

# A Revision-Theoretic Analysis of the Arithmetical Hierarchy

Gian Aldo Antonelli\*

Department of Philosophy

University of Pittsburgh

Pittsburgh, Pennsylvania 15260

## Abstract

In this paper we apply the idea of *Revision Rules*, originally developed within the framework of the theory of truth and later extended to a *general* mode of definition, to the analysis of the arithmetical hierarchy. This is also intended as an example of how ideas and tools from philosophical logic can provide a different perspective on mathematically more “respectable” entities. Revision Rules were first introduced by A. Gupta and N. Belnap as tools in the theory of truth, and they have been further developed to provide the foundations for a general theory of (possibly circular) definitions. Revision Rules are non-monotonic inductive operators that are iterated into the transfinite beginning with some given “bootstrapper” or “initial guess.” Since their iteration need not give rise to an *increasing* sequence, Revision Rules require a particular kind of operation of “passage to the limit,” which is a variation on the idea of the inferior limit of a sequence. We then define a sequence of sets of strictly increasing arithmetical complexity, and provide a representation of these sets by means of an operator  $G(x, \phi)$  whose “revision” is carried out over  $\omega^2$  beginning with any total function satisfying certain relatively simple conditions. Even this relatively simple constraint is later lifted, in a theorem whose proof is due to Anil Gupta.

In this paper we apply the idea of *Revision Rules*, originally developed within the framework of the theory of truth and later extended to a *general* mode of definition, to the analysis of the arithmetical hierarchy. This is also intended as an example of how ideas and tools from philosophical logic can provide a different perspective on mathematically more “respectable” entities.

Revision Rules were first introduced by A. Gupta [2] and N. Belnap [1] and, independently, Herzberger [4] as tools in the theory of truth, and have found their most detailed

---

\* Current address: Yale University, Department of Philosophy, P.O. Box 208306, New Haven, CT 06520

exposition to date in Gupta & Belnap [3], where they provide the foundations for a general theory of (possibly circular) definitions. Revision Rules are non-monotonic inductive operators that are iterated into the transfinite beginning with some given “bootstrapper” or “initial guess.” Since their iteration need not give rise to an *increasing* sequence, Revision Rules require a particular kind of operation of “passage to the limit:” rather than, as is usual in the monotone case, taking the cumulative result or union of what is obtained at previous stages, in Revision theory we take, *first* the “inferior limit” of the sequence, i.e., we assign to an item the value, if any, to which it eventually stabilizes in the sequence, and *then* we invoke the bootstrapper again to supply a value for those items that are not covered by the inferior limit.

The language we are going to use is the language of arithmetic, interpreted over the standard model. We will use lower-case italics  $a, b, c, \dots$  with or without subscripts to indicate variables ranging over the natural numbers, and uppercase italics  $A, B, C, \dots, F, G, \dots$  to refer to sets as well as functions of type  $\omega^k \times (\omega^\omega)^l$  (for natural numbers  $k$  and  $l$ ). In general, with any function  $G$  we can associate the set  $\{p : G(p) = 0\}$ , and quite often we will just speak of  $G$  as a set. This is why we choose not to make the distinction typographically sharper. Constants of either kind will be denoted by boldface symbols: so for instance  $\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots, \mathbf{p}_i, \mathbf{q}_j$  are particular numbers that we find convenient to name, while  $\mathbf{K}, \mathbf{Tot}, \mathbf{Par}, \dots$  are particular sets. We assume that some enumeration of the recursive functions has been fixed; we let  $\{c\}^A$  denote the recursive function whose code is  $c$  and whose oracle is  $A$ ; if  $A$  is recursive (in particular, empty) we just write  $\{c\}$  for  $\{c\}^A$ . In keeping with common usage in Generalized Recursion Theory, our symbols for ordinals are taken from the end of the Greek alphabet  $\rho, \sigma, \tau, \dots$ , with only a few exceptions to this rule. For instance,  $\phi, \psi$  do not refer to ordinals but to functions.<sup>1</sup> Finally, “ $\omega$ ” will do

---

<sup>1</sup> Other exceptions: we reserve  $\mu$  for the “least” operator, and  $\lambda$  for *limit* ordinals (in virtue of its obvious mnemonic value). We also use  $\lambda$  with Church’s functional notation, as in  $\lambda x.x+1$ . It is also worth noting that the conventions concerning second-order and ordinal variables are somewhat idiosyncratic to Generalized Recursion Theory. Nonetheless, we adopt them here.

double duty, in that it will be used to indicate both the first limit ordinal and the set of natural numbers. Let us first say a few informal words in the way of motivation.

Consider some equation in the language of arithmetic:

$$G(x_1, \dots, x_n) = \dots G \dots \quad (*)$$

We want to take (\*) as a *definition* of  $G$ , but of course not all such equations succeed in uniquely picking out a function. Indeed, there might be many  $G$ 's satisfying (\*), or there could be none, and we don't have in general a criterion to assess the situation one way or the other.

However, even in the absence of a suitable criterion, one sensible way to proceed is as follows: let us make some hypothesis as to what  $G$  is (for instance, suppose  $G$  is some function  $\psi$ ) and let's try to use this hypothesis in order to compute  $G$ . That is, whenever the instructions for computing  $G(x)$ , as coded by equation (\*), require that we supply some value  $G(y)$ , we use  $\psi(y)$  instead. Should we find out that our hypothesis  $\psi$  does not satisfy (\*) and therefore must be corrected in some way or other, we will revise it accordingly. Thus we obtain a new function, with which we can start the process all over again. This amounts to taking (\*) as a *rule of revision* in the style of Gupta [2], Belnap [1], and Herzberger [4], a way of getting a "better" function satisfying (\*), when supplied with some hypothesis as to the extension (or graph) of  $G$ .<sup>2</sup>

The crucial point in this procedure is to use equation (\*) to "revise" hypothetical solutions. This can be accomplished as follows. Modify the "definition" of  $G$  by introducing an extra function parameter as follows:

$$G(x_1, \dots, x_n, \phi) = \dots \phi \dots \quad (**)$$

We shall not lay any other constraints on  $\phi$ , except that it should be *total*, and this for technical as well as philosophical reasons. If we allow ourselves a "bootstrapper" to provide

---

<sup>2</sup> Indeed, the approach to computability we are about to consider was first suggested by an observation of Nuel Belnap. The sense in which the rule of revision yields a "better" candidate for  $G$  will become clear in what follows. For a general reference see Gupta and Belnap [3].

a “guess” any time the initial equation does not suffice to determine a unique output value, this might as well be total. In other words: if the function we are trying to define will fail to yield a value for some arguments, only the “definition” of  $G$  should be to blame, and not the fact that we have built partial answers right into the formalism. Moreover, as is well known, higher-type functionals usually are ill-behaved if partial functions are allowed to occur as their arguments. In particular, if such arguments are allowed, it is more difficult to assess their complexity. So, unless otherwise indicated, we decide to avoid them altogether.

Given (\*\*), and having chosen some particular function  $\psi$  to be used as a “bootstrapper,” we apply the Revision process. This process is organized in stages, as follows: at the first stage we compute  $G(\vec{x})$  according to (\*\*) (where  $\vec{x} = x_1, \dots, x_n$ ), using the bootstrapper to supply those values  $G(\vec{y})$  that might be required. This specifies some function  $G'$ . Now we repeat the process, except that we use  $G'$  to supply those values that might be required by any computation of  $G$ . Again, in this way we obtain some function  $G''$  that can then itself be used in the revision process. As we might want to iterate this process into the transfinite, we have to decide what to do at limit stages, when there is no “previous stage” to rely upon. The idea is then to set  $G(\vec{x}) = y$  at some limit stage  $\lambda$  if the value assigned to  $\vec{x}$  eventually settles =  $y$  beginning with some stage  $\rho < \lambda$  and set  $G(\vec{x}) = \psi(\vec{x})$  if the value of  $\vec{x}$  keeps oscillating as we approach  $\lambda$  from below. It is perfectly reasonable, moreover, to let the value of the bootstrapper  $\psi$  vary according to the stage (zero or limit) at which it is invoked, so that in principle we could allow  $\psi$  to have an ordinal parameter. In order to keep things simple, however, we adopt the *constant limit rule*, and use the same guess throughout. These ideas are spelled out more formally below.

**1 DEFINITION.** Let  $G(x_1, \dots, x_n, \phi)$  be an arithmetical functional; let  $\psi$  be a total function from  $n$ -tuples of natural numbers into the natural numbers. Define the  $\rho$ -revision of

$G$  under initial hypothesis  $\psi$ , by induction on  $\rho$ :

$$\begin{aligned} G_0^\psi(x_1, \dots, x_n) &= \psi(x_1, \dots, x_n) \\ G_{\rho+1}^\psi(x_1, \dots, x_n) &= G(x_1, \dots, x_n, G_\rho^\psi) \\ G_\lambda^\psi(x_1, \dots, x_n) &= \begin{cases} z, & \text{if } (\exists \sigma < \lambda)(\forall \tau < \lambda)(\sigma < \tau \\ & \rightarrow G_\tau^\psi(x_1, \dots, x_n) = z); \\ \psi(x_1, \dots, x_n), & \text{otherwise.} \end{cases} \end{aligned}$$

For limit ordinals  $\lambda$ , we also write<sup>3</sup>

$$G_\lambda^\psi(\vec{x}) = \lim_{\sigma \rightarrow \lambda} G_\sigma^\psi(\vec{x}).$$

We would like to know whether  $G_\rho^\psi(x_1, \dots, x_n)$  converges to some function as  $\rho$  increases. If so, we may consider such limit as the function defined by  $G$  relative to  $\psi$ . On the other hand, such a limit might not exist in general for all  $n$ -tuples of arguments, and yet it is possible that for *certain* arguments  $k_1, \dots, k_n$  the value of  $G_\rho^\psi(k_1, \dots, k_n)$  stabilizes at some limit as  $\rho$  increases. In this case, although the “definition”  $G(x_1, \dots, x_n, \phi) = \dots \phi \dots$  fails uniquely to identify a function, it allows us to settle the value to be assigned to  $k_1, \dots, k_n$ .

**NOTATION.** For any non-successor ordinal  $\rho$  we define  $G_{<\rho}^\psi$  to be the same as  $G_\rho^\psi$  but without the contribution of  $\psi$  at  $\rho$ . Formally:

$$G_{<\rho}^\psi(x) = y \iff (\exists \sigma < \rho)(\forall \tau < \rho)(\sigma \leq \tau \rightarrow G_\tau^\psi(x) = y).$$

From the definition it follows that if  $\rho = 0$  then  $G_{<\rho}^\psi = \emptyset$ . We can also employ a “limit” notation for this notion:

$$\begin{aligned} G_{<\rho}^\psi(x) &= \liminf_{\sigma \rightarrow \rho} G_\sigma^\psi(x) \\ &= \bigcup_{\sigma < \rho} \left( \bigcap_{\sigma < \tau < \rho} G_\tau^\psi(x) \right). \end{aligned}$$

---

<sup>3</sup> This is *not* the usual notion of limit: in the case of revision rules, the usual limit might not exist at all; in accord with the above definition, the limit should rather be construed as the “inferior limit” (in the usual sense: see below), integrated with the contribution of the bootstrapper to give a total function.

We are going to use Revision Rules to represent sets of natural numbers having higher and higher arithmetical complexity. First of all, we characterize the complexity of revision processes of length  $< \omega^2$  obtained by iterating a *recursive* revision rule.

**2 DEFINITION.** Given a natural number  $t$  and a function  $\phi$ , we write  $t \subset \phi$  if  $t$  is (the code of) an *initial segment* of  $\phi$ : for some  $n > 0$ ,

$$t = \langle \langle 0, \phi(0) \rangle, \dots, \langle n-1, \phi(n-1) \rangle \rangle.$$

Notice that the assertion “ $t \subset \phi$ ” does not involve any universal quantifier, but is equivalent to a finite conjunction of atomic sentences of the form  $((t)_i)_1 = \phi(i)$  for  $i < \ell(t)$ . Moreover, by “ $x \in t$ ” we mean  $\exists p < \ell(t) \cdot (t)_p = x$  (where  $\ell(t)$  is the length of  $t$ ).

If  $G(x, \phi)$  is a functional, by  $G'(x, t)$  we refer to the function that computes the same program as  $G$ , except that whenever  $G$  would query the oracle for the value  $\phi(n)$ ,  $G'$  looks for the first  $m$  such that  $\langle n, m \rangle \in t$  (if no such pair is in  $t$ , then  $G'$  is undefined). If  $G(x, \phi)$  is recursive, then  $G'(x, t)$  is  $\Sigma_1^0$ , and  $G(x, \phi) = y$  if and only if  $\exists t G'(x, t) = y$ . We will write  $G(x, t)$  for  $G'(x, t)$ . Observe also that “ $s \subset \lambda x \cdot G(x, t)$ ” can be written as

$$\forall x \in s \exists y \exists k [T(y, e, (x)_0) \wedge \ell(y) = k \wedge (y)_k = (x)_1],$$

where the first quantifier is clearly bounded,  $e$  is the code of a partial function computing  $G(x, t)$ , and  $T$  is Kleene’s T-predicate. It follows that if  $G$  is itself  $\Sigma_n^0$ , then so is “ $s \subset \lambda x \cdot G(x, t)$ .”

**3 THEOREM.** Let  $G(x, \phi)$  be a recursive functional. Then for every  $n$  we have that

- (a<sub>n</sub>)  $G_{<\omega \cdot n}^\psi$  is  $\Sigma_{2n}^0[\psi]$ ;
- (b<sub>n</sub>) for all  $p$ ,  $G_{\omega \cdot n + p}^\psi$  is  $\Delta_{2n+1}^0[\psi]$ .

*Proof.* We establish (a<sub>n</sub>) and (b'<sub>n</sub>) simultaneously by induction on  $n$ , where (b'<sub>n</sub>) is the statement that there is a  $\Delta_{2n+1}^0[\psi]$  predicate  $P_n(x, y, q)$  such that

$$P_n(x, y, q) \iff G_{<\omega \cdot n + q}^\psi(x) = y.$$

Clearly, b'<sub>n</sub> follows.

**CASE**  $n = 0$ . Then  $(\mathbf{a}_0)$  holds trivially since  $G_{<0}^\psi = \emptyset$  is recursive, in particular recursive in  $\psi$ , i.e.  $\Sigma_0^0[\psi]$ .

To establish  $(\mathbf{b}_0)$  we must show that there is a  $\psi$ -recursive predicate  $P(x, y, q)$  such that

$$P(x, y, q) \iff G_q^\psi(x) = y.$$

The idea is to use a form of “course-of-values” recursion. In fact, we can write:

$$\begin{aligned} G_q^\psi(x) = y &\iff \exists s[\ell(s) = q + 1 \wedge (s)_0 \subset \psi \wedge \\ &\forall i \leq p((s)_{i+1} \subset \lambda x \cdot G(x, (t)_i)) \wedge \langle x, y \rangle \in (t)_p] \end{aligned} \quad (1)$$

$$\begin{aligned} &\iff \forall s[\ell(s) = q + 1 \wedge (s)_0 \subset \psi \wedge \\ &\forall i \leq p((s)_{i+1} \subset \lambda x \cdot G(x, (t)_i)) \rightarrow \langle x, y \rangle \in (t)_p] \end{aligned} \quad (2)$$

Since  $G_q^\psi(x) = y$  can be written both in existential and universal form, it follows that it is recursive.

**CASE**  $n + 1$ . In order to establish  $(\mathbf{a}_{n+1})$ , we observe that it is possible to characterize  $G_{<\omega \cdot (n+1)}^\psi$  as follows:

$$G_{<\omega \cdot n + \omega}^\psi(x) = y \iff \exists p \forall q \geq p G_{\omega \cdot n + q}^\psi(x) = y.$$

But  $G_{\omega \cdot n + q}^\psi(x) = y$  is a  $\Delta_{2n+1}^0[\psi]$  predicate of  $x, y$  and  $q$  by the inductive hypothesis  $(\mathbf{b}'_n)$ ; it follows that  $G_{<\omega \cdot n + \omega}^\psi$  is  $\Sigma_{2n+2}^0[\psi]$ . Thus,  $(\mathbf{a}_{n+1})$  holds.

Now we tackle  $(\mathbf{b}'_{n+1})$ . This case is dealt with in a manner similar to case for  $(\mathbf{b}'_0)$ , using the inductive hypothesis to obtain both a  $\Pi_{2(n+1)}^0[\psi]$  and a  $\Sigma_{2(n+1)}^0[\psi]$  characterization of  $G_{\omega \cdot (n+1)}^\psi$ : it suffices then to replace  $(s)_0 \subset \psi$  in (1) and (2) by  $(s)_0 \subset G_{\omega \cdot (n+1)}^\psi$ . The latter is still  $\Sigma_{2n+2}^0$ , so this gives that  $G_{\omega \cdot (n+1) + q}^\psi(x) = y$  is  $\Delta_{2n+3}^0$ , as required.  $\blacksquare$

Having obtained an upper bound on the complexity of revision processes of length  $< \omega^2$  with recursive revision rules, we show that this bound is optimal. We first need to establish the following definitions and auxiliary results.

**4 DEFINITION.** A set  $A$  is *many-one reducible* to a set  $B$ , written  $A \leq_m B$ , if there is a total recursive function  $f$  such that for all  $x$ ,

$$x \in A \iff f(x) \in B.$$

In this case, we say that  $A \leq_m B$  via function  $f$ .<sup>4</sup>

**5 DEFINITION.** For any set  $A$ , we denote by  $A^{[y]}$  the “ $y$ -th section” of  $A$ , namely:

$$A^{[y]} = \{x : \langle x, y \rangle \in A\}.$$

This can be easily generalized to functions:

$$\phi^{[y]}(x) = \phi(\langle x, y \rangle).$$

In general, as already noted, we will tend not to distinguish between sets and functions. Clearly, any set can be regarded as a function; conversely, with any function  $\phi$  we can associate the set  $\{x : \phi(x) = 0\}$ .

We now define a sequence of sets of natural numbers that not only have higher and higher arithmetical complexity, but also turn out to be the most suitable to be represented in terms of Revision Rules.

**6 DEFINITION.** For any set  $A$ , let

$$\mathbf{Tot}^A = \{x : \{x\}^A \text{ is total}\}$$

$$\mathbf{Par}^A = \omega - \mathbf{Tot}^A.$$

Thus,  $\mathbf{Par}^A$  is the set of the indices of all functions that are *partial* (i.e., *non-total*) in  $A$ .<sup>5</sup>

Next, we define the sets  $\mathbf{Par}^{(n)}$  by induction on  $n$ :

$$\mathbf{Par}^{(0)} = \emptyset$$

$$\mathbf{Par}^{(n+1)} = \mathbf{Par}^{\mathbf{Par}^{(n)}}.$$

Notice that  $\mathbf{Par}^{(1)} = \mathbf{Par}^\emptyset = \omega - \mathbf{Tot}$ .

Before we can use the sets  $\mathbf{Par}^{(n)}$ , we have to investigate their properties, beginning with their level in the arithmetical hierarchy.

<sup>4</sup> This definition, as well as the following are standard in Recursion Theory. As a reference see Rogers [5] or Soare [6].

<sup>5</sup> There is a sense in which all functions are partial. This is not it.

**FIRST LEMMA.** For all  $n > 0$ ,  $\mathbf{Par}^{(n)}$  is  $\Sigma_{2n}^0$ -complete.

*Proof.* The case for  $n = 1$  is found in Soare [6], p. 66. The inductive step generalizes that proof. So assume that the First Lemma holds for  $n > 0$ . We want to prove it for  $n + 1$ . It is clear that  $\mathbf{Par}^{(n+1)}$  is  $\Sigma_{2n+2}^0$  (since  $x \in \mathbf{Par}^{(n+1)}$  iff “there is  $p$  such that for all  $q$ ,  $q$  is not a halting computation of  $\{x\}^{\mathbf{Par}^{(n)}}$  on input  $p$ ”). Let  $S \in \Sigma_{2(n+1)}^0$ . Then there is a  $\Sigma_{2n}^0$  predicate  $P$  such that

$$S(x) \iff \exists p \forall q P(p, q, x)$$

By inductive hypothesis there is a total recursive function  $h$  such that

$$P(p, q, x) \iff h(p, q, x) \in \mathbf{Par}^{(n)}.$$

Use the *s-m-n* Theorem to define a recursive  $f$  such that

$$\{f(x)\}^{\mathbf{Par}^{(n)}}(p) = \mu q \cdot h(p, q, x) \notin \mathbf{Par}^{(n)}.$$

Even though we are not going to need it, observe that function  $\{f(x)\}$  is  $\Sigma_1^0$  in the complement of  $\mathbf{Par}^{(n)}$ ; hence (by inductive hypothesis) it is  $\Sigma_{2n+1}^0$ . Moreover, the following equivalences hold:

$$\begin{aligned} S(x) &\iff \exists p \forall q P(p, q, x) \\ &\iff \exists p \forall q h(p, q, x) \in \mathbf{Par}^{(n)} \\ &\iff \exists p \{f(x)\}^{\mathbf{Par}^{(n)}}(p) \text{ is undefined} \\ &\iff f(x) \in \mathbf{Par}^{(n+1)}. \quad \blacksquare \end{aligned}$$

The main idea that we are going to use is a procedure that determines whether  $e$  is the index of a recursive function that is *total* (relative to some given oracle). We can describe it informally as follows. Given a function  $\{e\}$ , “to simulate — or, more simply, to run —  $\{e\}$  according to the consecutive strategy” means to start  $\{e\}$  on input 0; if and when this computation halts, start  $\{e\}$  on input 1, and so on. It is clearly possible to keep track of the total number of steps  $\{e\}$  carries out while we run it according to the consecutive strategy. Now suppose you want to find out whether some function  $\{e\}$  is total or not. Here’s a natural way to proceed: we run the consecutive emulation of  $\{e\}$  on inputs 0, 1, 2,  $\dots$ ; after

each step (i.e., execution of one instruction) of  $\{e\}$  on whatever input it is running on, we *look*: if the machine stops *precisely* at that moment, we output 1 and start it on the next argument; otherwise we just output 0 and let it compute for one more step. When you get to  $\omega$ , turn back, and look at the sequence of values you output: if  $\{e\}$  is not total, it entered some infinite computation, so that the sequence of values you output stabilizes to 0; but if  $\{e\}$  is total, both 1 and 0 will occur cofinally in the sequence. Thus, if  $\{e\}$  is not total,  $e$  is assigned value 0 at limit stages, while if it is total, the bootstrapper will be invoked. But if we are only looking at stabilization *before* or *at*  $\omega$ , this is enough to allow us to decide whether  $\{e\}$  is total.

The difficult part of this strategy is to keep track of the time, i.e, the total number of steps for which  $\{e\}$  has already been simulated according to the consecutive strategy. This is accomplished by means of particular functions  $\{\mathbf{e}_i\}$  (for  $i \in \omega$ ), which behave as follows: for any  $i$  let  $\{\mathbf{e}_i\}$  be a function that computes for one step and outputs 0 on  $x \leq i$ , and never converges if  $x > i$ . The index  $\mathbf{e}_i$  can be generated uniformly in  $i$ . We are now going to look for a uniform representation of all sets of the form  $\Sigma_{2n}^0$  for each  $n$ . In the next definition we define certain indices  $\mathbf{t}_n$  that in a sense “code” the sets  $\mathbf{Par}^{(n)}$ .

**7 DEFINITION.** Let  $\mathbf{p}$  be any index such that for any  $A$  and all  $x$  we have  $\{\mathbf{p}\}^A(x) \uparrow$ . We define indices  $\mathbf{t}_i$ , for  $0 \leq i$ , by induction on  $i$  as follows. First define  $\mathbf{t}_0$ :

$$\{\mathbf{t}_0\}^A(x) = \begin{cases} 0, & \text{if } \mathbf{p} \in A; \\ \uparrow, & \text{otherwise.} \end{cases}$$

Now, using the Recursion theorem, define  $\mathbf{t}_{n+1}$ :

$\{\mathbf{t}_{n+1}\}^A(x)$ : if both **1.**  $\mathbf{t}_{n+1} \in A$ ; and **2.**  $\mathbf{t}_n \notin A$  hold, then output 0; else be undefined.

We need to see that  $\mathbf{t}_n$  can be found uniformly (i.e., recursively) in  $n$ . Again, we proceed by induction on  $n$ . Clearly,  $\mathbf{t}_0$  is no problem. Assume there is a way to generate  $\mathbf{t}_1, \dots, \mathbf{t}_n$  uniformly. Let  $g$  be a function such that for any  $i$ :

$\{g(i)\}^A(x)$ : if both  $i \in A$  and  $\mathbf{t}_n \notin A$  hold, then output 0; else be undefined.

Function  $g$  exists and is total recursive by Church’s thesis. By the Recursion Theorem,  $g$  has a “fixed point”  $e$ , for which  $\{e\}$  and  $\{g(e)\}$  are functionally equivalent. We let such

an  $e$  be  $\mathbf{t}_{n+1}$ . Moreover, it is part of the claim of the Recursion Theorem that  $e$  can be found effectively. Although in general functional equivalence is undecidable, in the case in question it is possible to overcome this limitation since the output of  $g$  consists of (the code of) a specific set of instructions, whose only variable element is the argument  $i$  of  $g$ . Such argument is recoverable from  $g(i)$ .<sup>6</sup> Thus, in order to find the fixed point  $e$  we only have to enumerate  $g(0), g(1), \dots$  until we hit the first  $e$  such that  $\{g(e)\}$  queries the oracle on its own index.

**SECOND LEMMA.** For any  $n$ :

- (a)  $\{\mathbf{t}_n\}^{\mathbf{Par}^{(n)}}$  is not total;
- (b)  $\{\mathbf{t}_n\}^{\mathbf{Par}^{(n+1)}}$  is total.

*Proof.* We establish (a) and (b) simultaneously by induction on  $n$ .

**Case  $n = 0$ .** Let  $\mathbf{p}$  be as in the previous definition. Then we have:

$$\begin{aligned} \mathbf{p} \notin \emptyset &\iff \mathbf{p} \notin \mathbf{Par}^{(0)} \\ &\iff \{\mathbf{t}_0\}^{\mathbf{Par}^{(0)}}(x) \uparrow \text{ for any } x. \end{aligned}$$

Since obviously  $\mathbf{p} \notin \emptyset$ , we have that (a) holds. Moreover,

$$\begin{aligned} \mathbf{p} \in \mathbf{Par}^{(1)} &\iff \{\mathbf{t}_0\}^{\mathbf{Par}^{(1)}}(x) = 0 \text{ for any } x \\ &\iff \{\mathbf{t}_0\}^{\mathbf{Par}^{(1)}} \text{ is total.} \end{aligned}$$

But  $\mathbf{p} \in \mathbf{Par}^{(1)}$  by definition, so (b) holds as well.

**Case  $n = m + 1$ .** The inductive hypothesis for (a) gives that  $\{\mathbf{t}_m\}^{\mathbf{Par}^{(m)}}$  is not total. This means that  $\mathbf{t}_m \in \mathbf{Par}^{(m+1)}$ . So condition **2** of the program for  $\{\mathbf{t}_{m+1}\}$  (as given above) fails for  $\{\mathbf{t}_{m+1}\}^{\mathbf{Par}^{(m+1)}}$ , whence  $\{\mathbf{t}_{m+1}\}^{\mathbf{Par}^{(m+1)}}$  is not total. So the inductive step for (a) holds.

To establish the inductive case for (b) we must show that  $\{\mathbf{t}_{m+1}\}^{\mathbf{Par}^{(m+2)}}$  is total. It suffices to observe that the two conditions of the program for  $\{\mathbf{t}_{m+1}\}$  are met:

---

<sup>6</sup> I am here assuming that the coding is standard and that  $g$  is defined in the obvious way. Furthermore, notice that I am *not* claiming that  $i$  is recoverable from the index of any function equivalent to  $g(i)$ .

1. The inductive step for (a) — which we have already proved — now gives us that  $\{\mathbf{t}_{m+1}\}^{\mathbf{Par}^{(m+1)}}$  is not total. This means that  $\mathbf{t}_{m+1} \in \mathbf{Par}^{(m+2)}$ .
2. The inductive hypothesis gives that  $\{\mathbf{t}_m\}^{\mathbf{Par}^{(m+1)}}$  is total, i.e. that  $\mathbf{t}_m \notin \mathbf{Par}^{(m+2)}$ .

This completes the proof of the Second Lemma. ■

We are finally ready to establish our theorem, according to which it is possible to give a revision rule that, when iterated through  $\omega^2$  will in turn compute arithmetical sets of higher and higher arithmetical complexity, i.e., our sets  $\mathbf{Par}^{(n)}$ . As we announced, we will first impose certain constraints on the bootstrapper; these will be expressed in the form of an arithmetical predicate  $P(\alpha)$ . We will then show how these constraints can be lifted, provided we are willing to give up a stage-by-stage correspondence between the complexity of the set being represented and the length of the revision process necessary to represent it.

**8 THEOREM.** There are a recursive functional  $G$  and an arithmetical constraint  $P(\alpha)$  such that

$$\forall n \forall \psi (P(\psi) \implies \mathbf{Par}^{(n)} \leq_m G_{<\omega \cdot n}^\psi).$$

Moreover, this result is optimal, since  $G_{<\omega \cdot n}^\psi$  is itself  $\Sigma_{2n}^0$ .

*Proof.* The basic idea is to define a recursive functional  $G(x, \phi)$  with the understanding that in the revision process the argument  $\phi$  will be replaced by the previous revision stage. Essentially, this is the idea of the proof: we want  $G(x, G_{\omega \cdot n + p}^\psi)$  to “find out” the values of  $n$  and  $p$ , and then proceed as follows: if  $x$  is not of the form  $\langle n + 1, z \rangle$ , simply output  $G_{\omega \cdot n + p}^\psi(x)$ ; if on the other hand  $x = \langle n + 1, z \rangle$ , we run  $\{z\}^{\mathbf{Par}^{(n)}}$  on inputs  $0, 1, 2, \dots$ , for  $p + 1$  steps: if  $\{z\}^{\mathbf{Par}^{(n)}}$  halts precisely at the  $(p + 1)$ -st step we output 1, we else output 0. The crucial, and most difficult part is how to “find out”  $n, p$ . The reader is advised that the proof is long.

Let  $\mathbf{t}$  be index of some total function that computes for at least two steps on any argument  $x$ . Let  $\mathbf{t}_i$  be as before; i.e.,  $\{\mathbf{t}_n\}^{\mathbf{Par}^{(n)}}$  is everywhere undefined while  $\{\mathbf{t}_n\}^{\mathbf{Par}^{(n+1)}}$  is total. We now slightly modify the definition of  $\mathbf{e}_i$ , i.e.:

$$\{\mathbf{e}_i^A\}(x) = \begin{cases} 0, & \text{in 1 step, if } x < i; \\ \uparrow & \text{if } i \leq x. \end{cases}$$

Intuitively,  $\mathbf{t}_i$  and  $\mathbf{e}_i$  are what allows us to find out  $n$  and  $p$ . Suppose we are given as input some function  $\phi$  as the result of the previous stage of the revision process: we can determine whether  $\mathbf{Par}^{(n+1)}$  has already been computed by checking whether  $\mathbf{t}_i \in \phi$ . This gives us  $n$ . Likewise, we can determine  $p$  by checking if  $\phi^{[n]}$  has already discovered that  $\{\mathbf{e}_i\}$ .

Let  $P(\alpha)$  be the conjunction of the following conditions, which we suppose satisfied in the following:

$$\begin{aligned}\alpha(x) &> 0 \quad \text{for all } x; \\ \alpha(\langle i, \mathbf{t} \rangle) &= 2 \quad \text{for } i \geq 0; \\ \alpha(\langle i+1, \mathbf{t}_i \rangle) &= 1 \quad \text{for } i \geq 0; \\ \alpha(\langle 1, \mathbf{e}_m \rangle) &= 1 \quad \text{for } m \geq 0.\end{aligned}$$

We now define  $G(x, \phi)$  by cases as follows:

- 1.** If  $\phi(\langle 1, \mathbf{e}_0 \rangle) = 1$  then if  $x$  is not of the form  $\langle 1, z \rangle$  output  $\phi(x)$ ; if  $x = \langle 1, z \rangle$  then run  $\{z\}(0)$  for 1 step: output 1 if it halts after one step, and output 0 otherwise.
- 2.** If  $\phi(\langle 1, \mathbf{e}_0 \rangle) = 0$  then:
  - 2.1** find the least  $i > 0$  such that  $\phi(\langle i+1, \mathbf{t}_{i-1} \rangle) = 1$ ;
  - 2.2** if  $\phi(\langle i, \mathbf{t} \rangle) \leq 1$  and  $x = \langle i, z \rangle$  then find the least  $j$  such that  $\phi(\langle i, \mathbf{e}_j \rangle) = 1$ ; run the consecutive simulation of  $\{z\}$  on  $0, 1, 2, \dots$  with oracle  $\phi^{[i-1]}$  if  $i > 1$  and empty oracle if  $i = 1$ , for a total of  $j+1$  steps. If at the  $(j+1)$ -st step there is a convergence, output 1, else output 0; if  $x$  is not of the form  $\langle i, z \rangle$  then output  $\phi(x)$ .
  - 2.3** if  $\phi(\langle i, \mathbf{t} \rangle) = 2$  and  $x = \langle i+1, z \rangle$ , then run  $\{z\}$  on argument 0 with oracle  $\phi^{[i]}$  for 1 step: if it halts after one step output 1, else output 0; if  $x$  is not of the form  $\langle i+1, z \rangle$  then output  $\phi(x)$ .

The conditions of the cases **1**, **2.2**, and **2.3** will serve to identify the argument  $\phi$  as the bootstrapper  $\psi$ , the function  $G_{\omega \cdot n + p + 1}^\psi$ , and the function  $G_{\omega \cdot (n+1)}^\psi$  respectively.

We show that  $G(x, \phi)$ , defined in this way, meets the following desiderata. The first three show  $G$  to be recursive (in that all its unbounded searches succeed), and the last one

establishes the theorem.

$$\begin{aligned}
\mathbf{d1} \quad n > 0 \vee p > 0 &\implies G_{\omega \cdot n + p}^\psi(\langle i + 1, \mathbf{t}_i \rangle) = \begin{cases} 0, & \text{if } i \leq n \text{ and } p > 0; \\ 0, & \text{if } i < n \text{ and } p = 0; \\ 1, & \text{if } i = n \text{ and } p = 0; \\ 1, & \text{if } i = n + 1 \text{ and } p > 0; \end{cases} \\
\mathbf{d2} \quad n = 0 \vee p > 0 &\implies G_{\omega \cdot n + p}^\psi(\langle n + 1, \mathbf{e}_m \rangle) = \begin{cases} 0, & \text{if } m < p; \\ 1, & \text{if } m \geq p; \end{cases} \\
\mathbf{d3} \quad i \neq n + 1 \wedge q \leq p &\implies G_{\omega \cdot n + q}^\psi(\langle i, x \rangle) = G_{\omega \cdot n + p}^\psi(\langle i, x \rangle); \\
\mathbf{d4} &\quad \{z : G_{<\omega \cdot n}^\psi(\langle n, z \rangle) = 0\} = \mathbf{Par}^{(n)}.
\end{aligned}$$

Observe, once and for all, that **d3**, as it were, has an “upward” as well as a “downward” direction. It tells us, first, that all of the arguments already dealt with by  $G$  are preserved at later stages (a form of monotony), and also that arguments of the form  $\langle n + 1, x \rangle$  are not dealt with before stage  $n$ , so that their values are to some extent also preserved “downward.”

Desiderata **d1–d4** are established simultaneously by induction on the lexicographic ordering of the pairs  $\langle n, p \rangle$ . So fix  $n$  and  $p$ , and assume **d1–d4** hold for all  $m$  and  $q$  such that either  $m < n$  or  $m = n$  and  $q < p$ . We distinguish three main cases, according as  $n = p = 0$ , or  $p > 0$ , or  $n > p = 0$ .

**CASE**  $n = p = 0$ . To establish **d1** consider

$$G_0^\psi(\langle i + 1, \mathbf{t}_i \rangle) = \psi(\langle i + 1, \mathbf{t}_i \rangle);$$

then **d1** holds vacuously. Moreover, **d2** holds by choice of  $\psi$  as well:

$$G_0^\psi(\langle 1, \mathbf{e}_m \rangle) = \psi(\langle 1, \mathbf{e}_m \rangle) = 1.$$

Finally, **d3** is obvious in this case, while **d4** can be easily established since

$$\{x : G_{<0}^\psi(\langle 0, x \rangle) = 0\} = \emptyset = \mathbf{Par}^{(0)}.$$

**CASE**  $p > 0$ . For **d1**, we are looking at

$$G_{\omega \cdot n + p}^\psi(\langle i + 1, \mathbf{t}_i \rangle) = G(\langle i + 1, \mathbf{t}_i \rangle, G_{\omega \cdot n + p - 1}^\psi).$$

If  $p - 1 = 0$  and  $n = 0$  then case **1** of the definition of  $G$  applies, and we run  $\{\mathbf{t}_0\}^\emptyset(0)$  for 1 step; this function never halts, so we output 0 as required by the first case of **d1**; if  $i > 0$  then  $i + 1 > 1$ , so we output

$$G_{\omega \cdot n + p - 1}^\psi(\langle i + 1, \mathbf{t}_i \rangle) = \psi(\langle i + 1, \mathbf{t}_i \rangle) = 1$$

by choice of  $\psi$ , as required by the fourth case of **d1**.

So suppose  $p > 1$  or  $n > 0$ . If  $p > 1$  then by the inductive hypothesis on **d2**

$$G_{\omega \cdot n + p - 1}^\psi(\langle 1, \mathbf{e}_0 \rangle) = 0,$$

so case **2** applies. By the inductive hypothesis on **d1** the first  $i$  such that

$$G_{\omega \cdot n + p - 1}^\psi(\langle i + 1, \mathbf{t}_i \rangle) = 1$$

is  $n + 1$ . By the inductive hypothesis on **d2** the first  $j$  such that

$$G_{\omega \cdot n + p - 1}^\psi(\langle n + 1, \mathbf{e}_j \rangle) = 1$$

is  $p - 1$ . So we run  $\{\mathbf{t}_n\}$  for  $p$  steps with oracle  $(G_{\omega \cdot n + p - 1}^\psi)^{[n]}$ . By the inductive hypothesis on **d4** this is the same as **Par**<sup>(n)</sup>. Function  $\{\mathbf{t}_n\}$  with such an oracle is then empty and we output 0 as required by the first case of **d1** (the first result for  $i < n$  is given by the inductive hypothesis on **d1** and **d3**). If  $i = n + 1$  then, as required by the fourth case of **d1**, we output

$$G_{\omega \cdot n + p - 1}^\psi(\langle n + 2, \mathbf{t}_{n+1} \rangle),$$

which is 1 by choice of  $\psi$  and the fact that this value is preserved (inductive hypothesis on **d3**).

If  $n > 0$  but  $p = 1$  then again the inductive hypothesis on **d2** tells us that case **2** applies. By the inductive hypothesis on **d1** the first  $i$  such that  $G_{\omega \cdot n}^\psi(\langle i + 1, \mathbf{t}_i \rangle) = 1$  is  $n$ . Moreover,

$$G_{\omega \cdot n}^\psi(\langle n, \mathbf{t} \rangle) = \lim_{k \rightarrow \omega} G_{\omega \cdot (n-1) + k}^\psi(\langle n, \mathbf{t} \rangle) = 2,$$

since these values keep oscillating between 0 and 1. So case **2.3** applies, and we run  $\{\mathbf{t}_n\}$  on 0 for 1 step with oracle  $(G_{\omega \cdot n}^\psi)^{[n]} = \mathbf{Par}^{(n)}$  (by the inductive hypothesis on **d4**). This

function with such an oracle is then empty, and we output zero as required by the first case of **d1** (the case for  $i < n$  is taken care of by the inductive hypothesis as before). On the other hand, if  $i = n + 1$ , reasoning as before we see that  $G$  outputs 1 as required by the fourth case of **d1**. This establishes **d1**.

For **d2** we are looking at

$$G_{\omega \cdot n + p}^{\psi}(\langle n + 1, \mathbf{e}_m \rangle) = G(\langle n + 1, \mathbf{e}_m \rangle, G_{\omega \cdot n + p - 1}^{\psi}).$$

We distinguish the same subcases as for **d1**. If  $p - 1 = 0$  and  $n = 0$  then  $G_{\omega \cdot n + p - 1}^{\psi} = \psi$  so case **1** applies. We run  $\{\mathbf{e}_m\}(0)$  for 1 step. If  $m = 0$  this never halts and we output 0 as required; if  $m \geq 1 = p$  this halts immediately, so we output 1 as required.

So suppose  $p > 1$  or  $n > 0$ . Reasoning as in the corresponding case for **d1**, we see that we run  $\{\mathbf{e}_m\}$  on  $0, 1, 2, \dots$ , for  $p$  steps with  $\mathbf{Par}^{(n)}$  as an oracle: this never halts if  $m < p$ , so that we output 0, while it halts immediately if  $m \geq p$ , so that we output 1 as required. Therefore **d2** holds. Finally, **d3** and **d4** are clear from the inductive hypothesis and the definition of  $G$ : here is where we use the fact that  $0 \notin \text{rng}(\psi)$ , which ensures that any calls  $G$  might make to  $G_{\omega \cdot n}^{\psi}$  return the correct result.

**CASE**  $n > p = 0$ . For **d1** we must show that

$$\begin{aligned} G_{\omega \cdot n}^{\psi}(\langle i + 1, \mathbf{t}_i \rangle) &= \lim_{k \rightarrow \omega} G_{\omega \cdot (n-1) + k}^{\psi}(\langle i + 1, \mathbf{t}_i \rangle) \\ &= \begin{cases} 0, & \text{if } i < n; \\ 1, & \text{if } i = n. \end{cases} \end{aligned}$$

But this follows from the inductive hypothesis, since if  $i < n$  then  $i \leq (n - 1)$ , and we only need apply the inductive hypothesis on the first case of **d1** to see that at the limit we obtain 0; on the other hand, if  $i = n$  then  $i = (n - 1) + 1$ , so that by the same token we obtain 1 at the limit.

Finally, we observe that **d2** and **d3** hold vacuously, and that the case for **d4** is clear from the definition of  $G$  and the inductive hypothesis. This completes the proof of the theorem. ■

**COROLLARY.** The set of (the codes of) the true sentences of first order arithmetic is many-one reducible to  $G_{< \omega^2}^{\psi}$ . ■

We now proceed to generalize this result. As mentioned, if we are willing to give up a precise stage-by-stage correspondence between the complexity of the sets we compute and the length of the revision process necessary to compute them, then we can drop the restriction of the bootstrapper.

The following theorem is due to Anil Gupta (personal communication). The proof given here is a simplification of his original proof. The basic idea is to let the computed set lag  $\omega$  cycles behind the revision process. This gives the revision process enough “time” to compute both  $\mathbf{Par}^{(n)}$  and its complement. This also shows that unless there are constraints on the bootstrapper, we cannot have a stage-by-stage correspondence between the revision process and the complexity of the computed set.

**9 THEOREM.** (Gupta) There is a recursive functional  $G$  such that

$$\forall n \forall \psi [\mathbf{Par}^{(n)} \leq_m G_{<\omega \cdot (n+1)}^\psi].$$

*Proof.* For any  $\phi \in \omega^\omega$ , let

$$\phi^{[n]} = \begin{cases} \emptyset, & \text{if } n = 0; \\ \{e : \phi(\langle 1, \langle n-1, e \rangle \rangle) = 0\}, & \text{otherwise.} \end{cases}$$

Then we can define the functional  $G(x, \phi)$  by cases as follows (recall that in the revision process the function argument will be replaced by the function obtained at the previous stage). So let:

$$\begin{aligned} G(\langle 0, 0 \rangle \phi) &= \phi(\langle 0, 0 \rangle) + 1; \\ G(\langle 1, \langle n, e, x \rangle \rangle, \phi) &= \begin{cases} 1, & \text{if } \{e\}^{\phi^{[n]}}(x) \downarrow^r, \text{ and } r \leq \phi(\langle 0, 0 \rangle); \\ 0, & \text{otherwise;} \end{cases} \\ G(\langle 2, \langle n, e \rangle \rangle, \phi) &= \begin{cases} 1, & \text{if } (\forall x \leq \phi(\langle 0, 0 \rangle)) \cdot \phi(\langle 1, \langle n, e, x \rangle \rangle) > 0; \\ 0, & \text{otherwise;} \end{cases} \\ G(x, \phi) &= \phi(x), \text{ if } x \text{ has any other form.} \end{aligned}$$

Now let  $\psi$  be any bootstrapper. We need to show that for any  $n > 0$  and  $\sigma$  such that  $\omega^2 > \sigma \geq \omega \cdot (n+1)$ ,

$$\mathbf{Par}^{(n)} = \{e : G_\sigma^\psi(\langle 2, \langle n-1, e \rangle \rangle) = 0\},$$

whence the theorem follows. In turn, this follows from the following facts, and in particular from **5.**:

- 0.**  $G_{\omega \cdot n + p}^{\psi}(\langle 0, 0 \rangle) \geq p$ ;
- 1.** if  $\sigma \geq \omega \cdot n$  then:  $\{e\}^{\mathbf{Par}^{(n)}}(x) \downarrow^r$  if and only if  $\exists p \forall q \geq p \cdot G_{\sigma+q}^{\psi}(\langle 1, \langle n, e, x \rangle \rangle) = r$ ;
- 2.** if  $\sigma > \omega \cdot n$  and  $\{e\}^{\mathbf{Par}^{(n)}}(x) \uparrow$  then  $G_{\sigma}^{\psi}(\langle 1, \langle n, e, x \rangle \rangle) = 0$ ;
- 3.** if  $\sigma \geq \omega \cdot n$  and  $e \in \mathbf{Par}^{(n+1)}$  then  $\exists p \forall q \geq p \cdot G_{\sigma+q}^{\psi}(\langle 2, \langle n, e \rangle \rangle) = 0$ ;
- 4.** if  $\sigma > \omega \cdot (n+1)$  and  $e \notin \mathbf{Par}^{(n+1)}$  then  $G_{\sigma+q}^{\psi}(\langle 2, \langle n, e \rangle \rangle) = 1$ ;
- 5.** if  $\sigma > \omega \cdot (n+1)$  and  $n > 0$  then:  $G_{\sigma+q}^{\psi}(\langle 2, \langle n-1, e \rangle \rangle) = 0$  if and only if  $e \in \mathbf{Par}^{(n)}$ .

**Fact 0.** is easily established. Observe that  $G_{\omega \cdot n + p}^{\psi}(\langle 0, 0 \rangle)$  acts as a “counter,” in that it starts out with value  $\psi(\langle 0, 0 \rangle)$ , then it increases strictly monotonically as a function of  $p$ , it is “reset” to  $\psi(\langle 0, 0 \rangle)$  at  $\omega$ , then it increases again, and so on. More precisely,

$$G_{\omega \cdot n + p}^{\psi}(\langle 0, 0 \rangle) = \psi(\langle 0, 0 \rangle) + p.$$

We now establish the remaining facts **1.–5.** by simultaneous induction on  $\sigma$ .

**CASE:**  $\sigma = 0$ . For **1.**, suppose  $\{e\}(x) \downarrow^r$ : then  $G$  simulates  $\{e\}$  with empty oracle for the number of steps given by the counter; as soon as this counter becomes  $\geq r$ , we have  $G_q^{\psi}(\langle 1, \langle 0, e, x \rangle \rangle) = r$ . Since the converse holds too, **1.** follows.

Facts **2.**, **4.** and **5.** hold vacuously when  $\sigma = 0$ . So we take up fact **3.**: suppose  $e \in \mathbf{Par}^{(1)}$ , and let  $x$  be the smallest argument on which  $\{e\}(x) \uparrow$ . Then as soon as  $G_q^{\psi}(\langle 1, \langle 0, e, x \rangle \rangle)$  becomes constant  $= 0$ , so does  $G_q^{\psi}(\langle 2, \langle n, e \rangle \rangle) = 0$ .

**CASE:**  $\sigma$  limit. Fact **1.** is perfectly parallel to the case for  $\sigma = 0$ , using the inductive hypothesis on **5.** to show that the right oracle is used in the simulation of  $\{e\}$ . Facts **2.** and **4.** follow immediately from the inductive hypothesis. As for  $\sigma = 0$ , fact **3.** follows from **1.**, and **5.** follows from **3.**

**CASE:**  $\sigma = \tau + 1$ . Fact **1.** is easily established using the inductive hypothesis on **5.** that tells us that the right oracle  $(G_{\tau}^{\psi})^{[n]} = \mathbf{Par}^{(n)}$  is used in the simulation of  $\{e\}$ . For fact **2.**, suppose  $\{e\}^{\mathbf{Par}^{(n)}}(x) \uparrow$ ; then, given that  $\{e\}$  is simulated with the correct oracle, we always have value 0, no matter how large the value of the counter.

Fact **3.** follows from **2.:** if  $e \in \mathbf{Par}^{(n+1)}$ , let  $x$  be the first argument on which  $\{e\}^{\mathbf{Par}^{(n)}}$  diverges; then as soon as the counter is at least as large as  $x$ , we have  $G_{\sigma+q}^{\psi}(\langle 1, \langle n, e, x \rangle \rangle) = 0$ , whence also  $G_{\sigma+q}^{\psi}(\langle 2, \langle n, e \rangle \rangle) = 0$ , as desired.

Similarly, fact **4.** follows from **2.** Notice that now we have to “go  $\omega$  cycles up” in order to obtain the right result. If  $e \notin \mathbf{Par}^{(n+1)}$ , we have to wait for all computations of  $\{e\}^{\mathbf{Par}^{(n)}}$  to be terminated before we have enough evidence to assign  $\langle 2, \langle n, e \rangle \rangle$  value 1. Finally, **5.** follows from **3.** and **4.** ■

This completes our analysis of the arithmetical hierarchy. We have succeeded in giving a Rule of Revision, whose iteration through  $\omega^2$  computes increasingly complex arithmetical sets. In so doing we have, first, laid certain constraints on the allowable bootstrappers. These constraints are elementary, however, compared to the complexity of the sets we represent. Then we have shown how even these constraints can be avoided, provided we give up a precise correspondence between the ordinal length of the revision process and the complexity of the set it computes.

**ACKNOWLEDGEMENTS** I thank Nuel Belnap, Ken Manders, Rick Statman, and Jamie Tappenden for much helpful interaction and an anonymous referee for sound advice. A version of this paper was presented at the 1993 annual meeting of the Association for Symbolic Logic held at the University of Notre Dame.

## REFERENCES

- [1] Belnap, N. D., *Gupta's Rule of Revision Theory of Truth*, **Journal of Philosophical Logic**, 11 (1982) pp. 103-16.
- [2] Gupta, A., *Truth and Paradox*, **Journal of Philosophical Logic**, 11 (1982) pp. 1-60.

- [3] Gupta, A. & Belnap, N. D., **The Revision Theory of Truth**, MIT Press, Cambridge, MA, 1993, XII–299 pp.
- [4] Herzberger, H.G., *Notes on Naive Semantics*, **Journal of Philosophical Logic**, 11 (1982) pp. 61-102.
- [5] Rogers, H., **Theory of Recursive Functions and Effective Computability**, McGraw-Hill, New York, NY, 1967, XIV–482 pp.
- [6] Soare, R. I., **Recursively Enumerable Sets and Degrees**, Springer-Verlag, Berlin, Heidelberg, New York, 1978, 480 pp.