alpha_m), \ldots, \exists x(\xi_1 \wedge \cdots \wedge \xi_t)$ is equivalent to a quantifier-free formula.

  And $\exists x\theta$ is equivalent to the disjunction of those quantifier-free formulas, which is still quantifier-free.

3. **Theorem:** $Th(\mathfrak{N}_S)$ *admits elimination of quantifiers.*

Proof: Use the lemma. Consider a formula $\exists x(\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_m)$, where each α_i is atomic or negated atomic.

- Note that each α_i is of the form $t_1 = t_2$ or $(\neg t_1 = t_2)$, for t_1 and t_2 terms. **(¿Why?)**
- And each term is of one of 3 forms: $S^n(x)$, $S^n(0)$, and $S^n(v_i)$, for $v_i \neq x$ — for some natural number n for each term.

Enderton writes out the algorithm — so I'll just do some examples that show the ideas.

(a) θ is $v_j \neq S^{13}(v_i) \ \wedge \ S^3(v_i) = S^7(v_k) \ \wedge \ S^9(0) = S(v_j) \ \wedge \ S^3(0) \neq S^2(v_i)$

x does not appear in θ, so the $\exists x$ is redundant

— and $\exists x\theta$ is logically equivalent to θ.

(b) θ is $x \neq S^{13}(v_i) \ \wedge \ \mathbf{S^3(x) = S^7(0)} \ \wedge \ S^9(x) = S(v_j) \ \wedge \ S^3(0) \neq S^2(x)$

The formula *requires that* $S^3(x) = S^7(0)$, so x *must be* $S^4(0)$.

— so $\exists x\theta$ is equvalent in $Th(\mathfrak{N}_S)$ to

$S^4(0) \neq S^{13}(v_i) \ \wedge \ \mathbf{S^7(0) = S^7(0)} \ \wedge \ S^{13}(0) = S(v_j) \ \wedge \ S^3(0) \neq S^6(0)$

(just replacing each occurrence of x with $S^4(0)$ and dropping the now redundant quantifier).

(c) θ is $x \neq S^{13}(v_i) \ \wedge \ \mathbf{S^7(x) = S^3(0)} \ \wedge \ S^9(x) = S(x) \ \wedge \ S^3(0) \neq S^2(x)$

Note that $S^7(x) = S^3(0)$ is always false in $\mathfrak{N}_S$.

— so $\exists x\theta$ is equivalent (in $Th(\mathfrak{N}_S)$) to $0 \neq 0$.

(d) θ = $x \neq S^{13}(v_i) \ \wedge \ \mathbf{S^3(x) = S^7(v_k)} \ \wedge \ S^9(x) = S(v_j) \ \wedge \ S^3(0) \neq S^2(x)$

Same idea part (3b): replace x with $S^4(v_k)$ and drop the $\exists x$.

(e) θ = $x \neq S^{13}(v_i) \ \wedge \ \mathbf{S^7(x) = S^3(v_i)} \ \wedge \ S^9(x) = S(v_j) \ \wedge \ S^3(0) \neq S^2(x)$

This says $S^4(x) = v_i$, so $x = S^{-4}(v_i)$

— but we have no symbol in the language for S^{-1}.

Furthermore, we don't know of $S^{-4}(v_i)$ even exists.

But *if it does exist*: since S is 1-1 in $\mathfrak{N}_S$: in $\mathfrak{N}_S$,

$S^3(S^{-4}(v_i) = S^k(y)$ is equivalent to $S^3(v_i) = S^{k+4}(y)$.

So $\exists x\theta$ is equivalent in $Th(\mathfrak{N}_S)$ to

$$\mathbf{v_i \neq 0 \wedge v_i \neq S(0) \wedge v_i \neq S(S(0)) \wedge v_i \neq S(S(S(0)))}$$
$$\wedge$$
$$v_i \neq S^{17}(v_i)) \ \wedge \ \mathbf{S^7(v_i) = S^7(v_i)} \ \wedge \ S^9(v_i) = S^5(v_j) \ \wedge \ S^6(0) \neq S^2(v_i)$$

(f) θ is $\mathbf{x \neq S^{13}(v_i)} \ \wedge \ \mathbf{S^7(x) \neq S^3(v_i)} \ \wedge \ S^9(0) = S(v_j) \ \wedge \ S^3(v_j) \neq S^2(v_i)$

Here, all the references to x are negative

— so (for any fixed interpretation s), all those references say about x is that it is not one of finitely many elements

— and $|\mathfrak{N}_S|$ is infinite, so there always is such an x.

So $\exists x\theta$ is equivalent, in $Th(\mathfrak{N}_S)$, to

$$\mathbf{0 = 0} \ \wedge \ \mathbf{0 = 0} \ \wedge \ S^9(0) = S(v_j) \ \wedge \ S^3(v_j) \neq S^2(v_i).$$

4. Enderton notes that the above theorem can be expanded to another proof that $Cn(\mathbb{A}_S) = Th(\mathfrak{N}_S)$:
   (a) Adjust proof to show elimination of quantifiers in $Cn(\mathbb{A}_S)$,
   (b) Show that every *quantifer-free sentence* is provably true or provably false from $\mathbb{A}_S$,
   (c) And, again, use the fact that, since $\mathfrak{N}_S \models \mathbb{A}_S$, $Cn(\mathbb{A}_S) \subseteq Th(\mathfrak{N}_S)$.

Homework: §3.1 ## 1, 4

Next topic: Σ_1-definability in $\mathfrak{N}_E = (\mathbb{N}; 0, S, <, +, \cdot, E)$

Below restrict attention for formulas in the language of $\mathfrak{N}_E$ (and including =):

1. It's common to look at special syntactic classes of formulas and their properties — notably:
   - **Existensial Formulas:** $\exists x_1 \exists x_2 \ldots \exists x_i \varphi$, for φ iquantifier-free.
   - **Universal Formulas:** $\forall x_1 \forall x_2 \ldots \forall x_i \varphi$, for φ quantifier-free.

   Relations defined by those formulas often have interesting properites.

2. In $\mathfrak{N}_E$ we look at somewhat larger classes:

   Bounded Quantifiers are
   $\exists x_{<t}$ $\exists x_{<t} \varphi$ abbreviates $\exists x(x < t \wedge \varphi)$
   $\forall x_{<t}$: $\forall x_{<t} \varphi$ abbreviates $\forall x(x < t \to \varphi)$
   where, in both cases, variable x does not appear in term t.

   Δ_0**-formulas:** Formulas where all quantifiers are bounded.

   Observation: We can easily write a program that, given a Δ_0 formula φ, computes whether it is true in $\mathfrak{N}_E$.
   (And we could run it too, if the numbers don't get too big.)

   Σ_1 **formulas:** Formulas $\exists x \varphi$ where φ is Δ_0.

   The following definitions are equivalent to the standard definitions:

   Definition: A relation R on $\mathbb{N}$ is *recursively enumerable* (*r.e.*) if R is definable by a Σ_1 formula in $\mathfrak{N}_E$.

   A relation R on $\mathbb{N}$ is *recursive* if r and its complement are both recursively enumerable.

   A function $f : \mathbb{N}^k \to \mathbb{N}$ is *recursive* iff its graph — a $k+1$-ary relation on $\mathbb{N}$ — is *r.e.*

3. **Church-Turing Thesis:**

   A function is recursive iff it is, intuitively, algorithmically computable.

4. **Review Intuitions:**

   - **¿Why is a recursive function computable?**
   - **¿Why is a recursively enmerable set semi-decidable?**
   - **¿Why is a recursive set decidable?**

   Main intuition for believing we've characterized those 3 sets:

   - In particular, this agrees with the final forms of all other formalisms that have been accepted, such as
     (a) **Turing machines**
     (b) **Lambda calculus expressions**
     (c) **Kolmogorov machines**
     (d) **General recursive functons** (a different, but equivalent, definition of "recursive")
     (e) **Random access machines** (essentially, von Neuman architecture machines with unlimited memory)
   - Turing's image of a Turing machine — of a human computer working with an unlimited supply of pencils and erasers and two potentially infinite stacks of paper — was convincing.
   - Nobody has been able to come up with a satisfactory intuitive justfication for calling any non-recursive function computable, *etc.*
   - **Note:** quantum computation, as now conceived (at least as I understand it), can do no more than exponentially increase the speed of a computation — and it also has some severe limitations.

5. **Vocabulary you'll run into:**

- A relation on **N** is *co-r.e.* if its complement is r.e.
  (Note analogue: "co-NP" means the set's complement is NP.)
- A Π_1 formula is a formula $\forall x\varphi$ where φ is Δ_0.
  By de Morgan's laws for $\forall$ and $\exists$, we see that Π_1 formulas define co-r.e. sets.

6. **Theorems:**

(a) *Every Δ_0-definable relation R on $\mathbb{N}$ is recursive.*

Proof: If Δ_0-formula φ defines R, and v_i is a variable not occurring in φ,

$\exists v_i\varphi$ defines R, and
$\exists v_i(\neg\varphi)$ defines the complement of R.

(b) *Every recursive relation is r.e.* **Proof:** Trivial.

(c) *If formula ψ and θ is Σ_1,*
then so are $\psi \wedge \theta$, $\psi \vee \theta$, $\exists v_i\psi$, $\exists v_{i\leq t}\psi$, and $\forall v_{i\leq t}\psi$.

Proof: Say ψ is $\exists x\alpha$ and θ is $\exists y\beta$. Then, in $Th(\mathfrak{N}_E)$,

$$\begin{aligned} \psi \wedge \theta &\models\!=\!\models \exists z(\exists x_{\leq z}\alpha \wedge \exists y_{\leq z}\beta) \\ \psi \vee \theta &\models\!=\!\models \exists z(\alpha \vee \beta) \\ \exists v_i\psi &\models\!=\!\models \exists z(\exists v_{i\leq z}\exists x_{\leq z}\alpha) \end{aligned}$$

(where we assume all the new variables don't occur in ψ and θ.)
(**N.B.:** $\neg\psi$ need not be equivalent to a Σ_1 formula. More later.)

(d) *A function f is recursive iff its graph is recursive.*

Intuition of proof: $f(x) \neq y$ iff $\exists z(f(x) = z \wedge z \neq y)$.

(e) If $R_1, R_2 \subseteq \mathbb{N}^k$ are both recursive, so are
$R_1 \cup R_2$, $R_1 \cap R_2$, and the complements of R_1 and R_2.
Proof: **¡Figure it out!** (It's fairly trivial.)

(f) A k-ary relation R is recursive iff its *characteristic function* $\chi_R(x_1, \ldots, x_k) = \begin{cases} 1 & \text{if } (x_1, \ldots, x_k) \in R \\ 0 & \text{otherwise} \end{cases}$ is Σ_1-definable.

Proof: **Homework Problem D**

7. **The following functions are recursive** (*i.e.*, for us, Σ_1 definable)**:**

(a) Successor, addition, multiplication, and exponentiation (*e.g.* $f(x, y) = x \cdot y$ by $(v_3 = v_1 \cdot v_2)$)

(b) All constant functions, *e.g.*, $f(x) = 0$ and $g(x, y, z) = 1347298$.

(c) The projection functions $I^m_j(x_1, \ldots, x_m) = x_j$

(d) Compositions of recursive functions, *e.g.*

$$f(x, y, z) = x + S(y \cdot E(x + z)) + 23$$

— but read note on the subtlety on page 4 of my notes.

(e) Suppose R is a recursive $k + 1$ ary relation on $\mathbb{N}$, and suppose it happens to be true that, for each $x_1, \ldots, x_k$, there is at least one x_{k+1} for which $R(x_1, \ldots, x_k, x_{k+1})$ is true. The the function $(f(x_1, \ldots, x_{k+1}) = \mu x_{k+1} R(x_1, \ldots, x_k, x_{k_1})$ — where f returns the least $x_k + 1$ where $R(x_1, \ldots, x_k, x_{k_1})$ — is recursive.

And here, to save time, I'll start skimming. You should skim, up through the end of page 10 of my notes, just to get a general idea of what people do.

If you have studied Turing machines, as you go through this material, you can see the tools for proving that every function computable by a Turing machine is Σ_1-definable.

8. **Sequence coding:** Encode a sequence $(n_0, n_1, \ldots, n_k)$ with integers according to the integers' prime factorizations:

$$\text{seq-code}(n_0, n_1, \ldots, n_k) \;=\; 2^{n_0+1}3^{n_1+1}\cdots p_k^{n_k+1}$$

where p_i is the ith prime — starting with $p_0 = 2$.

The following functions are recursive:

- For each fixed k, $f(x_1, \ldots, f_k) \;=\; \text{seq-code}(n_0, n_1, \ldots, n_k)$.
- Determining whether an integer n is a sequence code.
- If n is a sequence code, determining the length of the sequence.
- If n is a sequence code of length $\geq k$, finding the k element of the sequence.
- For sequence codes m and n, $m * n$ is the code for the concatenation of the 2 sequences.

9. **Now find a way to encode formulas of the language of $\mathfrak{N}_E$ with integers:**

- Start by picking an almost arbitrary h code for representing the symbols:

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	$14 + 2n$	A_i	$15 + 2n$

(a) I chose to do 1st order logic and propositional logic together.

(b) We have to decode function h.

- Then encode formulas — sequences of the above symbols — as sequence codes for the (encoding of) the symbols — for a string of symbols $s_0 s_1 \ldots s_k$, Enderton calls the code

$$\#(s_0 s_1 \ldots s_k).$$

- Encode sets Γ of formulas: $\#(\Gamma) = \{\#(\varphi) : \varphi \in \Gamma\}$.
- To encode formal deductions (in Enderton's system), it suffices to encode only the sequence of statements; the order gives the statement numbers, and one can infer the reasons by merely considering all possibilities.

  Define $\mathcal{G}(\alpha_0, \alpha_1, \ldots, \alpha_k)$ to be the sequence code for

$$(\#(\alpha_0), \#(\alpha_1), \ldots, \#(\alpha_k)).$$

  $\mathcal{G}(P)$ is called the *Gödel number* of P.

  (Enderton is precise: I'll also refer to the earlier $\#$ and h as Gödel numbers.)

10. **Several Theorems** I'll gloss over: All the syntactic properties we defined about formulas and formal deductions are recursive.
11. So, for any finite — or recursive or recursively enumerable — set $\mathbb{A}$ of axioms,

$$\{\#(\varphi) : \varphi \in Cn(\mathbb{A})\} \text{ is r.e..}$$

Idea of proof: $\#\varphi \in \#Cn(\mathbb{A})$ iff]

$$\exists P(P \text{ is the Gödel number of a proof of } \varphi \text{ from } \mathbb{A})$$

and, after all the encoding, the part inside the parentheses above is recursive.

12. Axiom set $\mathbb{A}_E$ (slightly modified from Enderton's set)

Axiom Set $\mathbb{A}_E$:			
[S1]	$\forall x(Sx \neq 0)$.	**[S2]**	$\forall x, y(Sx{=}Sy{\rightarrow}x{=}y)$.
[S3]	$\forall x(x \neq 0{\rightarrow}\exists y x{=}Sy)$.		
[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]**	$\forall x, y(xESy{=}xEy \cdot x)$.

Easy Proposition: $\mathbb{A}_E \models \mathbb{A}_S$.

13. **And a version of Peano's axioms for arithmetic:** — $\mathbb{A}_E$ plus an induction axiom for every first-order definable relation.

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

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.

Plausibility argument: $\mathbb{A}_E$ is finite, and hence recursive. Think about how you'd write a program test whether a string is an induction axiom.

Theorem: $Cn(\mathbb{PA}_E)$ is Σ_1 definable.
Proof: Follows from the previous theorms.

Freya's Day

Aside 1: On class mechanics:

1. Recall that, when you put solutions on the board, I'm going to require you to present them by memory.

   (So, before each class, review the next 4-5 problems that are due for presentation to be sure you remember how to do them.)

   If there's *1 unrememberable step*, you may *ask* me for permission to look on your notes.
2. I also encourage you to go back frequently to review definitions and major concepts from earlier in the course.
3. Ask more questions in class or see me more in my office hours.

   And review each day's material before the next class — and don't let things go inadequately understood.

   (You should have a good notion of what I expect. "Sort of understanding" isn't good enough.)
4. If you did badly on the midterm, I'll do no more than lower grades for students whose grades are near borderlines: low A's I'll change to A–'s, low A–'s I'll change to B+'s, *etc.*
   - **BUT:** I'll ask a final on the same format.
   - And compute the *maximum* of your grade on the 2 tests.
   - I'll still compute a grade for you based upon your homeworks, number of problems claimed, and quality of presentations.
   - **BUT** I'll treat that maximum of the 2 test grades as (at least approximately) a ceiling on your grades.
   - With this in view, I'll assign letter grades on the midterm:
     Grade in 30's: A (A– or A)
     Grade in 20's: B (B+, B, or B–)
     (I'll probably be a little less generous on the final exam.)

Aside 2: Important computability issue we skip over:

partial recursive functions:

1. (The graph of) a *partial function* $f : \mathbb{N}^k \to \mathbb{N}$
   is a set $F \subseteq \mathbb{N}^{k+1}$ where
   for each $(n_1, \ldots, n_k) \in \mathbb{N}^k$
   there is *at most* one n_{k+1} where $(n_1, \ldots, n_{k+1}) \in F$.
2. So every function is also a partial function. But the converse is, pretty obviously false.
   We could say "*non-total partial function*" to emphasize that it's not a function.
3. *Partial recursive function*: a partial function whose graph is r.e.

   It's graph need not be recursive.

   (The trick we used to prove that the graph of a recursive function is recursive simply doesn't work here)

   Proof: follows from Turing's proof of undecidability of halting.
4. It can be argued that partial recursive functions, not (total) recursive functions, are the proper study of much of compueter science, *e.g.*, algorithms.

   Given a computer program, you can (generally) tell by inspection that it computes a partial recursive function. But there's no algorithm to check whether it computes a total function.

   You can define a partial recursive function freely, recursively, in terms of itself — but you can't guarantee it's going to be total.

   We often think of a partial function as one for which we haven't bothered to compute the rest of the values — to *extend* the function to a total function. But there are partial recursive functions which cannot be extended to total recursive functions!

Back to $\mathfrak{N}_E \;\; = \;\; (\mathbb{N}; 0, S, <, +, \cdot, E)$

1. Underlying Assumption: $\mathfrak{N}_E \models \mathbb{A}_E$. **So** $Cn(\mathbb{A}_E) \subseteq Th(\mathfrak{N}_E)$. But, as we'll show later, $Cn(\mathbb{A}_E) \neq Th(\mathfrak{N}_E)$.

2. **Representability:**

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

   $$\langle a_1, \ldots, a_m \rangle \in R \;\; \rightarrow \;\; \mathfrak{N}_E \models \rho([a_1], \ldots, [a_m]), \text{ and}$$
   $$\langle a_1, \ldots, a_m \rangle \notin R \;\; \rightarrow \;\; \mathfrak{N}_E \models \neg\rho([a_1], \ldots, [a_m]).$$

   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 $\mathbb{A}_E$:

   Formula $\rho(v_1, \ldots, v_m)$ *represents* R *in* $\mathbb{A}_E$ if, for all $a_1, \ldots, a_m$ in $\mathbb{N}$,

   $$\langle a_1, \ldots, a_m \rangle \in R \;\; \rightarrow \;\; \mathbb{A}_E \models \rho([a_1], \ldots, [a_m]), \text{ and}$$
   $$\langle a_1, \ldots, a_m \rangle \notin R \;\; \rightarrow \;\; \mathbb{A}_E \models \neg\rho([a_1], \ldots, [a_m]).$$

   A relation R is *representable in* $\mathbb{A}_E$ if there is a formula ρ representing it.

   For any other axiom set $\mathbb{A}$ in the same language, we define *representable in* $\mathbb{A}$ analogously.

   So: Every representable relation on $\mathfrak{A}$ is definable (by the same formula) —

   but we'll show not every definable relation is representable.

   Consequence of Homormorphism Theorem: Definability *bu a quantifier-free formula* implies representability (by the same formula). **But we'll get more.**

3. **Notation** (@ meta-theoretic level)**:** For $k \in \mathbb{N}$, $\underline{k}$ is the term $S^k(0)$
   So for $\mathfrak{N} \models \mathbb{A}_E$, $\underline{k}^{\mathfrak{N}}$ is the interpretation of the term $\underline{k}$ in $\mathfrak{N}$.
4. **Thrm:** For any $\mathfrak{A} \models \mathbb{A}_E$ and any $i, j, k \in \mathbb{N}$:
   - $\mathfrak{A} \models \underline{i} < \underline{j}$ *iff* $i < j$.
   - $\mathfrak{A} \models \underline{i} + \underline{j} = \underline{k}$ *iff* $i + j = k$.
   - $\mathfrak{A} \models \underline{i} \cdot \underline{j} = \underline{k}$ *iff* $i \cdot j = k$.
   - $\mathfrak{A} \models \underline{i} E \underline{j} = \underline{k}$ *iff* $iEj = k$.
5. **So, ¿what does a model of $\mathbb{A}_E$ look like?**

   & Recall: $\mathbb{A}_E \models \mathbb{A}_S$. So use what we know about models of $\mathbb{A}_S$.

   Let $\mathfrak{A} \models \mathbb{A}_E$, and call it's ω-chain $\mathcal{S}t$.

   By part (4) above,
   - $0^{\mathfrak{A}} \in \mathcal{S}t$.
   - If $x, y \in \mathcal{S}t$, $S(x)$, $x +^{\mathfrak{A}} y$, $x \cdot^{\mathfrak{A}} y$, and $x\, E^{\mathfrak{A}} y$ are all in $\mathcal{S}t$.
     Vocabulary: $\mathcal{S}t$ is *closed* under $S^{\mathfrak{A}}$, $+^{\mathfrak{A}}$, $\cdot^{\mathfrak{A}}$ and $E^{\mathfrak{A}}$.
   - So $S^{\mathfrak{A}} \cap (\mathcal{S}t \times \mathcal{S}t)$ is a (total) function on $\mathcal{S}t$
     and similarly for $+^{\mathfrak{A}} \cap (\mathcal{S}t^3)$, $\ldots \cdot^{\mathfrak{A}} \cap \mathcal{S}t^3$, and $E^{\mathfrak{A}} \cap (\mathcal{S}t^3)$.
   - So $(\mathcal{S}t; 0^{\mathfrak{A}}, <^{\mathfrak{A}} \cap \mathcal{S}t^2, +^{\mathfrak{A}} \cap (\mathcal{S}t^3), \ldots)$ is a structure for the language of $\mathfrak{N}_E$. **Call that structure $\mathfrak{S}t$.**
   - And it's isomorphic to $\mathfrak{N}_E$.

6. And, for $\mathfrak{A} \models \mathbb{A}_E$,
   - (By axiom $L1$) if $x \in \mathcal{S}t$, $y \in |\mathfrak{A}|$, and $x <^{\mathfrak{A}} y$, then $y \in \mathcal{S}t$.
   - **Vocabulary:** $\mathfrak{S}t$ is an *initial substructure* of $\mathfrak{A}$.