8 Arithmetical Definability

In this chapter, we'll show that all computable functions and relations on the natural numbers can be defined in the language $\mathfrak{L}_A$ of arithmetic. As foreshadowed at the end of Chapter 5, it will follow that there can be no true, computably axiomatizable, and complete theory of arithmetic. Further limitative consequences will be explored in the next two chapters.

8.1 Definability

In Section 4.1, I introduced the language $\mathfrak{L}_A$, with non-logical symbols for the number 0 ('0'), the successor function ('s'), addition ('+'), and multiplication ('×'). Other arithmetical concepts can be defined in terms of these primitives. Let's think about what this involves.

Back in Section 4.1, I suggested that we can define the less-than relation $<$ by stipulating that for any terms t_1 and t_2, '$t_1 < t_2$' is short for '$\exists z(t_1 + s(z) = t_2)$', where z is a variable not occurring in t_1 or t_2. This works because every $\mathfrak{L}_A$-term denotes a natural number (in the standard model $\mathfrak{N}$), and a number a is less than a number b iff there is a non-zero number c such that $a + c = b$.

An $\mathfrak{L}_A$-*numeral* is an $\mathfrak{L}_A$-term constructed from 0 and s alone. Every natural number is denoted by a unique $\mathfrak{L}_A$-numeral (in $\mathfrak{N}$): 0 is denoted by '0', 1 by '$s(0)$', 2 by '$s(s(0))$', and so on. To avoid clutter, I will use '$\overline{n}$' as a shorthand for the $\mathfrak{L}_A$-numeral of the number n. So $\overline{5}$ is '$s(s(s(s(s(0)))))$'. Since every $\mathfrak{L}_A$-term denotes (in $\mathfrak{N}$) a number n that is also denoted by an $\mathfrak{L}_A$-numeral $\overline{n}$, it suffices to focus on $\mathfrak{L}_A$-numerals when examining whether an $\mathfrak{L}_A$-expression defines a relation on the natural numbers.

Officially, we'll say that an n-ary relation is defined by an $\mathfrak{L}_A$-formula with n free variables. The less-than relation $<$, for example, is defined by the formula $\exists z(x + s(z) = y)$, with free variables x and y. The free variables are placeholders. $\exists z(x + s(z) = y)$ defines the less-than relation because, for any numbers a and b, $\exists z(\overline{a} + s(z) = \overline{b})$ is true (in $\mathfrak{N}$) iff $a < b$. In general:

Definition 8.1

An $\mathfrak{L}_A$-formula $A(x_1, \ldots, x_n)$ *defines* an n-place relation R on $\mathbb{N}$ iff, for all numbers $a_1, \ldots, a_n$, $A(\overline{a_1}, \ldots, \overline{a_n})$ is true in $\mathfrak{A}$ iff $a_1, \ldots, a_n$ stand in the relation R.

Exercise 8.1 Give two other $\mathfrak{L}_A$-formulas that define the less-than relation, and explain why they do so.

So far, I've talked about relations. We can also define new functions in $\mathfrak{L}_A$. For example, we might say that the expression $x \times x$ defines the squaring function that maps any number a to a^2. Often, however, a function won't be definable in this way by an $\mathfrak{L}_A$-term. Instead, we have to resort to what I called a "syncategorematic" definition in Section 4.2. For example, we'll see that there is an $\mathfrak{L}_A$-formula $F(x, y)$ such that $F(\overline{a}, \overline{b})$ is true (in $\mathfrak{A}$) iff b is the factorial $a!$ of a. (That is, $b = 1 \times 2 \times \ldots \times a$.) With the help of this formula, any statement about factorials can be expressed in $\mathfrak{L}_A$. The (false) statement that every number is less than its factorial, for example, can be expressed as

$$\forall x \forall y (F(x, y) \rightarrow x < y).$$

It's important to clarify what kind of "definition" we are after. In the previous chapter, I talked about defining functions by primitive recursion. The primitive recursive definition of the factorial function would look as follows:

$$0! = 1$$
$$s(y)! = s(y) \times y!.$$

But this doesn't define the factorial function *in* $\mathfrak{L}_A$. A definition in $\mathfrak{L}_A$ would be a single $\mathfrak{L}_A$-formula $F(x, y)$ that is true of x and y iff y is the factorial of x.

Definition 8.2

An $\mathfrak{L}_A$-formula $A(x_1, \ldots, x_n, y)$ *defines* a (total) n-ary function f on $\mathbb{N}$ iff, for all numbers $a_1, \ldots, a_n, b$, $A(\overline{a_1}, \ldots, \overline{a_n}, \overline{b})$ is true in $\mathfrak{A}$ iff $f(a_1, \ldots, a_n) = b$.

This definition could be extended to partial functions, but we'll only be interested in total functions in this chapter.

We say that a relation or function is *definable* in $\mathfrak{L}_A$ if it is defined by some $\mathfrak{L}_A$-formula.

Exercise 8.2 Give an $\mathfrak{L}_A$-formula that defines the addition function.

Exercise 8.3 Give an $\mathfrak{L}_A$-formula that defines the switcheroo function δ that maps any positive number to 0 and 0 to 1.

We'll show that all recursive functions and relations – and therefore, by the Church-Turing thesis, all computable functions and relations – are definable in $\mathfrak{L}_A$. This is not obvious. The factorial function, for example, is recursive, but it is hard to find an $\mathfrak{L}_A$-formula $F(x, y)$ that defines it. (Try it!)

For a different type of example, consider the halting-with-bound relation that holds between the code number of a Turing machine M, a number n, and a number k iff M halts on input n within k steps. This relation is computable: we can simply run M on n for k steps, and return 'yes' if M has halted by then, and 'no' otherwise. Since all computable relations are definable in $\mathfrak{L}_A$, there must be an $\mathfrak{L}_A$-expression $H(x, y, z)$ so that $H(\overline{m}, \overline{n}, \overline{k})$ is true (in $\mathfrak{A}$) iff the Turing machine coded by m halts on input n within k steps. It's not at all obvious what such a formula would look like.

As I mentioned in Section 5.5, the definability of halting-with-bound leads to a version of Gödel's incompleteness theorem. If $H(x, y, z)$ defines halting-with-bound then $\exists z H(x, y, z)$ defines the halting relation: $\exists z H(\overline{m}, \overline{n}, z)$ is true (in $\mathfrak{A}$) iff the Turing machine coded by m halts on input n. We know from Theorem 6.1 that there is no algorithm that decides the halting relation. It follows that there can be no complete, computably axiomatizable, and true theory of arithmetic: otherwise we could decide the halting relation by checking, for any numbers m and n, whether $\exists z H(\overline{m}, \overline{n}, z)$ or its negation is entailed by the axioms.

This version of the incompleteness theorem is sometimes called "semantic", as it concerns theories that are *true*. Gödel himself gave prominence to a "syntactic" version of incompleteness that (in its contemporary form) only requires the relevant theories to be consistent. The syntactic theorem relies not on definability in $\mathfrak{L}_A$ but on a slightly stronger concept – representability – that I'm going to introduce next.

Exercise 8.4 Explain why every finite set is definable in $\mathfrak{L}_A$.

Exercise 8.5 Explain why the Busy Beaver function Σ is definable in $\mathfrak{L}_A$. (Hint: consider the relation that holds between four numbers m, n, t, k iff m codes a Turing machine with n states that halts, on blank input, after t steps leaving k strokes on the tape.)

8.2 Representability

'+', '×', 's' and '0' are non-logical symbols. They have an *intended interpretation*, but this interpretation isn't built into the language. An axiomatic $\mathfrak{L}_A$-theory doesn't automatically "know" what the non-logical symbols mean. '0' and '$s(0)$', for example, denote different numbers in the intended interpretation, but an axiomatic theory may not be able to prove '$0 \neq s(0)$'.

Exercise 8.6 Specify an $\mathfrak{L}_A$-theory that proves $0 = s(0)$.

The standard axiomatic theory of arithmetic, Peano Arithmetic (PA), can prove $0 \neq s(0)$. Indeed, whenever $a \neq b$, PA contains $\overline{a} \neq \overline{b}$. Trivially, if $a = b$ then PA contains $\overline{a} = \overline{b}$, for then $\overline{a}$ and $\overline{b}$ are the same term. So PA knows all particular facts about equality between numbers: that $\overline{17} = \overline{17}$, that $\overline{17} \neq \overline{29}$, and so on.

PA also knows all particular facts about the less-than relation: whenever $a < b$, PA contains $\overline{a} < \overline{b}$, and whenever $a \not< b$, PA contains $\neg(\overline{a} < \overline{b})$. Of course, '$<$' isn't really part of the language. What I mean is that whenever $a < b$, PA contains $\exists z(\overline{a} + s(z) = \overline{b})$, and whenever $a \not< b$, PA contains $\neg\exists z(\overline{a} + s(z) = \overline{b})$. In that sense, $\exists z(x + s(z) = y)$ *represents* the less-than relation in PA.

Definition 8.3

An $\mathfrak{L}_A$-formula $A(x_1, \ldots, x_n)$ *represents* an n-ary relation R on $\mathbb{N}$ in a theory T iff, for all numbers $a_1, \ldots, a_n$,

(i) if R holds of $a_1, \ldots, a_n$, then $\vdash_T A(\overline{a_1}, \ldots, \overline{a_n})$, and
(ii) if R does not hold of $a_1, \ldots, a_n$, then $\vdash_T \neg A(\overline{a_1}, \ldots, \overline{a_n})$.

Exercise 8.7 Explain why every relation is representable in the inconsistent theory.

Exercise 8.8 Explain why a formula defines a relation in $\mathfrak{L}_A$ iff it represents the relation in $\mathrm{Th}(\mathfrak{A})$. ($\mathrm{Th}(\mathfrak{A})$ is the set of all $\mathfrak{L}_A$-sentences that are true in $\mathfrak{A}$).

We can also talk about representing functions:

Definition 8.4

An $\mathfrak{L}_A$-formula $A(x_1, \ldots, x_n, y)$ *represents* a (total) n-ary function f on $\mathbb{N}$ in T iff, for all numbers $a_1, \ldots, a_n$,

(i) $\vdash_T A(\overline{a_1}, \ldots, \overline{a_n}, \overline{f(a_1, \ldots, a_n)})$, and
(ii) $\vdash_T \forall y(A(\overline{a_1}, \ldots, \overline{a_n}, y) \to y = \overline{f(a_1, \ldots, a_n)})$.

Here, $\overline{f(a_1, \ldots, a_n)}$ is the $\mathfrak{L}_A$-term with $f(a_1, \ldots, a_n)$ occurrences of 's'. Condition (ii) effectively requires T to know that $A(x_1, \ldots, x_n, y)$ expresses a functional relationship (as discussed in Section 4.2).

An example may help. What is needed for a formula $F(x, y)$ to represent the factorial function in a theory T? Condition (i) requires that whenever b is the factorial of a then T proves $F(\overline{a}, \overline{b})$. This leaves open that T also proves $F(\overline{a}, \overline{c})$ for some $c \neq b$. Indeed, the formula $x = x \wedge y = y$ passes condition (i) in any theory T, but there's no good sense in which this formula represents the factorial function. By condition (ii), T must know that there is no y other than b for which $F(\overline{a}, y)$ holds.

If a function or relation is represented in a theory T by some formula, we say that the function or relation is *representable* in T.

Proposition 8.1

If an $\mathfrak{L}_A$-formula represents a function in a theory $T \subseteq \mathrm{Th}(\mathfrak{A})$, then the formula also defines that function.

Proof. Assume $A(x, y)$ represents a function f in a theory T, where $T \subseteq \mathrm{Th}(\mathfrak{A})$. I'll assume for readability that f is one-place. We have to show that for all numbers a, b, $A(\overline{a}, \overline{b})$ is true in $\mathfrak{A}$ iff $f(a) = b$. For the 'if' direction, assume that $f(a) = b$. By condition (i) in definition 8.4, $\vdash_T A(\overline{a}, \overline{b})$. Since $T \subseteq \mathrm{Th}(\mathfrak{A})$, $A(\overline{a}, \overline{b})$ is true in $\mathfrak{A}$. For the 'only if' direction, assume that $f(a) \neq b$. By condition (ii) in definition 8.4, $\vdash_T \forall y(A(\overline{a}, y) \to y = \overline{f(a)})$. So $\forall y(A(\overline{a}, y) \to y = \overline{f(a)})$ is true in $\mathfrak{A}$. Since $\overline{f(a)} \neq \overline{b}$ is true in $\mathfrak{A}$, it follows that $\neg A(\overline{a}, \overline{b})$ is true in $\mathfrak{A}$, and so $A(\overline{a}, \overline{b})$ is false in $\mathfrak{A}$. □

Exercise 8.9 '$x+y=z$' represents addition in PA. Let PA* be the theory obtained by swapping '+' and '×' everywhere in PA. Can you find a formula that represents addition in PA*?

In the following sections, we'll show that all recursive functions are representable in any arithmetical theory that knows some basic facts about arithmetic. It will follow by Proposition 8.1 that all recursive functions are definable in $\mathfrak{L}_A$.

We can focus on functions because the representability of recursive functions entails the representability of recursive relations, at least in any theory that can prove $0 \neq 1$. (Recall that a relation is recursive iff its characteristic function is recursive.)

Proposition 8.2

A relation R is representable in a theory T iff its characteristic function χ_R is representable in T, provided that $\vdash_T 0 \neq \overline{1}$.

Proof. I'll assume for readability that R is a one-place relation.

For the left-to-right direction, assume that R is represented in T by some formula $A(x)$. Let $C(x,y)$ be the formula $(A(x) \wedge y=\overline{1}) \vee (\neg A(x) \wedge y=0)$. I claim that $C(x,y)$ represents the characteristic function χ_R of R. Assume first that $\chi_R(a) = 1$. Then R holds of a, and T proves $A(\overline{a})$. Since $A(\overline{a})$ entails both $C(\overline{a},\overline{1})$ and $\forall y(C(\overline{a},y) \rightarrow y = \overline{1})$, T proves both of these as well. Similarly, if $\chi_R(a) = 0$, then T proves $\neg A(\overline{a})$, which entails $C(\overline{a},0)$ and $\forall y(C(\overline{a},y) \rightarrow y = 0)$. So whenever $\chi_R(a) = b$ then T proves $C(\overline{a},\overline{b})$ and $\forall y(C(\overline{a},y) \rightarrow y = \overline{b})$. So $C(x,y)$ satisfies the two conditions in definition 8.4.

For the other direction, assume $A(x,y)$ represents χ_R in T. This means that whenever R holds of a, then $\vdash_T A(\overline{a},\overline{1})$, by condition (i) in definition 8.4. Moreover, when R does not hold of a, then $\vdash_T A(\overline{a},0)$ by condition (i) and $\vdash_T \forall y A(\overline{a},y) \rightarrow y = 0$ by condition (ii). Assuming that $\vdash_T 0 \neq \overline{1}$, it follows that $A(x,1)$ represents R in T. □

8.3 Conditions for Representability I

We want to show that every recursive function is representable in any theory that knows some basic facts about arithmetic. We proceed by induction on the construction of recursive functions. By definition 7.4 in Section 7.3, every recursive function is constructed from the base functions zero, successor, and projection by composition, primitive recursion, and regular minimization. We'll first show that the base functions are representable

in any theory. Then we'll show that representability in a theory T is preserved under composition, primitive recursion, and regular minimization, provided that T knows certain facts about arithmetic.

Let's start with the zero function z that maps every number a to 0. It's not hard to find a formula $A(x, y)$ so that $A(\overline{a}, \overline{b})$ is true (in $\mathfrak{N}$) iff $b = 0$. The formula $x = x \wedge y = 0$ does the job. So $x = x \wedge y = 0$ defines z in $\mathfrak{L}_A$. We need to confirm that it also represents z in any theory T.

Lemma 8.1
The zero function z is representable in every theory T.

Proof. I claim that z is represented in every theory T by the formula $x = x \wedge y = 0$. By definition 8.4, this means that, for all numbers a, T can prove

(i) $\overline{a} = \overline{a} \wedge 0 = 0$, and
(ii) $\forall y((\overline{a} = \overline{a} \wedge y = 0) \to y = 0)$.

Both of these are logical truths. □

Lemma 8.2
The successor function s is representable in every theory T.

Proof. I claim that $s(x) = y$ represents the successor function in every theory T: for all numbers a, T can prove

(i) $s(\overline{a}) = \overline{s(a)}$, and
(ii) $\forall y(s(\overline{a}) = y \to y = \overline{s(a)})$.

Since $\overline{s(a)}$ and $s(\overline{a})$ are the same term, both of these are logical truths. □

Lemma 8.3
Each projection function π^n_i is representable in every theory T.

Proof. Exercise.

Exercise 8.10 Prove Lemma 8.3.

Now for the closure operations. We start with composition.

Lemma 8.4

If an m-place function f is representable in a theory T and m n-place functions $g_1, \dots, g_m$ are representable in T, then the composition $h = Cn[f, g_1, \dots, g_m]$ is representable in T.

Proof. For readability, I only give the proof for $n = 1$ and $m = 2$.

Assume that f is represented in a theory T by a formula $F(x_1, x_2, y)$, and that g_1, g_2 are represented in T by $G_1(x, y_1)$ and $G_2(x, y_2)$, respectively. I claim that $h = Cn[f, g_1, g_2]$ is represented in T by the formula

$$\exists v_1 \exists v_2 (G_1(x, v_1) \wedge G_2(x, v_2) \wedge F(v_1, v_2, y)).$$

Condition (i) for representations requires that whenever $h(a) = b$ then

$$\vdash_T \exists v_1 \exists v_2 (G_1(\overline{a}, v_1) \wedge G_2(\overline{a}, v_2) \wedge F(v_1, v_2, \overline{b})).$$

So assume $h(a) = b$. Then there are c_1, c_2 such that $g_1(a) = c_1$, $g_2(a) = c_2$, and $f(c_1, c_2) = b$. Since g_1 and g_2 are represented by $G_1(x, y_1)$ and $G_2(x, y_2)$, respectively, and f is represented by $F(x_1, x_2, y)$, we have

$$\begin{gathered} \vdash_T G_1(\overline{a}, \overline{c_1}) \\ \vdash_T G_2(\overline{a}, \overline{c_2}) \\ \vdash_T F(\overline{c_1}, \overline{c_2}, \overline{b}). \end{gathered}$$

The desired claim follows by the fact that T is closed under first-order consequence.

For condition (ii), we have to show that

$$\vdash_T \forall y (\exists v_1 \exists v_2 (G_1(\overline{a}, v_1) \wedge G_2(\overline{a}, v_2) \wedge F(v_1, v_2, y)) \rightarrow y = \overline{f(g_1(a), g_2(a))}).$$

This, too, follows from the representability conditions for F, G_1, and G_2, which yield

$$\vdash_T \forall y(G_1(\overline{a}, y) \rightarrow y = \overline{c_1})$$
$$\vdash_T \forall y(G_2(\overline{a}, y) \rightarrow y = \overline{c_2})$$
$$\vdash_T \forall y(F(\overline{c_1}, \overline{c_2}, y) \rightarrow y = \overline{f(c_1, c_2)}).$$

□

I leave the case of primitive recursion for last: it is by far the hardest. Let's turn to regular minimization. Suppose $h = \mathrm{Mn}[f]$, where f is a regular function. Recall that $f(x, y)$ is regular if it is total and for all x there is some y such that $f(x, y) = 0$. The function h takes a number x and returns the least y for which $f(x, y) = 0$. Assuming that f is represented in T by some formula $F(x, y, z)$, we have: $h(x) = y$ iff $F(x, y, 0)$ and there is no $z < y$ such that $F(x, z, 0)$. We can directly translate this into $\mathfrak{L}_A$:

$$F(x, y, 0) \wedge \forall z(z < y \rightarrow \neg F(x, z, 0)).$$

This formula defines h in $\mathfrak{L}_A$. Some assumptions are needed for it to represent h in a theory T. Informally speaking, the theory must have some idea of what $<$ means. The following conditions are sufficient.

R1 For all a, $\vdash_T \forall x(\overline{a} < x \vee x = \overline{a} \vee x < \overline{a})$.

R2 $\vdash_T \neg\exists x(x < 0)$.

R3 For all $a > 0$, $\vdash_T \forall x(x < \overline{a} \rightarrow (x = 0 \vee \ldots \vee x = \overline{a - 1}))$.

R4$^-$ For all $a > 0$, $\vdash_T \overline{a} \neq 0$

Officially, of course, '$<$' isn't part of the language. I assume here, and in what follows, that '$t_1 < t_2$' is short for '$\exists z(s(z) + t_1 = t_2)$', where z is a variable that doesn't occur in t_1 or t_2.

Lemma 8.5

If a regular function f is representable in a theory T, and T satisfies the conditions R1, R2, R3, and R4$^-$, then the minimization $\mathrm{Mn}[f]$ of f is representable in T.

Proof. Assume that $F(x, y, z)$ represents f in T, and T satisfies R1, R2, R3, and R4$^-$. For readability, I'll assume that f is a two-place function. I claim that $h = \mathrm{Mn}[f]$ is represented in T by $F(x, y, 0) \wedge \forall z(z < y \rightarrow \neg F(x, z, 0))$. We have to confirm that whenever $h(a) = b$ then T can prove

(i) $F(\overline{a}, \overline{b}, 0) \wedge \forall z(z < \overline{b} \rightarrow \neg F(\overline{a}, z, 0))$.
(ii) $\forall y(F(\overline{a}, y, 0) \wedge \forall z(z < y \rightarrow \neg F(\overline{a}, z, 0)) \rightarrow y = \overline{b})$.

From the fact that $h = \mathrm{Mn}[f]$ and $h(a) = b$, we know that $f(a, b) = 0$ and $f(a, c) \neq 0$ for all $c < b$. Since f is represented in T by F, T can prove

$$F(\overline{a}, \overline{b}, 0) \tag{1}$$

as well as (for $c < b$)

$$\forall y(F(\overline{a}, \overline{c}, y) \rightarrow y = \overline{f(a, c)}). \tag{2}$$

From (2) and R4$^-$, it follows that T can prove $\neg F(\overline{a}, \overline{c}, 0)$ for all $c < b$. By R3, T can prove $\forall z(z < \overline{b} \rightarrow (z = 0 \vee \ldots \vee z = \overline{b - 1}))$ whenever $b > 0$. So if $b > 0$ then T can prove

$$\forall z(z < \overline{b} \rightarrow \neg F(\overline{a}, z, 0)). \tag{3}$$

For $b = 0$, (3) follows from R2. (i) is the conjunction of (1) and (3).

For (ii), we show (by reasoning "inside T") that T can derive $y = \overline{b}$ from

$$F(\overline{a}, y, 0) \wedge \forall z(z < y \rightarrow \neg F(\overline{a}, z, 0)). \tag{4}$$

(1) and (4) imply $\neg(\overline{b} < y)$. From (3) and (4), we have $\neg(y < \overline{b})$. By R1, it follows that $y = \overline{b}$. □

Now for the hard part: primitive recursion.

8.4 Conditions for Representability II

Return to the factorial function that maps each number n to $n! = 1 \times 2 \times \ldots \times n$. The construction by primitive recursion goes as follows:

$$0! = 1$$
$$s(y)! = s(y) \times y!$$

We need to find a formula $F(x, y)$ that expresses this function in $\mathfrak{L}_A$, so that $F(x, y)$ is true of numbers a and b iff $a! = b$.

The trick is to see the recursive construction as defining a sequence: $\langle 0!, 1!, 2!, \ldots, n! \rangle$. Our formula $F(x, y)$ will say that "y is the last element of the sequence $\langle 0!, 1!, \ldots x! \rangle$".

Of course, $\mathfrak{L}_A$ doesn't have terms for sequences. But we know that sequences of numbers can be coded as single numbers. Suppose we can find a formula $\text{ENTRY}(x, i, y)$ that expresses "y is the i-th entry in the sequence coded by x". Using $\text{ENTRY}(x, i, y)$, we can then define a formula $\text{SEQ}(z, x)$ saying that

(i) z codes a sequence whose first entry is 1, and

(ii) for all $i < x$, the $s(i)$-th entry in the sequence coded by z is the product of the ith entry and $s(i)$.

So $\text{SEQ}(z, x)$ will say that z codes the sequence $\langle 0!, 1!, \dots, x! \rangle$. From this, we can define $F(x, y)$ as $\exists z(\text{SEQ}(z, x) \wedge \text{ENTRY}(z, s(x), y))$. This says that there is a number z that codes $\langle 0!, 1!, \dots, x! \rangle$ and that y is the $s(x)$-th entry in that sequence. (We need an $s(x)$ here because $F(x, 0)$ is the *first*, not the *zero*-th, entry in the sequence.)

The main task, then, is to find the formula $\text{ENTRY}(x, i, y)$ that holds of numbers x, i, y iff y is the i-th entry in the sequence coded by x.

You may remember that we faced a similar task in Section 7.2. There, we used the prime exponents method to code sequences of numbers, and we showed that there is a primitive recursive function $\text{entry}(x, y)$ that returns the y-th number in the sequence coded by x. Unfortunately, our construction of the entry function involved primitive recursion, so we can't assume that this function is definable in $\mathfrak{L}_A$. In fact, we won't code sequences of numbers in terms of prime exponents, as we don't even have an $\mathfrak{L}_A$-formula for exponentiation yet. We'll use a different coding method.

To explain that method, assume first that we want to code a sequence $\langle a_1, a_2, \dots, a_n \rangle$ of numbers all of which are below 9. We could simply concatenate their decimal representation: $\langle 1, 7, 0, 7 \rangle$ would be coded as 1707. To simplify accessing individual elements of the sequence, we might store the indices (the position numbers) of the elements in the code, so that $\langle 1, 7, 0, 7 \rangle$ gets coded as 11273047. The third element can now be identified as the digit to the right of the '3' (in decimal representation). As it stands, this doesn't quite work because the indices can also be among the coded elements, as is the case for the number 1 in the example: there are two digits to the right of a '1'. We can disambiguate the indices by prefixing them with yet another digit, 9, that doesn't occur among the coded numbers. The code of $\langle 1, 7, 0, 7 \rangle$ becomes 911927930947. The i-th element can now be retrieved as the unique digit to the right of '9i' (in decimal representation).

We'll adapt this scheme to code arbitrary sequences of numbers. We obviously can't assume that all of the numbers are below 9. We therefore code sequences not to base 10, but to some base p that is at least 2 greater than all numbers in the sequence and all index numbers ($p - 1$ is used to mark the index numbers). For convenience, we'll always use a prime number as the base p. The sequence $\langle 1, 12, 0 \rangle$, for example, would be coded in

base 17 as

$$16 \overset{17}{\frown} 1 \overset{17}{\frown} 1 \overset{17}{\frown} 16 \overset{17}{\frown} 2 \overset{17}{\frown} 12 \overset{17}{\frown} 16 \overset{17}{\frown} 3 \overset{17}{\frown} 0,$$

where '$\overset{17}{\frown}$' is the operation of concatenation in base 17. (I'll define this formally below.) If q is the code number of a sequence in base p, the i-th element of the sequence can be retrieved as

> alpha(p, q, i) = the unique number x for which $(p-1) \overset{p}{\frown} i \overset{p}{\frown} x$ is part of the base-p numeral of q.

We'll see that this can be expressed in $\mathfrak{L}_A$.

The alpha function retrieves elements from the code q of a sequence, but it also needs the base p as a key to the code. We can get rid of the extra argument by coding $\langle p, q \rangle$ into a single number. We have to use a different coding method here. We'll use the pairing function that we met in Section 3.1 when we discussed Cantor's zig-zag method:

$$\mathrm{j}(x, y) = \frac{1}{2}(x + y)(x + y + 1) + y$$

The j function is easily defined in $\mathfrak{L}_A$. We can also define two functions l and r that extract the elements of a pair encoded by j, so that $\mathrm{l}(\mathrm{j}(x, y)) = x$ and $\mathrm{r}(\mathrm{j}(x, y)) = y$. If we have coded a sequence of numbers $\langle a_1, \ldots, a_n \rangle$ by p and q as described above, and packaged these into a single number $c = \mathrm{j}(p, q)$, we can now retrieve any element a_i of the doubly coded sequence as

$$\mathrm{beta}(c, i) = \mathrm{alpha}(\mathrm{l}(c), \mathrm{r}(c), i).$$

We'll show that this function beta is definable in $\mathfrak{L}_A$. The formula $\mathrm{BETA}(x, y, z)$ that defines it is formula $\mathrm{ENTRY}(x, y, z)$ that we were looking for.

I'm not actually going to write down $\mathrm{BETA}(x, y, z)$ as an $\mathfrak{L}_A$-formula. Instead, I'll show that the function beta can be constructed by composition and minimization from certain functions and relations that are easily definable in $\mathfrak{L}_A$: projection, addition, multiplication, and the characteristic function $\chi_=$ of identity. $\chi_=$ is the two-place function that returns 1 if its two arguments are equal and 0 otherwise. It is defined in $\mathfrak{L}_A$ by $(x = y \wedge z = 1) \vee (x \neq y \wedge z = 0)$. Projection, addition, and multiplication are also easily definable. We've dealt with projection in Lemma 8.3. Addition is defined by $x + y = z$; multiplication by $x \times y = z$. To ensure that these formulas also *represent* the relevant functions in a theory T, we need the following assumptions:

R4 For all a, b, if $a \neq b$ then $\vdash_T \overline{a} \neq \overline{b}$.
R5 For all a, b, $\vdash_T \overline{a} + \overline{b} = \overline{a + b}$.
R6 For all a, b, $\vdash_T \overline{a} \times \overline{b} = \overline{a \times b}$.

Note that R4 subsumes R4$^-$.

Lemma 8.6

Projection, addition, multiplication, and $\chi_=$ are representable in every theory that satisfies R4–R6.

Proof. Projection is representable in every theory by Lemma 8.3.

The addition function is represented by $x + y = z$ in every theory T that satisfies R5: condition (i) in the definition of representation is given by R5; condition (ii) then holds by first-order logic.

The multiplication function is represented by $x \times y = z$ in every theory T that satisfies R6: condition (i) is given by R6; (ii) holds by first-order logic.

$\chi_=$ is represented by $(x = y \wedge z = 1) \vee (x \neq y \wedge z = 0)$ in every theory T that satisfies R4. For condition (i), we need to show that if $\chi_=(a, b) = c$ then $\vdash_T (\overline{a} = \overline{b} \wedge \overline{1} = \overline{c}) \vee (\overline{a} \neq \overline{b} \wedge \overline{0} = \overline{c})$. There are two ways in which $\chi_=(a, b) = c$ can hold: either $a = b$ and $c = 1$ or $a \neq b$ and $c = 0$. In the first case, $\overline{a} = \overline{b}$ and $\overline{1} = \overline{c}$ are logical truths. In the second case, $\overline{0} = \overline{c}$ is a logical truth and $\overline{a} \neq \overline{b}$ holds by R4. Condition (ii) holds by first-order logic. □

Now we need to show that beta can be constructed from projection, addition, multiplication, and $\chi_=$ by composition and minimization. To this end, let's first introduce a name for the class of functions that can be so constructed. I'll call a function *legit* if it can be constructed from projection, addition, multiplication, and $\chi_=$ by composition and minimization; I'll call a relation *legit* if its characteristic function is legit.

Lemma 8.7

All legit functions and relations are representable in any theory that satisfies R1–R6.

Proof. By Lemma 8.6, projection, addition, multiplication, and $\chi_=$ are representable in any theory that satisfies R4–R6. By Lemmas 8.4 and 8.5, composition and mini-

mization preserve representability in any theory that satisfies R1–R4. □

The following lemmas show that the class of legit relations is closed under truth-functional combinations, bounded quantification, and regular minimization. We've met the first two of these operations in Section 7.2. For the third, recall that the minimization $\mu y\, R(x_1, \dots, x_n, y)$ of an $n+1$-ary relation R is the n-ary function that maps $x_1, \dots, x_n$ to the least y such that $R(x_1, \dots, x_n, y)$. *Regular* minimization is minimization applied to a relation that is *regular* in the sense that for all $x_1, \dots, x_n$ there is some y such that $R(x_1, \dots, x_n, y)$.

Lemma 8.8
The legit relations are closed under truth-functional combinations.

Proof. Let R and S be legit relations (of arity 1, for readability), and χ_R and χ_S their characteristic functions. Then

$$\begin{aligned} \chi_{R \wedge S}(x) &= \chi_R(x) \times \chi_S(x) \\ \chi_{\neg R}(x) &= \chi_=(\chi_R(x), 0). \end{aligned}$$

All truth-functional combinations can be constructed from conjunction and negation. □

Lemma 8.9
The legit relations are closed under regular minimization.

Proof. Let $R(x_1, \dots, x_n, y)$ be a regular legit relation. Then $\mu y\, R(x_1, \dots, x_n, y)$ is $\mathrm{Mn}[\chi_{\neg R}]$. Since $\chi_{\neg R}$ is legit by Lemma 8.8, so is $\mathrm{Mn}[\chi_{\neg R}]$. □

Lemma 8.10
The legit relations are closed under bounded quantification: if $R(x_1, \dots, x_n, z)$ is a legit $n+1$-ary relation, then so are the $n+1$-ary relations $\forall z \leq y\, R(x_1, \dots, x_n, z)$ and $\exists z \leq y\, R(x_1, \dots, x_n, z)$.

Proof. Let $R(x_1, \dots, x_n, z)$ be a legit relation. For readability, I assume $n = 1$. Let $d(x, y) = \mu z[\neg R(x, z) \vee y = z]$. By Lemma 8.8, μz is here applied to a legit relation,

and the final disjunct ensures that the relation is regular. So d is legit by Lemma 8.9. If $d(x,y) < y$ then $\neg R(x, d(x,y))$; if $d(x,y) = y$ then $\forall z \leq y\, R(x,z)$. So $\forall z \leq y\, R(x,z)$ holds iff $d(x,y) = y$. (I.e., $\chi_{\forall z \leq y\, R(x,z)}(x,y) = \chi_{=}(d(x,y), y)$.)

Since $\exists z \leq y\, R(x,z)$ is equivalent to $\neg \forall z \leq y\, \neg R(x,z)$, it is legit by Lemma 8.8. □

Exercise 8.11 We know from the proof of Lemma 8.6 that addition is representable in every theory that satisfies R5. Use Lemmas 8.7 and Lemma 8.10 to infer that $\leq$ is representable in any theory that satisfies R1–R5, by defining $\leq$ in terms of addition and bounded quantification.

Lemma 8.11: Beta Function Lemma

There is a function beta such that for any finite sequence $\langle a_1, \ldots, a_n \rangle$ of natural numbers, there is a number c such that $\text{beta}(c,i) = a_i$ for all $1 \leq i \leq n$. Moreover, beta is representable in any theory that satisfies R1–R6.

Proof. We use the coding method described above: given a sequence $\langle a_1, \ldots, a_n \rangle$, let p be the smallest prime that's at least 2 greater than all of $a_1, \ldots, a_n$ and n; let q be the base-p numeral built as $(p-1) \overset{p}{\frown} 1 \overset{p}{\frown} a_1 \overset{p}{\frown} \ldots \overset{p}{\frown} (p-1) \overset{p}{\frown} n \overset{p}{\frown} a_n$; let $c = \text{j}(p,q)$. I'll now show how we can construct a function $\text{beta}(c,i)$ that retrieves the i-th element from the sequence coded by c:

$$x < y \Leftrightarrow \exists z \leq x (s(z) = y).$$

$$\text{Divides}(x,y) \Leftrightarrow \exists z \leq y\, (y = x \times z).$$

$$\text{Prime}(x) \Leftrightarrow x \neq 0 \wedge x \neq 1 \wedge \forall y \leq x\, (\text{Divides}(y,x) \rightarrow y = 1 \vee y = x).$$

$$\text{Pow}(x,p) \Leftrightarrow x \neq 0 \wedge \text{Prime}(p) \wedge \forall y \leq x\, (\text{Divides}(y,x) \rightarrow y = 1 \vee \text{Divides}(p,y))$$

(in words: x is a power of the prime p).

$$\eta(p,x) = \mu y\, ((\text{Pow}(y,p) \wedge x < y \wedge 1 < y) \vee (\neg \text{Prime}(p) \wedge y = 0))$$

(the smallest power of prime p greater than x).

$$x \overset{p}{\frown} y = x \times \eta(p,y) + y$$

(the base-p numeral of y appended to that of x).

$$\text{Part}(x,y,p) \Leftrightarrow \exists v \leq y\, \exists w \leq y\, (v \overset{p}{\frown} x \overset{p}{\frown} w = y \vee v \overset{p}{\frown} x = y \vee x \overset{p}{\frown} v = y \vee x = y)$$

(the base-p numeral of x is part of the base-p numeral of y).

$$x \dot{-} y = \mu z\,((y<x \to y+z=x) \wedge (\neg(y<x) \to z=0))$$

(truncated subtraction, as in Section 7.1).

$$\text{alpha}(p,q,i) = \mu x\,(\text{Part}((p \dot{-} 1)^{\frown_p} i^{\frown_p} x, q, p) \vee x = q)$$

(the x for which $(p-1)^{\frown_p} i^{\frown_p} x$ is part of the base-p numeral of q).

$$\text{J}(x,y,q) \Leftrightarrow 2 \times q = (x+y) \times (x+y+1) + 2 \times y.$$

$$\text{r}(q) = \mu y\,\exists x \le q(\text{J}(x,y,q)).$$

$$\text{l}(q) = \mu x\,\exists y \le q(\text{J}(x,y,q)).$$

$$\text{beta}(c,i) = \text{alpha}(\text{l}(c), \text{r}(c), i).$$

To be clear: this chain of definitions doesn't directly define the relevant functions and relations in $\mathfrak{L}_A$. The expressions on the right-hand side aren't $\mathfrak{L}_A$-formulas; they are metalinguistic descriptions of certain functions and relations. The chain of definitions merely shows how beta can be constructed from addition, multiplication, projection, and $\chi_=$ by composition, truth-functional combination, bounded quantification, and regular minimization, By Lemmas 8.8, 8.10, and 8.9, it follows that beta is legit. By Lemma 8.7, beta is representable in any theory T that satisfies R1–R6. □

Now we can show that representability is closed under primitive recursion.

Lemma 8.12

If f and g are representable in a theory T that satisfies R1–R6, and $h = \text{Pr}[f,g]$, then h is also representable in T.

Proof. Assume $h = Pr[f,g]$. For readability, I assume that f is one-place. Let $F(x,y)$, $G(x,y,z,w)$, and $\text{BETA}(x,y,z)$ be $\mathfrak{L}_A$-formulas that represent f, g, and beta in T, respectively. Let $\text{SEQ}(c,x,k)$ be the formula

$$\exists u(\text{BETA}(c,1,u) \wedge F(x,u)) \wedge$$
$$\forall i(i < k \to \exists t \exists u(\text{BETA}(c,s(i),t) \wedge \text{BETA}(c,s(s(i)),u) \wedge G(x,i,t,u))).$$

This represents (in T) the relation that holds of c,x,k iff c codes $\langle h(x,0), \ldots, h(x,k)\rangle$. Let $H(x,k,y)$ be

$$\exists c(\text{SEQ}(c,x,k) \wedge \text{BETA}(c,s(k),y)).$$

This represents the relation that holds of x,k,y iff y is the last element of $\langle h(x,0), \ldots, h(x,k)\rangle$,

It therefore represents h in T. □

Exercise 8.12 (a) Show that all legit functions are recursive. (b) Can you also show that all legit functions are recursive? (Hint: if $h = \Pr[f, g]$ then h can be constructed from f, g, and beta, roughly as in the proof of Lemma 8.12.)

8.5 Summing up

The lemmas from the previous two sections combine to give us our main result:

Theorem 8.1
All recursive functions are representable in any theory that satisfies R1–R6.

Proof. Let T be any theory that satisfies R1–R6. By Lemmas 8.1, 8.2, and 8.3, the zero function, successor function, and projection functions are representable in T. By Lemmas 8.4, 8.5, and 8.12, any function constructed by composition, regular minimization, or primitive recursion from functions that are representable in T is itself representable in T. Since all recursive functions can be constructed from zero, successor, and projection by these operations, it follows that all recursive functions are representable in T. □

Theorem 8.2
All recursive relations are representable in any theory that satisfies R1–R6.

Proof. Immediate from Theorem 8.1, Proposition 8.2, and the fact that any theory T that satisfies R4 can prove $0 \neq \overline{1}$. □

The conditions R1–R6 can be seen as defining a minimal arithmetical theory in which all recursive functions and relations are representable. This theory is not especially elegant. More elegant is the theory Q (or "Robinson Arithmetic") that we met in Section 4.1. As a reminder, here are the axioms of Q:

Q1 $\forall x \forall y\, (s(x) = s(y) \to x = y)$
Q2 $\forall x\, 0 \neq s(x)$
Q3 $\forall x\, (x \neq 0 \to \exists y\, x = s(y))$
Q4 $\forall x (x + 0 = x)$
Q5 $\forall x \forall y (x + s(y) = s(x + y))$
Q6 $\forall x (x \times 0 = 0)$
Q7 $\forall x \forall y (x \times s(y) = (x \times y) + x)$

The following proof shows that Q, and therefore every extension of Q, satisfies R1–R6. An *extension* of Q is a theory that is at least as strong as Q, in the sense that it contains all sentences in Q.

Theorem 8.3
All recursive functions and relations are representable in every extension of Q.

Proof. By Theorems 8.1 and 8.2, it suffices to show that Q (and therefore any extension of Q) satisfies R1–R6.

R1. We show by induction on a that $\vdash_Q \forall x(\overline{a} < x \ \vee \ x = \overline{a} \ \vee \ x < \overline{a})$. *Base:* $a = 0$. (I now reason "inside Q".) By Q3, for all x either $x = 0$ or $\exists y\, x = s(y)$. In the second case, $\exists y(s(y) + 0 = x)$ by Q4, and so $0 < x$. So $\forall x(x = 0 \ \vee \ 0 < x)$. *Induction step:* Let x be any number. We show that $s(\overline{a}) < x \ \vee \ x = s(\overline{a}) \ \vee \ x < s(\overline{a})$. By Q3, either $x = 0$ or $\exists y\, x = s(y)$. If $x = 0$ then $x < s(\overline{a})$ because $s(\overline{a}) + 0 = s(\overline{a})$ by Q4 and hence $\exists z(s(z) + 0 = s(a))$. Assume $\exists y\, x = s(y)$. By induction hypothesis, $\overline{a} < y \ \vee \ y = \overline{a} \ \vee \ y < \overline{a}$. If $\overline{a} < y$ then $\exists z(s(z) + \overline{a} = y)$ and $s(z) + s(\overline{a}) = s(y)$ by Q5; so $s(\overline{a}) < x$. If $y = \overline{a}$ then $x = s(\overline{a})$. If $y < \overline{a}$ then $\exists z(s(z) + y = \overline{a})$ and $s(z) + s(y) = s(\overline{a})$ by Q5; so $x < s(\overline{a})$.

R2. We show that Q contains $\neg\exists x(x < 0)$. Fix any x, z. By Q3, either $x = 0$ or $\exists y\, x = s(y)$. If $x = 0$, $s(z) + x = s(z) + 0 = s(z)$ by Q4, hence $s(z) + x \neq 0$ by Q2. If, alternatively, $\exists y\, x = s(y)$, then $s(z) + x = s(z) + s(y) = s(s(z) + y)$ by Q5, hence $s(z) + x \neq 0$ by Q2. So Q2–Q5 entail $\forall x \neg\exists z(s(z) + x = 0)$, which is equivalent to $\neg\exists x(x < 0)$.

R3. We show by induction on a that whenever $a > 0$ then $\vdash_Q \forall x(x < \overline{a} \to (x = 0 \ \vee \ \ldots \ \vee \ x = \overline{a - 1}))$. *Base:* $a = 1$. We show that $\vdash_Q \forall x(x < \overline{1} \to x = 0)$. Assume $x < \overline{1}$; i.e. $\exists z(s(z) + x = s(0))$. Suppose for reductio that $x \neq 0$. By Q3, $\exists y\, x = s(y)$; so

$s(z) + s(y) = s(0)$; so $s(s(z) + y) = s(0)$ by Q5, and $s(z) + y = 0$ by Q1; if $y = 0$ then $s(z) + 0 = s(z) = 0$ contradicting Q2; if $y = s(w)$ then $s(z) + s(w) = s(s(z) + w) = 0$, again contradicting Q2. *Induction step:* Assume $x < s(\overline{a})$, i.e. $\exists z(s(z) + x = s(\overline{a}))$. By Q3, either $x = 0$ or $\exists y\, x = s(y)$. In the second case, $s(z) + s(y) = s(\overline{a})$, so $s(s(z) + y) = s(\overline{a})$ by Q5, and $s(z) + y = \overline{a}$ by Q1; so $y < \overline{a}$. By induction hypothesis, $y = 0 \vee \ldots \vee y = \overline{a-1}$, so $x = s(y)$ is one of $1, \ldots, \overline{a}$. Combining both cases, we have $x = 0 \vee \ldots \vee x = \overline{a}$.

R4. We show that if $a \neq b$, then $\vdash_Q \overline{a} \neq \overline{b}$. Assume $a < b$. We show by induction on a that $\vdash_Q \overline{a} \neq \overline{b}$. *Base:* $a = 0$. Then $\overline{b} = \overline{s(b-1)}$ and hence $0 \neq \overline{b}$ by Q2. *Induction step:* $a = s(c)$. Then $b = s(d)$ for some d with $c < d$. By induction hypothesis, $\vdash_Q \overline{c} \neq \overline{d}$. So $\vdash_Q s(\overline{c}) \neq s(\overline{d})$ by Q1. The case for $b < a$ is analogous.

R5. We show by induction on b that for all a, b, $\vdash_Q \overline{a} + \overline{b} = \overline{a+b}$. *Base*: $b = 0$. Then $\overline{a} + \overline{b} = \overline{a+b}$ by Q4. *Induction step:* $b = s(c)$. By induction hypothesis, $\overline{a} + \overline{c} = \overline{a+c}$. By Q5, $\overline{a} + s(\overline{c}) = s(\overline{a} + \overline{c}) = s(\overline{a+c}) = \overline{a + s(c)} = \overline{a+b}$.

R6. We show by induction on b that for all a, b, $\vdash_Q \overline{a} \times \overline{b} = \overline{a \times b}$. *Base*: $b = 0$. Then $\overline{a} \times \overline{b} = \overline{a \times b}$ by Q6. *Induction step:* $b = s(c)$. Then $\overline{a} \times s(\overline{c}) = \overline{a} \times \overline{c} + \overline{a}$ by Q7, $= \overline{a \times c} + \overline{a}$ by induction hypothesis, $= \overline{a \times c + a}$ by R5, $= \overline{a \times b}$. □

By Proposition 4.1, Peano Arithmetic (PA) is an extension of Q. So all recursive functions and relations are representable in PA.

We can also prove a result that goes in the other direction:

Theorem 8.4

Every relation that is representable in a computably axiomatizable and consistent $\mathfrak{L}_A$-theory is recursive.

Proof. Assume $A(x_1, \ldots, x_n)$ represents R in a computably axiomatizable and consistent theory T. That is, if R holds of some numbers $a_1, \ldots, a_n$, then T can prove $A(\overline{a_1}, \ldots, \overline{a_n})$, and if R does not hold of $a_1, \ldots, a_n$, then T can prove $\neg A(\overline{a_1}, \ldots, \overline{a_n})$. Since T is consistent, it never proves both $A(\overline{a_1}, \ldots, \overline{a_n})$ and $\neg A(\overline{a_1}, \ldots, \overline{a_n})$. By Proposition 5.4, every computably axiomatizable first-order theory is computably enumerable: we can define an algorithm that lists all sentences provable in T. This gives us an algorithm for deciding R: to check whether R holds of some numbers $a_1, \ldots, a_n$, we wait until either $A(\overline{a_1}, \ldots, \overline{a_n})$ or $\neg A(\overline{a_1}, \ldots, \overline{a_n})$ appears on the list of sentences provable in T. By the Church-Turing thesis, it follows that R is recursive. □

Finally, let's return to definability. As I announced in Section 8.2, our result about representability shows that all recursive functions and relations can be defined in $\mathfrak{L}_A$:

Theorem 8.5
All recursive functions and relations are definable in $\mathfrak{L}_A$.

Proof. By Theorem 8.3, all recursive functions and relations are representable in Q. All axioms of Q are true in the standard model of arithmetic $\mathfrak{A}$. So $Q \subseteq Th(\mathfrak{A})$. By Proposition 8.1, every formula that represents a function or relation in Q therefore also defines that function or relation in $\mathfrak{L}_A$ □

Exercise 8.13 Are all recursive functions and relations representable in $Th(\mathfrak{A})$?

Exercise 8.14 Using the Church-Turing theses, explain why all computably enumerable relations are definable in $\mathfrak{L}_A$. (Hint: use Proposition 5.3.)

Exercise 8.15 Are all computably enumerable relations representable in Q?

Exercise 8.16 Say that a relation R is *weakly represented* by an $\mathfrak{L}_A$-formula $A(x_1, \ldots, x_n)$ in a theory T iff for all numbers $a_1, \ldots, a_n$,

$$R(a_1, \ldots, a_n) \text{ iff } \vdash_T A(\overline{a_1}, \ldots, \overline{a_n}).$$

Explain the following facts:

(a) A relation can be weakly representable in a theory without being representable in that theory.

(b) If R is weakly representable in a computably axiomatizable and consistent theory then R is computably enumerable. (Compare Theorem 8.4.)

(c) All recursive relations are weakly representable in any $\mathfrak{L}_A$-theory that is consistent with Q. (Hint: We know that every recursive relation R is represented in Q by some formula A. Let $\hat{Q}$ be the conjunction of the seven axioms of Q, and consider the formula $\hat{Q} \to A$.)