Extending Kleene's $\mathcal{O}$ Using Infinite Time Turing Machines, *or* *How With Time She Grew Taller and Fatter.*

MSc Thesis (*Afstudeerscriptie*)

written by

Ansten Mørch Klev
(born October 7th, 1982 in Lørenskog, Norway)

under the supervision of **Joel David Hamkins** and **Benedikt Löwe**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam*.

Date of the public defense:	**Members of the Thesis Committee:**
August 14th, 2007	Joel David Hamkins
	Benedikt Löwe
	Dick de Jongh
	Peter van Emde Boas

Institute for Logic, Language and Computation

Abstract

We define two successive extensions of Kleene's $\mathcal{O}$ using infinite time Turing machines. The first extension, $\mathcal{O}^+$, is proved to code a tree of height λ, the supremum of the writable ordinals, while the second extension, $\mathcal{O}^{++}$, is proved to code a tree of height ζ, the supremum of the eventually writable ordinals. Furthermore, we show that $\mathcal{O}^+$ is computably isomorphic to h, the lightface halting problem of infinite time Turing machine computability, and that $\mathcal{O}^{++}$ is computably isomorphic to s, the set of programs that eventually write a real. The last of these results implies, by work of Welch, that $\mathcal{O}^{++}$ is computably isomorphic to the Σ_2 theory of $\mathbf{L}_\zeta$, and, by work of Burgess, that $\mathcal{O}^{++}$ is complete with respect to the class of the arithmetically quasi-inductive sets. This leads us to conjecture the existence of a parallel of hyperarithmetic theory at the level of $\Sigma_2(\mathbf{L}_\zeta)$, a theory in which $\mathcal{O}^{++}$ plays the role of $\mathcal{O}$, the arithmetically quasi-inductive sets play the role of Π^1_1, and the eventually writable reals play the role of Δ^1_1.

Acknowledgements

I would like to warmly thank Joel Hamkins for suggesting to me the idea of doing Kleene's $\mathcal{O}$ in the infinite time Turing machine setting; for providing guidance and ideas – many of the results reported below were anticipated by him; for reading and commenting extensively on several earlier drafts of this thesis; finally, but not least, his great interest in my work all along the way, and his vitality and passion for mathematics, was an indispensable source of inspiration. Further, I would like to thank Benedikt Löwe for letting me change thesis topic at a late stage; for providing guidance in my individual study projects during my time at the ILLC; for spawning my interest in infinite time Turing machines, and for convincing me to come to Amsterdam. I would also like to thank Dick de Jongh for wanting to sit on my thesis committee, and for providing support as my academic mentor. Finally, I want to thank Peter Koepke and the Hausdorff Center for Mathematics in Bonn for inviting me to Bonn International Workshop for Ordinal Computability.

On a more personal side, I want to share with the reader my gratefulness to all the wonderful people I have met so far during my stay in Amsterdam, and I want to specially mention three persons that meant a lot to me during the writing of this thesis: Herman for continuing friendship, David for food and brotherhood, and Mari for rivers deep and mountains high.

"There is more behind and inside V. than any of us had suspected. Not who, but what: what is she. God grant that I may never be called upon to write the answer, either here or in any official report."
Journal note made by Sidney Stencil in Florence, April, 1899.
– Thomas Pynchon, *V.*

Introduction. Kleene's $\mathcal{O}$ is a subset of the set of natural numbers coding a tree of height ω_1^{CK}. The tree coded by Kleene's $\mathcal{O}$ is built up inductively: at successor stages of the construction one adds an element to each top node of the tree constructed so far. Then, at limit stages one checks whether there are Turing computable functions cofinal in the branches of the tree constructed so far; if the function $\{e\}$ is seen to be thus cofinal, a suitable natural number code for this function is put on top of the branch in which $\{e\}$ is cofinal. The final result of this procedure is a tree to the top of which no Turing computable function can climb.

The infinite time Turing machine, introduced by Hamkins and Lewis in their [5], extends the action of Turing machines to transfinite time, and these machines naturally give rise to two different notions of computability, that of ITTM computability and that of eventual computability. The main point of this thesis is to investigate what sort of set we get if we let either ITTM computable functions or eventually computable functions do the work of Turing computable functions in the definition of Kleene's $\mathcal{O}$. We will call the set thus obtained by the use of ITTM computable functions $\mathcal{O}^+$ and the set thus obtained by the use of eventually computable functions $\mathcal{O}^{++}$.

The results of our investigations are surprisingly natural: the set $\mathcal{O}^+$ codes a tree which is taller and fatter than that coded by $\mathcal{O}$, while the set $\mathcal{O}^{++}$ codes a tree which is taller and fatter than that coded by $\mathcal{O}^+$; and just as the tree coded by $\mathcal{O}$ has height equal to ω_1^{CK} the supremum of the Turing computable ordinals, so does the tree coded by $\mathcal{O}^+$ have height equal to λ, the supremum of the ITTM computable ordinals, and the tree coded by $\mathcal{O}^{++}$ height equal to ζ, the supremum of the eventually computable ordinals.

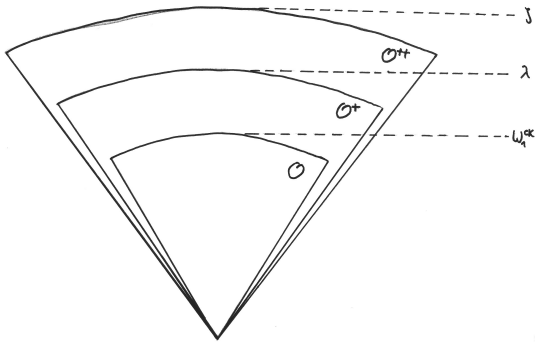

Figure 1: Portrait of $\mathcal{O}$ With Extensions

The thesis is organized as follows. All the necessary preliminaries for our study are given in the first chapter. In the second chapter we study the set $\mathcal{O}^+$. In the third chapter we introduce the class of eventually computable functions and, thereafter, study the set $\mathcal{O}^{++}$. At present there is a much richer theory surrounding $\mathcal{O}^{++}$ than there is surrounding $\mathcal{O}^+$; for that reason, Chapter 3 is considerably longer than Chapter 2.

Chapter 1

Preliminaries

In principle, anyone who is confident with the nuts and bolts of higher mathematics and who has some basic knowledge of mathematical logic and computability theory, and in particular, who knows what a Turing machine is, should be able to follow this thesis. No previous knowledge of Kleene's $\mathcal{O}$ or infinite time Turing machines is supposed.

This first chapter is organized as follows. We first fix notation and introduce some notions which are standards of mathematical logic. We then review the few results of computability theory that will be of use to us. Thereafter we give the definition of Kleene's $\mathcal{O}$; having done that, we can state more precisely the idea behind the definition of $\mathcal{O}^+$ and $\mathcal{O}^{++}$. Finally, we give a crash course into the theory of infinite time Turing machines.

1.1 Preliminary Preliminaries

Notation. Let $\mathbb{N}$ be the set of natural numbers, and $2^{\mathbb{N}}$ the set of infinite binary strings. Elements of $2^{\mathbb{N}}$ are also called *reals*. Throughout we let the letters a, b, and c, and their starred variations range over reals. The letters n, m, k, and ℓ range over natural numbers, as do e and d, but these latter letters will only be used as indices of functions. The letters A and B will sometimes be used to range over subsets of $2^{\mathbb{N}}$ and other times to range over subsets of $\mathbb{N}$, and a few times they will even range over arbitrary sets. The context will always make it clear which of these three cases we are in. The letters u, v, and w will range over $\mathbf{V}$, the universe of sets, which is also the universe within which this thesis is set. For any set A, we let $\wp(A)$ denote the power set of A.

If F is a total function from the set A to the set B, then we write $F\colon A \to B$. If F is a partial function from A to B, then we write $F\colon A \rightharpoonup B$. For a partial function $F\colon A \rightharpoonup B$, the domain of F is denoted by $\mathrm{dom}(F)$ and the range of F is denoted by $\mathrm{ran}(F)$. Given two partial functions F and G, we write $F(v) \simeq G(w)$ to mean that either $v \notin \mathrm{dom}(F)$ and $w \notin \mathrm{dom}(G)$, or both $v \in \mathrm{dom}(F)$ and $w \in \mathrm{dom}(G)$ and $F(v) = G(w)$. Extending this, we write $F \simeq G$ to mean that $\mathrm{dom}(F) = \mathrm{dom}(G)$, and $F(v) \simeq G(v)$ for all $v \in \mathrm{dom}(F)$. If $F\colon A \rightharpoonup B$ and $G\colon B \rightharpoonup C$, then $F \circ G$ denotes the composition of F and G, that is, the function $H\colon A \rightharpoonup C$ such that $H(v) \simeq F(G(v))$. For a function F with $A \subseteq \mathrm{dom}(F)$ we use the notation $F[A]$ for the direct image of A under F, that is, $F[A] := \{F(v) : v \in A\}$; if $A \subseteq \mathrm{ran}(F)$, then $F^{-1}[A]$ denotes the inverse image of A under F, that is, $F^{-1}[A] := \{v : F(v) \in A\}$.

At certain points we will utilize Knuth's up-arrow notation. We define

$n{\uparrow\uparrow}0 = 1$, and

$$n{\uparrow\uparrow}k := \underbrace{n^{n^{\cdot^{\cdot^{\cdot^{n}}}}}}_{k \text{ times}}$$

We suppose the reader is familiar with the notion of an ordinal. Throughout, we denote ordinals by small Greek letters α, β, η, *etc.* Unless otherwise noted, the symbol $\leq$ will denote the relation of less than or equal on the class of ordinals.

If $S = \{A_\alpha\}_{\alpha<\eta}$ is a sequence, then B is said to be *final* in S if there is some $\beta < \eta$ such that $A_\alpha = B$ for all α with $\beta < \alpha < \eta$.

Definability theory. Let us recall some notions from definability theory (for more details see, for instance, [8]). Let L be a two-sorted first-order language over the signature $\{+, \times, \leq, 0, 1\}$. The variables of L are either of the form $x, y, \ldots$, or of the form $\mathsf{f}, \mathsf{g}, \ldots$. We interpret L over the structure $\mathfrak{N} := (\mathbb{N} \cup 2^{\mathbb{N}}, +, \times, \leq, 0, 1)$, the standard structure of arithmetic. The variables $x, y, \ldots$ range over $\mathbb{N}$, while the variables $\mathsf{f}, \mathsf{g}, \ldots$ range over $2^{\mathbb{N}}$. A number quantifier $\forall x$ ($\exists x$) is said to be bounded if all occurrences of x within its scope are in contexts $x \leq t$ where t is any term not containing x. A formula of L is said to be Δ_0 if all its quantifiers are number quantifiers and these are all bounded. A formula is Σ^0_1 (Π^0_1) if it is of the form $\exists x \varphi$ ($\forall x \varphi$) where φ is Δ_0; it is Σ^0_{n+1} (Π^0_{n+1}) if it is of the form $\exists x \varphi$ ($\forall x \varphi$) where φ is Π^0_n (Σ^0_n). Further, a formula is Σ^1_1 (Π^1_1) if it is of the form $\exists \mathsf{f} \varphi$ ($\forall \mathsf{f} \varphi$), where φ is Π^0_1 (Σ^0_1); it is Σ^1_{n+1} (Π^1_{n+1}) if it is of the form $\exists \mathsf{f} \varphi$ ($\forall \mathsf{f} \varphi$) where φ is Π^1_n (Σ^1_n).

A set $A \subseteq \mathbb{N}^k \times (2^{\mathbb{N}})^\ell$ is said to be Σ^i_n (Π^i_n) if it is definable by a Σ^i_n formula (Π^i_n formula), that is, if there is a Σ^i_n formula (Π^i_n formula) $\varphi(\vec{x}, \vec{\mathsf{f}})$ such that

$$(\vec{m}, \vec{a}) \in A \Leftrightarrow \mathfrak{N} \models \varphi[\vec{m}, \vec{a}].$$

The set A is said to be arithmetical if it is Σ^0_n for some n, and it is said to be Δ^i_n if it is both Σ^i_n and Π^i_n. It is then a basic theorem, whose proof we omit here, that for any set $A \subseteq \mathbb{N}^k \times (2^{\mathbb{N}})^\ell$ which is definable using some formula of L, there is some least i and n such that A is Σ^i_n and/or Π^i_n. Furthermore, it is basic that we have $\Sigma^i_n \subsetneq \Sigma^i_{n+1}$, $\Sigma^i_n \not\subseteq \Delta^i_n$, and $\Sigma^0_n \subsetneq \Delta^1_1$; hence the Σ^i_n and Π^i_n classes form a real hierarchy of sets, and we think of higher i and n as meaning higher complexity. For any class $\mathcal{A}$ of sets, a set $A \subseteq 2^{\mathbb{N}}$ is said to be complete with respect to the class $\mathcal{A}$ if A is in $\mathcal{A}$ and if for every set B in $\mathcal{A}$, there is a Σ^0_1 definable total function $F\colon 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ such that $B = F^{-1}[A]$.

Coding matters. Via the correspondence

$$n \leftrightsquigarrow \overbrace{\langle 1, \ldots, 1\rangle}^{n \text{ times}} {}^\frown \langle 0, 0, \ldots\rangle,$$

we may think of $\mathbb{N}$ as sitting inside $2^{\mathbb{N}}$. In fact, there is a Δ_0 definable bijection $\ulcorner \cdot \urcorner$ such that for any $k, \ell \in \mathbb{N}$, we have $(\vec{n}, \vec{a}) \in \mathbb{N}^k \times (2^{\mathbb{N}})^\ell$ if and only if $\ulcorner \vec{n}, \vec{a} \urcorner \in 2^{\mathbb{N}}$. Note, moreover, that a real is just the characteristic function of a set $A \subseteq \mathbb{N}$, and we often suppress the difference between a subset of $\mathbb{N}$ and its characterstic function, writing, for instance, $n \in a$ instead of $a(n) = 1$. We let

$\langle\cdot\rangle$ be a Δ_0 definable coding of $\mathbb{N}^{<\omega}$, the set of finite strings of natural numbers. Thus, via $\langle\cdot\rangle$ any relation on $\mathbb{N}$ can be thought of as a real. We define

$$\text{field}(a) := \{n \in \mathbb{N} : \exists m(\langle m, n\rangle \in a \vee \langle n, m\rangle \in a\}.$$

Thus, if a codes a binary relation, then $\text{field}(a)$ is simply the field of that relation. If a codes a partial order, then we write $\leq_a$ or $<_a$ for this order, depending on whether or not we have the strict relation in mind. If a codes a well-founded relation, then any $n \in \text{field}(a)$ can be assigned a norm $|\cdot|_a$ which is defined by

$$|n|_a := \sup\{|k|_a + 1 : k <_a n\}.$$

In this case, we let $|a| := \sup\{|n|_a + 1 : n \in \text{field}(a)\}$. We let $\text{WO} \subseteq 2^{\mathbb{N}}$ be the set of reals coding well-orders. Implicitly, a real $a \in \text{WO}$ also codes a countable ordinal, namely the ordinal $|a|$.

Inductive Definitions. We recall some basics of the theory of inductive definitions over $\mathfrak{N}$ (for more details see, for instance, [8]). A sequence $S = \{A_\alpha\}_{\alpha<\omega_1}$ of sets $A_\alpha \subseteq \mathbb{N}$ such that $A_0 = \emptyset$ and such that $\alpha < \beta$ implies $A_\alpha \subseteq A_\beta$ is called an inductive definition. The set $B \subseteq \mathbb{N}$ which is final in the sequence S is called the fixed-point of the inductive definition S. A function $\Gamma\colon \wp(\mathbb{N}) \to \wp(\mathbb{N})$ which is monotone with respect to the $\subseteq$-relation induces an inductive definition $\{\Gamma^\alpha\}_{\alpha<\omega_1}$ as follows

$$\begin{array}{lcll} \Gamma^0 & := & \emptyset & \\ \Gamma^{\alpha+1} & := & \Gamma(\Gamma^\alpha) & \\ \Gamma^\eta & := & \bigcup_{\alpha<\eta}\Gamma^\alpha & \text{for limit ordinals } \eta. \end{array}$$

In this case we denote by Γ^∞ the fixed-point of the inductive definition $\{\Gamma^\alpha\}_{\alpha<\omega_1}$. One can quite easily prove that the set Γ^∞ is the least fixed-point of the function Γ, that is, $\Gamma(\Gamma^\infty) = \Gamma^\infty$ and if $A \subseteq \mathbb{N}$ is such that $\Gamma(A) = A$, then $\Gamma^\infty \subseteq A$. Indeed, one can prove that if A satisfies $\Gamma(A) \subseteq A$, then $\Gamma^\infty \subseteq A$. If $\varphi(x, \mathsf{f})$ is an L-formula in which f occurs only positively, then $\varphi(x, \mathsf{f})$ defines a monotone function Γ on $\wp(A)$ by

$$\Gamma(A) := \{n \in \mathbb{N} : \mathfrak{N} \models \varphi[n, A]\}.$$

If $\varphi(x, \mathsf{f})$ is Π^i_n (Σ^i_n), then Γ is said to be a Π^i_n (Σ^i_n) operator. The operator Γ is said to be arithmetical if it is Σ^0_n for some $n \in \mathbb{N}$, and it is said to be Δ^i_n if it is both Π^i_n and Σ^i_n. For $\subseteq$-monotone operators Γ, the inductive definition induced by Γ is said to be arithmetical if Γ is arithmetical. Observe that if Γ is defined by an L-formula in this sense, then the formula

$$\forall \mathsf{f}[(\Gamma(\mathsf{f}) \subseteq \mathsf{f}) \to x \in \mathsf{f}]$$

defines Γ^∞. It follows that if Γ is a Π^1_1 operator, then Γ^∞ is a Π^1_1 set. In fact, something much stronger holds. Say that a set $A \subseteq \mathbb{N}$ is arithmetically inductive if A is m-reducible[1] to the fixed-point of an arithmetical inductive definition. Kleene proved in his [11] that a set $A \subseteq \mathbb{N}$ is Π^1_1 if and only if it is arithmetically inductive.

[1] See below for the definition of m-reducible.

1.2 Classical Computability Theory

We use the term *classical computability theory* for what is usually called computability theory. We suppose the reader has some basic familiarity with this field of mathematical logic. In this thesis we call any function $F\colon \mathbb{N} \rightharpoonup \mathbb{N}$ deemed computable by classical computability theory a Turing computable function. We will follow the old-fashioned notation and write $\{e\}$ for the partial function $F\colon \mathbb{N} \rightharpoonup \mathbb{N}$ computed by the Turing machine with code e. If $F\colon \mathbb{N} \rightharpoonup \mathbb{N}$ is computed by some Turing machine, then we say that F is Turing computable. Classical computability theory knows several notions of reducibility, the most prominent of which is the Turing reducibility, $\leq_{\mathrm{T}}$. In this thesis we will mostly be concerned with the notions of m- and 1-reducibility. A set $A \subseteq \mathbb{N}$ is said to be m-reducible to $B \subseteq \mathbb{N}$, written $A \leq_m B$, if there is a total Turing computable function $F\colon \mathbb{N} \to \mathbb{N}$ such that $A = F^{-1}[B]$. The set A is said to be 1-reducible to B if there is an *injective* total Turing computable function $F\colon \mathbb{N} \to \mathbb{N}$ such that $A = F^{-1}[B]$.

As is standard, we will let A' denote the Turing jump of $A \subseteq \mathbb{N}$. The set $0'$ is then the halting problem for Turing machines.

The s-m-n- and Recursion Theorem. Kleene's s-m-n- and Recursion Theorem are fundamental tools in the study of Kleene's $\mathcal{O}$, and the fact that they extend to the two notions of computability with which we will mostly be concerned is essential to our work. We review these two theorems here.

The s-m-n Theorem 1.2.1 (Kleene). *Let $F\colon \mathbb{N}^{m+n} \rightharpoonup \mathbb{N}$ be Turing computable. Then, there is an injective total Turing computable $s\colon \mathbb{N}^m \to \mathbb{N}$ such that for any $\vec{k} \in \mathbb{N}^m$ and $\vec{\ell} \in \mathbb{N}^n$ we have $\{s(\vec{k})\}(\vec{\ell}) \simeq F(\vec{k}, \vec{\ell})$.*

Proof. Fix a program e computing F. To compute $s(\vec{k})$, add to the program e instructions for writing $\vec{k}$ on the input tape before starting the execution of e; output the index of this new program. By the padding lemma we may suppose that s is injective. q.e.d.

The Recursion Theorem 1.2.2 (Kleene). *Let $F\colon \mathbb{N} \to \mathbb{N}$ be Turing computable. Then, there is an e such that $\{F(e)\} \simeq \{e\}$.*

Proof. By the s-m-n Theorem we may suppose that there is a Turing computable function $s\colon \mathbb{N} \to \mathbb{N}$ such that

$$\{s(d)\}(n) := \begin{cases} \{\{d\}(d)\}(n) & \text{if } \{d\}(d)\downarrow \\ \uparrow & \text{otherwise.} \end{cases}$$

Let d be an index such that $\{d\} \simeq F \circ s$. Let $e := s(d)$ and note that

$$\{e\} \simeq \{s(d)\} \simeq \{\{d\}(d)\} \simeq \{F(s(d))\} \simeq \{F(e)\}.$$

q.e.d.

Computable isomorphism. At several points in this thesis we will have the opportunity to show that two sets are computably isomorphic, a notion we now explain. Two sets $A, B \subseteq \mathbb{N}$ are computably isomorphic, written $A \equiv_1 B$ if there is a bijective Turing computable function $F\colon \mathbb{N} \to \mathbb{N}$ such that $F[A] = B$.

A property $\mathfrak{P}$ of tuples $\vec{A}$ of sets of natural numbers is said to be computably invariant if for all computable isomorphisms we have $\mathfrak{P}(\vec{A}) \Leftrightarrow \mathfrak{P}(F[\vec{A}])$. Following Rogers' influential textbook [17], we may say that classical computability theory is the study of those properties of sets of natural numbers which are computably invariant.[2] Examples of computably invariant properties are "A is c.e.", and "$A \leq_{\mathrm{T}} B$". The important thing for us is the idea that two computably isomorphic sets can be considered identical from the viewpoint of classical computability theory.

A useful theorem in establishing computable isomorphism is Myhill's Isomorphism Theorem (see [16]), the Cantor-Schröder-Bernstein of computability theory.

Myhill's Isomorphism Theorem 1.2.3. *If $A \leq_1 B$ and $B \leq_1 A$, then A and B are computably isomorphic.*

Proof. See, for instance, theorem I.5.4. in Soare's book [21]. q.e.d.

1.3 Kleene's $\mathcal{O}$

We now give the precise definition of Kleene's $\mathcal{O}$.

Definition 1.3.1 (Kleene). *Let $<_{\mathcal{O}}$ be the least binary relation on $\mathbb{N}$ satisfying the following closure conditions:*

(1)	$1 <_{\mathcal{O}} 2$	*(anchor)*
(2)	*If* $m <_{\mathcal{O}} n$, *then* $n <_{\mathcal{O}} 2^n$	*(successor)*
(3)	*If* $k <_{\mathcal{O}} m$ *and* $m <_{\mathcal{O}} n$, *then* $k <_{\mathcal{O}} n$	*(transitivity)*
(4)	*If* $\mathrm{dom}(\{e\}) = \mathbb{N}$, *and we have* $\{e\}(n) <_{\mathcal{O}} \{e\}(n+1)$ *for all* $n \in \mathbb{N}$, *then* $\{e\}(n) <_{\mathcal{O}} 3 \cdot 5^e$ *holds for all* $n \in \mathbb{N}$	*(limit)*

Kleene's $\mathcal{O}$ is the subset of $\mathbb{N}$ coding the relation $<_{\mathcal{O}}$.

We should remark that Kleene in his [9] defines $\mathcal{O}$ to be the field of the relation $<_{\mathcal{O}}$, and not the set coding the relation $<_{\mathcal{O}}$ – it is, however, easy to see that $\mathcal{O}$ as we have defined it, and the field of the relation $<_{\mathcal{O}}$, that is, $\mathcal{O}$ as Kleene defined it, are computably isomorphic.

As the reader will notice, Definition 1.3.1 is impredicative, that is, we define a relation, $<_{\mathcal{O}}$, by quantifying over a set of which the relation we are defining itself is a member. We now show how to obtain $\mathcal{O}$ as the fixed-point of an inductive definition. This is not only a technical exercise, for by so doing, we will be able to compute an upper bound on the complexity of $\mathcal{O}$. Let $U\colon \mathbb{N}^2 \rightharpoonup \mathbb{N}$ be the function computed by the universal Turing machine; thus, $\{\lambda x.U(e,x)\}_{e\in\mathbb{N}}$ is a listing of all Turing computable functions. From Kleene's Normal Form Theorem, it follows that U is Σ^0_1 definable. To define $\mathcal{O}$, let $\varphi(n, \mathsf{f})$ be the formula

$$\begin{aligned}
& n = \langle 1, 2\rangle \\
\vee\ & \exists \langle x, y\rangle \in \mathsf{f}\,(n = \langle y, 2^y\rangle) \\
\vee\ & \exists \langle x, y\rangle \in \mathsf{f}\,\exists \langle y, z\rangle \in \mathsf{f}\,(n = \langle x, z\rangle) \\
\vee\ & \exists e\,[\forall x \exists y\, U(e,x) = y \wedge \\
& \qquad \forall x \langle U(e,x), U(e,x+1)\rangle \in \mathsf{f} \wedge \\
& \qquad \exists x (n = \langle U(e,x), 3 \times 5^e\rangle)]
\end{aligned}$$

[2] This, of course, is just Klein's Erlangen Program executed in the setting of computability theory.

Then $\varphi(n, \mathsf{f})$ defines a monotone function $\Gamma_\varphi \colon 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ such that $\Gamma_\varphi^\infty = \mathcal{O}$. Since $\varphi(n, \mathsf{f})$ is Σ_3^0, Γ_φ is surely definable by a Π_1^1 formula, and thus $\mathcal{O}$ is Π_1^1. In fact, Kleene proved in his [10] that any Π_1^1 set is m-reducible to $\mathcal{O}$; thus $\mathcal{O}$ is complete with respect to the class of Π_1^1 sets of integers. For later reference we note that this implies

Proposition 1.3.2 (Kleene). *Kleene's $\mathcal{O}$ is complete with respect to the class of arithmetically inductive sets.*

Basic properties of Kleene's $\mathcal{O}$. The set $\mathcal{O}$ was introduced by Kleene in his [9] as a system of ordinal notations which is maximal in possessing the following constructive features: the predicates "x is a notation for 0 (notation for a successor ordinal, notation for a limit ordinal)" and "x is a notation for the ordinal which is a successor of the ordinal for which y is a notation" are Turing computable, and there is a Turing computable function $Q \colon \mathbb{N}^2 \rightharpoonup \mathbb{N}$ such that if x is a notation for a limit ordinal α, then there is a sequence of ordinals $\{\alpha_n\}_{n\in\omega}$ whose limit is α and such that $Q(x, n)$ is a notation for α_n.

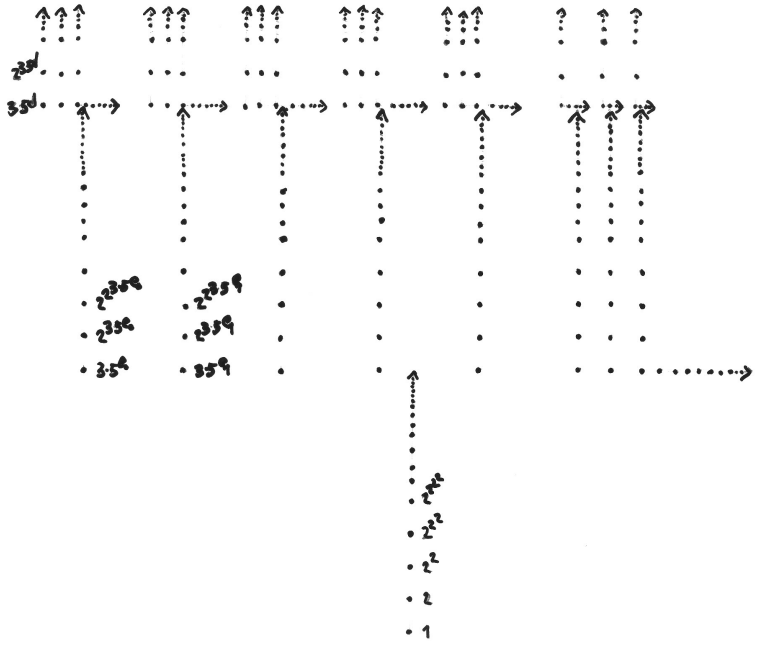

Figure 1.1: The Lower Part of $\mathcal{O}$

One can quite easily show that the relation $<_\mathcal{O}$ is a well-founded partial order. Note that, since any Turing computable function has infinitely many indices, $<_\mathcal{O}$ is not linear; but it is easy to see that for any $n \in \text{field}(\mathcal{O})$, the set $\mathcal{O}\restriction n := \{\langle k, m\rangle \in \mathcal{O} : m <_\mathcal{O} n\}$ codes a linear ordering, and thus a well-ordering; the relation $<_\mathcal{O}$ is thus a well-founded tree in the partial order sense. It is intuitively clear, and relatively easy to prove, that the successor elements of the relation $<_\mathcal{O}$ all have the form 2^k for some $k \in \mathbb{N}$ and that all limit elements of $<_\mathcal{O}$ all have the form $3 \cdot 5^e$ for some $e \in \mathbb{N}$. For $n \in \text{field}(\mathcal{O})$ we denote by $|n|_\mathcal{O}$ the norm of the well-ordering $\mathcal{O}\restriction n$. Because of the contructive

features of $\mathcal{O}$ mentioned above, the set $C := \{|n|_{\mathcal{O}} : n \in \text{field}(\mathcal{O})\}$ is called the set of constructive ordinals. The ordinal $\sup C\,(= |\mathcal{O}|)$ is denoted by ω_1^{CK}, and called Church-Kleene ω_1, or constructive ω_1. Kleene proved that the ordinal ω_1^{CK} coincides with the supremum of the Turing computable ordinals, that is, the supremum of those ordinals which is the norm of some Turing computable well-order.

The hyperarithmetic hierarchy. The well-founded tree $<_{\mathcal{O}}$ is the spine of the hyperarithmetic sets. Define, by recursion on $<_{\mathcal{O}}$, the sequence $\{H_n\}_{n\in\text{field}(\mathcal{O})}$ by

$$\begin{array}{lcll} H_1 & := & \emptyset & \\ H_{2^n} & := & H'_n & \text{for } n \in \text{field}(\mathcal{O}) \\ H_{3\cdot 5^e} & := & \{\langle x, \{e\}(n)\rangle : x \in H_{\{e\}(n)}\} & \text{for } 3\cdot 5^e \in \text{field}(\mathcal{O}) \end{array}$$

A set $A \subseteq \mathbb{N}$ is said to be hyperarithmetic if there is some $n \in \text{field}(\mathcal{O})$ such that $A \leq_{\text{T}} H_n$.

A beautiful theorem of Kleene's is that a set of integers is hyperarithmetic if and only if it is Δ^1_1. For an overview of the theory of hyperarithmetic sets see Part A of Sacks' book [19].

Generalizing Kleene's $\mathcal{O}$ The reader will notice that the only property of the Turing computable functions which is used in the definition of Kleene's $\mathcal{O}$ is the fact that they are indexed by natural numbers. Thus, for any indexed family $\mathcal{F} = \{F_e\}_{e\in\mathbb{N}}$ of partial functions on some set $A \supseteq \mathbb{N}$, we can define a set $\mathcal{O}^{\mathcal{F}}$ by replacing the relation $<_{\mathcal{O}}$ with $<_{\mathcal{O}^{\mathcal{F}}}$ throughout Definition 1.3.1, and by replacing clause *(4)* there with

> *(4)* *If* $\text{dom}(F_e) = \mathbb{N}$, *and we have* $F_e(n) <_{\mathcal{O}^{\mathcal{F}}} F_e(n+1)$ *for all* $n \in \mathbb{N}$, *then* $F_e(n) <_{\mathcal{O}^{\mathcal{F}}} 3\cdot 5^e$ *holds for all* $n \in \mathbb{N}$.

We will call the set $\mathcal{O}^{\mathcal{F}}$ thus obtained *the analogue of Kleene's* $\mathcal{O}$ *for the family* $\mathcal{F}$. The two main objects of study of this thesis are, firstly, the analogue of Kleene's $\mathcal{O}$ for the family of infinite time Turing machine computable functions, and, secondly, the analogue of Kleene's $\mathcal{O}$ for the family of so-called eventually computable functions. We will call the former such analogue $\mathcal{O}^+$ and the latter $\mathcal{O}^{++}$. As already mentioned, we will see that $\mathcal{O}^+$ and $\mathcal{O}^{++}$ extend $\mathcal{O}$ in a very natural way. Not only is it the case that the tree coded by $\mathcal{O}^+$ extends that coded by $\mathcal{O}$, and the tree coded by $\mathcal{O}^{++}$ extends that coded by $\mathcal{O}^+$; there is also the following analogy. For a family $\mathcal{F}$ of functions, say that a set $A \subseteq \mathbb{N}$ is $\mathcal{F}$-decidable if $\chi_A \in \mathcal{F}$, and say that an ordinal α is $\mathcal{F}$-decidable if there is an $\mathcal{F}$-decidable set $A \subseteq \mathbb{N}$ coding α. Now, just as the height of $\mathcal{O}$ equals the supremum of the ordinals decidable by the Turing computable functions (remember that this is the family of functions used to define $\mathcal{O}$), so is the height of $\mathcal{O}^+$ equal to the supremum of the ordinals decidable by the infinite time Turing machine computable functions, and the height of $\mathcal{O}^{++}$ equal to the supremum of the ordinals decidable by the eventually computable functions.

1.4 Infinite Time Turing Machines

We now introduce all the material on infinite time Turing machines that we will need in our study $\mathcal{O}^+$, that is, in our study of the analogue of Kleene's $\mathcal{O}$ for

the family of functions computed by said machines. We do not presuppose on part of the reader any previous acquaintance with the theory of infinite time Turing machines. What follows is a relatively dense compactification of parts of Hamkins & Lewis's paper [5], where the infinite time Turing machine was first defined.

The infinite time Turing machine is one way of making precise the idea of transfinitely long computations. Fundamentally, an infinite time Turing machine is a classical Turing machine which is able to do an infinite number of computation steps before halting. We naturally think of computation steps as being ordered – one step comes after another – with the last step of the computation being a halt. If we want the computation to halt after an infinite number of steps, we are lead to think of its steps as being ordered by ordinals, that being an ordered structure where some elements have infinitely many predecessors, and where every element has a unique successor. Before describing how the infinite time Turing machine works, we describe its hardware.

Hardware. The architecture of an infinite time Turing machine is just like that of a three-tape Turing machine. Thus, an infinite time Turing machine has three tapes, all of which are of infinite length to the right; the upper tape of the machine is the input tape, the middle the scratch tape, and the lower the output tape. In addition to the tapes, there is the so-called head which moves left and right, and reads and writes symbols. An ITTM program is a set of quintuples $\langle s, i, j, t, R/L\rangle$ any of which is to be understood as the following instruction: if in state s reading i, then write j, go to state t and move right/left. Throughout this thesis we suppose that the machines can only read and write 0's and 1's, so in the notation just used we have $i, j \in \{0, 1\}$. Any infinite time Turing machine is completely specified by its program. Programs and states are given natural number codes in a Turing computable manner, so there is an infinite time Turing machine which on input e simulates the program coded by e. In classical computability theory, Turing machines have two special states, namely the initial state and the halting state; in addition to these two states, an infinite time Turing machine has another special state, called the limit state; for reasons of fixity we suppose that the state with the least natural number code is the initial state, the state with the next to least code is the halting state, and the state with the third least code is the limit state.

With this set-up of machine architecture, any Turing machine program is an ITTM program and vice versa. As we will now see, it is only the way a program is executed on an infinite time Turing machine that makes these machines differ from their finite time counterparts.

ITTM computation. As already indicated, the stages in an ITTM computation are indexed by ordinal numbers (this is often phrased as "infinite time Turing machines run on ordinal time"). At the successor stages $\alpha + 1$, an infinite time Turing machine behaves just like its finite time counterpart: the computation up to stage α has put the head over a certain cell; then, at stage $\alpha + 1$ the machine reads what is in this cell and acts according to its program. If an infinite time Turing machine computes transfinitely long it will pass through a limit stage of the computation. At such a limit stage η, the machine goes into the limit state, the head is reset to the beginning of the tape, and the following gets written

on the tape. If there is some $\alpha < \eta$ such that the value in a cell has been i at every stage β for $\alpha < \beta < \eta$, then the value in that cell at stage η is set to i. If the value in a cell has vacillated between 0 and 1 cofinally often before the stage η, then the value in that cell is set to 1. Short-hand for this is to say that an infinite time Turing machine computation acts according to the lim sup rule at limit stages of the computation.[3] The machine goes on computing until it reaches the halting state. If the machine reaches the halting state, then what is written on the output tape at that stage is said to be the output of the computation. We write $P_e(a)$ for the computation of an infinite time Turing machine using the program coded by e on input a. Further, we write $P_e(a)\downarrow$ to express that the computation $P_e(a)$ halts, and similarly $P_e(a)\downarrow = b$ to express that the computation halts with output b. By abuse of notation we call the function computed by program e simply P_e.

One should note that an infinite time Turing machine behaves in a truly infinitary manner at the limit stages of computation. At such a stage the machine has to look at every cell – of which there are infinitely many – and for every cell it has to know what has happened in that cell at all the previous stages of the computation – of which there will be infinitely many.

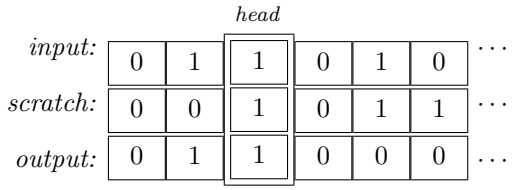

Figure 1.2: An ITTM Caught in Action

ITTM computability. As infinite time Turing machines can compute for transfinitely long, they can also read infinitely long input strings. The right setting for ITTM computability is thus the set of reals, $2^{\mathbb{N}}$. We say that a partial function on the reals, $F\colon 2^{\mathbb{N}} \rightharpoonup 2^{\mathbb{N}}$, is infinite time computable, or ITTM computable, if there is an infinite time Turing machine which on any input $a \in \mathrm{dom}(F)$ halts and outputs $F(a)$ and which does not halt on any input $a \notin \mathrm{dom}(F)$. A set $A \subseteq 2^{\mathbb{N}}$ is infinite time decidable, or ITTM decidable, if its characteristic function χ_A is infinite time computable. A set $A \subseteq 2^{\mathbb{N}}$ is infinite time semi-decidable if there is an infinite time computable function F such that $A = \mathrm{dom}(F)$. Notice that the set A is infinite time semi-decidable if and only if the function $1\restriction A$, that is, the unique function with domain A and range $\{1\}$, is ITTM computable. Notice further that any real $a \in 2^{\mathbb{N}}$ is the characteristic function of a set $A \subseteq \mathbb{N}$ which again is coded as a set $B \subseteq 2^{\mathbb{N}}$. Hence, we may also talk of reals as being ITTM decidable, *etc.*

Snapshots and loops. A snapshot of an ITTM computation is a complete record of some stage in the computation, that is, a record of the contents of

[3] This is contrasted with the lim inf rule where a cofinally vacillating cell is set to 0. We will see later that machines using the lim inf rule have the same computational power as machines using the lim sup rule.

the tapes, which state the machine is in, which cell the head is over, and which program the machine is running. If, in some computation, the same snapshot repeats, then that computation is caught in a loop. Now, if a classical Turing machine computation is caught in a loop, then that computation will never halt. An infinite time Turing machine which is caught in a loop, however, may escape the loop at a limit of the repeating snapshots, and thus halt at some later stage. This is the case, for instance, with the computation which does nothing for ω many steps but then halts at step $\omega + 1$. Less trivial examples can easily be found, where the value in some cell is 0 for a repeating snapshot, but due to the lim sup rule, it is 1 in the limit. For an ITTM computation to be caught in a loop from which it can not escape it is thus necessary (and sufficient) that the limit of any repeating snapshot is again the same repeating snapshot. In this paper, whenever we use the phrase "the computation $P_e(a)$ is caught in a loop" without modification, we mean that the computation $P_e(a)$ is caught in a loop from which it never escapes.

The length of ITTM computations. The first observation done by Hamkins & Lewis in their paper [5] is that a computation which has not halted at some countable ordinal stage $\alpha < \omega_1$ is caught in a loop. The argument is a basic cofinality argument, and we omit the details. The import of this observation is that we do not need quantification over the whole class of ordinals when doing ITTM theory.

The s-m-n- and Recursion Theorem. As already stated, it is important for the work of this thesis that The s-m-n and Recursion Theorem carries over to ITTM computability.

Observation 1.4.1 (The s-m-n Theorem for ITTM computability. Kleene, Hamkins & Lewis). *Let* $F\colon \mathbb{N}^m \times 2^{\mathbb{N}} \rightharpoonup 2^{\mathbb{N}}$ *be* ITTM *computable. Then, there is an injective total Turing computable* $s\colon \mathbb{N}^m \to \mathbb{N}$ *such that* $P_{s(\vec{k})}(a) \simeq F(\vec{k}, a)$ *holds for all* $a \in 2^{\mathbb{N}}$.

Proof. The proof is the same as in the finite time setting. q.e.d.

We emphasize that the function s asserted to exist by this observation can be taken to be *finite time* Turing computable.

Observation 1.4.2 (The Recursion Theorem for ITTM computability. Kleene, Hamkins & Lewis). *For any* ITTM *computable* $F\colon \mathbb{N} \to \mathbb{N}$, *there is an* e *such that* $P_e \simeq P_{F(e)}$.

Proof. Again, the proof is the same as in the finite time setting. q.e.d.

The strength of ITTM computability. Notice that, as the difference between infinite time Turing machines and their finite time counterparts lies solely in the way the former do computations, any Turing machine program is also an ITTM program. Hence, any $F\colon \mathbb{N} \rightharpoonup \mathbb{N}$ which is Turing computable is also ITTM computable. That the computational power of infinite time Turing machines is strictly greater than that of classical Turing machines follows from the following observation.

Observation 1.4.3 (Hamkins & Lewis). *The set $0'$ is is infinite time decidable.*

Proof. Given input (e, n) simulate the computation $P_e(n)$. If this simulation is seen to halt after a finite number of steps, then output 1; if the simulation, however, has not halted by stage ω, then output 0. The point to notice is that an infinite time Turing machine can recognize the ω-th step of its computation, for instance, by flashing the first cell of the scratch tape after every step of the simulation (and being programmed not to be in the limit state when it does this flashing); the machine has then computed for ω many steps the first time it finds itself in the limit state and there is a 1 on the first scratch cell. q.e.d.

Recall that the halting problem for classical Turing machines corresponds to a complete Σ^0_1 set. The Turing undecidability of this set thus implies that no arithmetical set of complexity higher than Σ^0_1 is Turing decidable. It is relatively easy to see, however, that any arithmetical set is ITTM decidable. We omit the proof here and present instead something much stronger.

The Count-Through Theorem 1.4.4 (Hamkins & Lewis). *Every Π^1_1 set is infinite time decidable. Hence, every Σ^1_1 set is decidable as well.*

Proof. It is well-known that WO, the set of reals coding well-orders, is a complete Π^1_1 set.[4] Hence, it is sufficient to show that WO is infinite time decidable. But this follows from the observations that, firstly, given input $a \in 2^{\mathbb{N}}$ an infinite time Turing machine can recognize whether a codes a linear order (this will typically take ω many steps), and then the machine can start searching for and erase smallest elements (searching for a smallest element and then erasing it typically takes $\omega + \omega$ many steps). If, at some point, no smallest element is found, but not all of a is yet erased, then we know that a does not code a well-order, so we can let the machine halt and output 0. If, at last, all of a is erased, then we know that a does code a well-order, so we can let the machine halt and output 1. q.e.d.

One can prove that the relation $P_n(a) = b$ is Δ^1_2. The proof amounts to writing down two formulas – one a Π^1_2 formula and the other a Σ^1_2 formula – defining the relation in question, and we omit that here. The main thrust of the proof is that to say that an infinite time Turing machine on input a halts and outputs b is tantamount to saying that *there is some* countable sequence of computation stages (indexed by ordinals) with certain Δ^1_1 properties which halts with output b; this again is equivalent to saying that *every* sequence of computation stages with certain Δ^1_1 properties halts and outputs b. Together with The Count-Through Theorem 1.4.4 this implies that the limit of the complexity of the infinite time decidable sets lies somewhere between Π^1_1 and Δ^1_2. We also get from this that every semi-decidable set is Δ^1_2: if A is semi-decidable, then $1\!\upharpoonright\! A$ is ITTM computable and thus computed by a program P_e; but then $a \in A$ if and only if $P_e(a) = 1$, and this is Δ^1_2.

Let

$$h := \{e \in \mathbb{N} : P_e(0)\!\downarrow\}$$

This set h is called the lightface halting problem.[5] As expected, we have

[4] See, for instance, [8, p. 136].

[5] Contrasted by the boldface halting problem $H := \{(e, a) \in 2^{\mathbb{N}} : P_e(a)\!\downarrow\}$ which is a much more complex set than what h is. We will not be concerned with H in this thesis.

Proposition 1.4.5 (Hamkins and Lewis). *The set h is not infinite time decidable.*

Proof. The proof is exactly as in the classical setting. Suppose that h is decidable. Use The s-m-n Theorem for ITTM computability to find a Turing computable F such that

$$P_{F(e)}(a) = \begin{cases} \uparrow & \text{if } e \in h \\ 1 & \text{if } e \notin h \end{cases}$$

Let d be a fixed-point of F as given by The Recursion Theorem for ITTM computability. Then note that

$$P_d(0)\downarrow \Leftrightarrow d \in h \Leftrightarrow P_{F(d)}(0)\uparrow \Leftrightarrow P_d(0)\uparrow,$$

a contradiction. q.e.d.

Relativized ITTM computability. The notion of oracle computation – a computation where the machine can ask membership questions about some set $A \subseteq \mathbb{N}$ – is of fundamental importance in classical computability theory. To introduce this notion to the ITTM setting, one should remember that infinite time Turing machines in general will have reals not coding natural numbers on their tapes during a computation. Hence, it is natural to let the oracles for ITTM computations be subsets of $2^{\mathbb{N}}$. One way to think of an infinite time Turing machine with oracle is as follows. To the standard machine architecture is added an extra tape, a query tape, and an "oracle box" full of reals is somehow connected to the machine. The program of an oracle machine can include tuples of the form $\langle q, i, s, t\rangle$ which is to be read as the instruction: if in state q reading symbol i, check whether the real written on the query tape is in the oracle box; if yes, go to state s; if no, go to state t.

We write $P_e^A(a)$ for the ITTM computation of oracle program e on input a, using oracle $A \subseteq 2^{\mathbb{N}}$. We can then introduce a partial pre-order $\leq_\infty$ on $\wp(2^{\mathbb{N}})$ by defining $A \leq_\infty B$ if there is some e such that $\chi_A \simeq P_e^B$. We say that A and B are infinite time equivalent, written $A \equiv_\infty B$, if $A \leq_\infty B$ and $B \leq_\infty A$. We write $[A]_{\equiv_\infty}$ for the $\equiv_\infty$-equivalence class of A, and call these classes ITTM degrees. The partial pre-order $\leq_\infty$ lifts in the natural way to a partial order on the set of $\equiv_\infty$-equivalence classes. In this thesis we will only be interested in real degrees, that is, degrees with reals as members. It is easy to see that $\leq_m \subseteq \leq_{\mathrm{T}} \subseteq (\leq_\infty \cap (2^{\mathbb{N}})^2)$.

All ITTM notions introduced so far have their relativized counterpart; thus, for instance, a function $F\colon 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ is ITTM A-computable if it is computable by an oracle infinite time Turing machine on oracle A, and h^A is the set

$$\{e \in \mathbb{N} : P_e^A(0)\downarrow\}.$$

It is straightforward that The s-m-n Theorem and The Recursion Theorem holds for ITTM A-computability, and thus that h^A is not ITTM A-decidable.

In definability theory, the relativized classes $\Sigma_n^i[A]$ and $\Pi_n^i[A]$ for $A \subseteq \mathbb{N}$ are obtained by adding a predicate symbol interpreted as A to the language. It is then clear that all $\Pi_1^1[A]$ sets of reals are ITTM A-decidable.

Writable, eventually writable, and accidentally writable reals. If $a \in 2^{\mathbb{N}}$ is such that there is some e with $P_e(0)\!\downarrow = a$, then a is said to be a writable real. Obviously, there are only countably many writable reals. Notice that a real is writable if and only if it is decidable: if $a \in 2^{\mathbb{N}}$ is writable, to decide whether $n \in a$ simply write a and look; and if $a \in 2^{\mathbb{N}}$ is decidable, then just systematically go through $\mathbb{N}$ and write n if and only if $\chi_a(n) = 1$.

Even if the computation $P_e(0)$ does not halt, it may be that after some stage what is written on the output tape remains unchanged. This is for instance the case with a machine writing the real h – there is an infinite time Turing machine which simulates all computations $\{P_e(0)\}_{e\in\mathbb{N}}$ and outputs e once it observes that $P_e(0)$ halts – finally this machine will have written h on the output tape. This machine can, however, never halt, for if it did, then h would be writable and thus decidable. The real h is an example of an eventually writable real. To define this notion precisely, let us introduce some notation that will recur later. For $a \in 2^{\mathbb{N}}$ and $\alpha < \omega_1$, let $P_{e,\alpha}(a)$ denote the content of the output tape of the computation $P_e(a)$ at stage α (for the sake of well-definition, we let $P_{e,\alpha}(a) = P_e(a)$ if the computation $P_e(a)$ halts before stage α). If $a \in 2^{\mathbb{N}}$ is such that there is some program $e \in \mathbb{N}$ and some ordinal α such that $P_{e,\beta}(0) = a$ for all $\beta > \alpha$, then a is said to be eventually writable. Eventual writability will play an important role later in this thesis.

A real is said to be accidentally writable if it appears on the output tape at some stage in some computation $P_e(0)$, that is, if there is some $e \in \mathbb{N}$ and some $\alpha < \omega_1$ such that $P_{e,\alpha}(0) = a$.

It is clear that any writable real is eventually writable, and that any eventually writable real is accidentally writable.

Writable, eventually writable, and accidentally writable ordinals. An ordinal $\alpha < \omega_1$ is said to be writable if some real coding it is writable, eventually writable if some real coding it is eventually writable, and accidentally writable if some real coding it is accidentally writable.

Proposition 1.4.6 (Hamkins & Lewis). *There are no gaps in the writable ordinals, that is, if α is writable and $\beta < \alpha$, then β is writable as well. Similarly, there are no gaps in the eventually writable ordinals, and no gaps in the accidentally writable ordinals.*

Proof. If α is writable, then it is coded by a writable real a; and if $\beta < \alpha$, then there there is some $n \in \mathrm{field}(a)$ such that the set $b := \{\langle k, m\rangle \in a \,|\, \langle m, n\rangle \in a\}$ codes β. But b is clearly writable when a is: first write a, then erase all elements not in b.

Now suppose that the a coding α is eventually (or accidentally) writable. We can have a computation eventually (or accidentally) write a on some part of the scratch tape, and for any approximation a^* to a we can write $b^* := \{\langle k, m\rangle \in a^* : \langle m, n\rangle \in a^*\}$ on the output tape, given that b^* is not already written there. Such a procedure will eventually (or accidentally) write b. q.e.d.

In [5], the supremum of the writable ordinals was called λ, and the supremum of the eventually writable ordinals was called ζ. The supremum of the accidentally writable ordinals has later been called Σ. Even though we find this naming unfortunate, it has become standard, so we will follow it here.

Proposition 1.4.7 (Hamkins & Lewis). *We have* $\omega_1^{\mathrm{CK}} < \lambda < \zeta < \Sigma$.

Proof. By a theorem of Feferman and Spector (see [2, Th. 3.7]), there is a Π^1_1 path through Kleene's $\mathcal{O}$; this implies, of course, that there are Π^1_1 reals coding ω_1^{CK}. By The Count-Through Theorem 1.4.4 such reals are decidable and thus writable.

We next show that λ is eventually writable. It follows that $\lambda < \zeta$. Consider a computation which simulates all computations $P_e(0)$. If and when it is found that $P_e(0)\!\downarrow = a$, then check whether a codes a well-order. If so, then put a on top of the well-order already written on the output tape (of course, we are assuming some systematic way of doing this, of which there are plenty). Wait for a new computation $P_e(0)$ to halt.

Eventually this computation will have an ordinal greater than or equal to λ written on its output tape.

The proof that ζ is accidentally writable amounts to the same. The reader is referred to Proposition 3.1.3 below for details. q.e.d.

An ordinal $\alpha < \omega_1$ is said to be clockable if there is some ITTM computation $P_e(0)$ which reaches its halting state at stage $\alpha + 1$. Contrary to the writable ordinals, there are gaps in the clockable ordinals; this is because an ITTM simulating all programs can recognize the first ordinal stage at which no simulated program halts. The supremum of the clockable ordinals was named γ in [5]. Hamkins & Lewis showed that the order type of the clockable ordinals is λ (see [5, Th. 3.8], or the proof of Proposition 2.1.23 below), but left open the question whether $\gamma = \lambda$. This was answered affirmatively by Welch in his [23]. We state Welch's theorem here for later reference

Theorem 1.4.8 (Welch). $\lambda = \gamma$.

We have to wait until after Corollary 3.1.6 below for the proof of this theorem.

All notions of writability and clockability have relativized relatives of A-writability and A-clockability. We write, for instance, λ^A for the supremum of the A-writable ordinals, and γ^A for the supremum of the A-clockable ordinals. The relativized version of Welch's theorem says that $\lambda^A = \gamma^A$ for any $A \subseteq 2^{\mathbb{N}}$.

Chapter 2

The Set $\mathcal{O}^+$

This chapter is devoted to the study of $\mathcal{O}^+$, the analogue of Kleene's $\mathcal{O}$ for the family of ITTM computable functions. After having defined $\mathcal{O}^+$ we provide an algorithm the execution of which eventually writes $\mathcal{O}^+$; next we significantly improve upon this by showing that $\mathcal{O}^+$ is computably isomorphic to h. Finally we determine the height of the tree coded by $\mathcal{O}^+$ to be λ.

Defining $\mathcal{O}^+$. As already indicated, the idea behind the definition of $\mathcal{O}^+$ is simply to replace the Turing computable functions in clause *(4)* of Definition 1.3.1 with ITTM computable functions.

Definition 2.1.9. *Let $<_{\mathcal{O}^+}$ be the least binary relation on $\mathbb{N}$ satisfying the following closure conditions:*

(1)	$1 <_{\mathcal{O}^+} 2$	*(anchor)*
(2)	*If* $m <_{\mathcal{O}^+} n$, *then* $n <_{\mathcal{O}^+} 2^n$	*(successor)*
(3)	*If* $k <_{\mathcal{O}^+} m$ *and* $m <_{\mathcal{O}^+} n$, *then* $k <_{\mathcal{O}^+} n$	*(transitivity)*
(4)	*If* $\mathrm{dom}(P_e) = \mathbb{N}$, *and we have* $P_e(n) <_{\mathcal{O}^+} P_e(n+1)$ *for all* $n \in \mathbb{N}$, *then* $P_e(n) <_{\mathcal{O}^+} 3 \cdot 5^e$ *holds for all* $n \in \mathbb{N}$	*(limit)*

The set $\mathcal{O}^+$ is the subset of $\mathbb{N}$ coding $<_{\mathcal{O}^+}$.

As was the case with Definition 1.3.1, Definition 2.1.9 is also impredicative. Just like we did for $<_{\mathcal{O}}$, however, we can produce an inductive definition of $<_{\mathcal{O}^+}$. In so doing, one should notice that, as ITTM computations are not finite objects, the relation $\mathbb{N} \subseteq \mathrm{dom}(P_e)$ is not arithmetical. In fact, as the predicate $P_e(a)\!\downarrow$ is Δ^1_2, it follows easily that the relation $\mathbb{N} \subseteq \mathrm{dom}(P_e)$ is Δ^1_2 as well. As the other clauses in Definition 2.1.9 are arithmetical, it follows that $\mathcal{O}^+$ is a set inductively defined by a Δ^1_2 function $\Gamma\colon 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ and is thus itself apparently Π^1_2. We will shortly see, in Corollary 2.1.11, that this classification is not optimal.

As with Kleene's $\mathcal{O}$, the set $\mathcal{O}^+$ codes a tree $<_{\mathcal{O}^+}$ on the natural numbers. For any $n \in \mathrm{field}(\mathcal{O}^+)$, we write $\mathcal{O}^+ \restriction n$ for the set $\{\langle k, m\rangle \in \mathcal{O}^+ \mid \langle m, n\rangle \in \mathcal{O}^+\}$. Furthermore, for $n \in \mathrm{field}(\mathcal{O}^+)$, we define $|n|_{\mathcal{O}^+}$ to be the height of the well-ordering $\mathcal{O}^+ \restriction n$, and $|\mathcal{O}^+| := \sup\{|n|_{\mathcal{O}^+} + 1 : n \in \mathrm{field}(\mathcal{O}^+)\}$. In Theorem 2.1.21 below, we will give a characterization of the ordinal $|\mathcal{O}^+|$.

Classifying $\mathcal{O}^+$. We now work slowly towards the classification of $\mathcal{O}^+$ as being computably isomorphic to h. Recall that h is an eventually writable real. The next proposition states $\mathcal{O}^+$ is also eventually writable.

Proposition 2.1.10. *The set $\mathcal{O}^+$ is eventually writable.*

Proof. The idea behind a computation eventually writing $\mathcal{O}^+$ is to do computations $\{P_e(k)\}_{e,k\in\mathbb{N}}$ on the side, while building up $\mathcal{O}^+$ from the bottom and up on the output tape, writing up limit elements $3 \cdot 5^e$ as soon as it is found that $P_e \restriction \mathbb{N}$ is total and cofinal in the current approximation to $\mathcal{O}^+$. Of course, since we do not know whether $P_e(k)$ will halt for arbitrary $e, k \in \mathbb{N}$, we will not know when to stop this process.

A more precise description of an ITTM program which eventually writes $\mathcal{O}^+$ on the output tape is the following.

We make use of a marker, for instance at the first cell of the scratch tape. We think of the scratch tape as being (computably) divided into infinitely many parts. The program starts with the marker being turned off.

Start by writing $\langle 1, 2\rangle$ on the output tape and continue to the limit without any actions. At any limit stage of the construction there are two possibilities according as to whether the marker is on or off.

Marker off. Put marker on, and in ω many steps do the following. For each $e \in \mathbb{N}$ and $n \in \mathbb{N}$ simulate ω many new computation steps of $P_e(n)$, using the $2 \cdot \langle e, n\rangle$-th part of the scratch tape; at the same time close what is written on the output tape off under $\mathcal{O}^+$-successor and transitivity.

Marker on. Put marker off. Let Scratch 0 be some part of the scratch tape not used for the simulation of the computations $\{P_e(n)\}_{e,n\in\mathbb{N}}$. Clear Scratch 0, then use this part of the scratch tape to check whether some $e \in \mathbb{N}$ meets the requirement:

> $(\star)$ All simulations $P_e(n)$ for $n \in \mathbb{N}$ have halted, $\{\langle P_e(n), P_e(n+1)\rangle\}_{n\in\mathbb{N}}$ is written on the output tape, but 3×5^e is not in the field of what is on the output tape.

If some e is found to meet $(\star)$, then write $\{\langle P_e(n), 3 \cdot 5^e\rangle\}_{n\in\mathbb{N}}$ out on the output tape for the least such e.

Notice that even if some stage is reached at which no e is found which meets requirement $(\star)$, this does not mean that no e meeting $(\star)$ will be found at any later stage – after all, it may be that e will meet $(\star)$ once all the computations $\{P_e(n)\}_{n\in\mathbb{N}}$ have halted, but at the current stage some of those computations have yet to halt.

It is clear that an ITTM executing the program just described eventually writes some real a on the output tape and that we have $a \subseteq \mathcal{O}^+$. We show by induction on $<_{\mathcal{O}^+}$ that $\mathcal{O}^+ \subseteq a$. The successor stages are trivial, so let $3 \cdot 5^e \in \text{field}(\mathcal{O}^+)$ and suppose $\mathcal{O}^+ \restriction 3 \cdot 5^e \subseteq a$. There is some stage α in the execution of the program when $\{P_e(n)\}_{n\in\mathbb{N}}$ is written on the scratch tape and $\mathcal{O}^+ \restriction 3 \cdot 5^e$ is written on the output tape. It is then clear that at stage $\alpha + (\omega \cdot 2e)$ one will have found that e is the least number which meets requirement $(\star)$. Hence, at that stage we will have $\{\langle P_e(n), 3 \cdot 5^e\rangle\}_{n\in\mathbb{N}}$ written out on the output tape. q.e.d.

We will see below, in Corollary 2.1.14, that $\mathcal{O}^+$ is not writable.

Corollary 2.1.11. *The set $\mathcal{O}^+$ is semi-decidable, and thus has complexity Δ^1_2.*

Proof. We show that the function $1\!\!\uparrow \mathcal{O}^+$ is ITTM computable. Consider the following procedure. Given input $a \in 2^{\mathbb{N}}$, start the procedure from Theorem 2.1.10 eventually writing $\mathcal{O}^+$. If and when it is found that a is in the field of what is written on the output tape, then halt and output 1. Surely, there is an ITTM program which operates according to this procedure and such a program computes $1\!\!\uparrow \mathcal{O}^+$.

The second claim follows from the fact that every infinite time semi-decidable set is Δ^1_2. q.e.d.

It should be noticed that this corollary depends on the monotonicity of the writing of $\mathcal{O}^+$ in the procedure of Proposition 2.1.10 – in that procedure, once a number is written on the output tape it is never again erased.

We indicated in the proof of Proposition 2.1.10 that we do not know when to stop the writing of $\mathcal{O}^+$ as we do not know h. Let us make this precise by observing that

Observation 2.1.12. *We have $\mathcal{O}^+ \leq_\infty h$.*

Proof. Let

$$h^* := \{(e,n) \in \mathbb{N}^2 \,:\, P_e(n)\!\downarrow\}.$$

Then define $g : \mathbb{N}^2 \times 2^{\mathbb{N}} \rightharpoonup 2^{\mathbb{N}}$ by

$$g(e,n,a) :\simeq P_e(n).$$

Then g is ITTM computable, so The s-m-n Theorem for ITTM computability gives a Turing computable $F\colon \mathbb{N}^2 \to \mathbb{N}$ such that $P_{F(e,n)}(a) \simeq g(e,n,a)$. We have

$$P_{F(e,n)}(0)\!\downarrow \Leftrightarrow g(e,n,0)\!\downarrow \Leftrightarrow P_e(n)\!\downarrow.$$

It follows that $h^* = F^{-1}[h]$; thus, h^* is m-reducible to h via F.

Now consider the procedure in Proposition 2.1.10, and suppose we are using h as an oracle. Using the function F, we may suppose that h^* is written out on the scratch tape and is functioning as an oracle. In ω many steps, using h^* we may write the set $b := \{e \in \mathbb{N} \,:\, \mathrm{dom}(\mathbb{N}) \subseteq \mathrm{dom}(P_e)\}$ out on the scratch tape. Then, using the set b as an oracle, we may simulate during the Marker off subprocedure only the computations $P_e(n)$ for which $e \in b$. When we find that all these simulations have halted, we continue the writing of $\mathcal{O}^+$ until the stage where no e is found meeting requirement $(\star)$. Then we halt with $\mathcal{O}^+$ written on the output tape.

This shows that $\mathcal{O}^+$ is h-writable. Hence, it is also ITTM h-decidable. q.e.d.

In fact, something much stronger obtains:

Theorem 2.1.13. *The sets $\mathcal{O}^+$ and h are computably isomorphic.*

Proof. By Myhill's Isomorphism Theorem 1.2.3 it suffices to prove that the sets $\mathcal{O}^+$ and h 1-reduce to each other.

To show that $h \leq_1 \mathcal{O}^+$, define a function $G\colon \mathbb{N}^2 \rightharpoonup \mathbb{N}$ by

$$G(e,n) := \begin{cases} 2\!\uparrow\!\uparrow n & \text{if } e \in h \\ \uparrow & \text{if } e \notin h \end{cases}$$

It is easy to see that G is ITTM computable, so by The s-m-n Theorem, there is some injective Turing computable $F\colon \mathbb{N} \to \mathbb{N}$ such that $P_{F(e)}(n) \simeq G(e,n)$. We then have

$$e \in h \Rightarrow 3 \cdot 5^{F(e)} \in \text{field}(\mathcal{O}^+) \Rightarrow \langle 1, 3 \cdot 5^{F(e)} \rangle \in \mathcal{O}^+,$$

and

$$\langle 1, 3 \cdot 5^{F(e)} \rangle \in \mathcal{O}^+ \Rightarrow \mathbb{N} \subseteq \text{dom}(P_{F(e)}) \Rightarrow e \in h.$$

Hence, h is 1-reducible to $\mathcal{O}^+$ via the function $n \mapsto \langle 1, 3 \cdot 5^{F(n)} \rangle$.

For the other direction, note that, by Corollary 2.1.11, the function $1\restriction \mathcal{O}^+$ is ITTM computable. Define $G\colon \mathbb{N} \times 2^{\mathbb{N}} \rightharpoonup \mathbb{N}$ by

$$G(n,a) :\simeq 1\restriction \mathcal{O}^+(n).$$

It is clear that G is ITTM computable, so The s-m-n Theorem yields an injective total Turing computable function $F\colon \mathbb{N} \to \mathbb{N}$ such that $G(n,a) \simeq P_{F(n)}(a)$. Note that

$$P_{F(n)}(0)\downarrow \Leftrightarrow G(n,0)\downarrow \Leftrightarrow 1\restriction \mathcal{O}^+(n)\downarrow \Leftrightarrow n \in \mathcal{O}^+.$$

Thus $n \in \mathcal{O}^+ \Leftrightarrow F(n) \in h$, so $\mathcal{O}^+ \leq_1 h$.

We remark that this proof generalizes to show that any ITTM semi-decidable set is 1-reducible to h. q.e.d.

Corollary 2.1.14. *The set $\mathcal{O}^+$ is not writable.*

Proof. If $\mathcal{O}^+$ were writable, then it would also be ITTM decidable, and so would h as well by Theorem 2.1.13. q.e.d.

It follows from Corollary 2.1.14 that the complexity estimate made in Corollary 2.1.11 is optimal, for if $\mathcal{O}^+$ were Π^1_1 or Σ^1_1 it would have been decidable and thus writable.

The accidentally writable norm of $\mathcal{O}^+$. If the computation $P_e(0)$ accidentally writes the real a, that is, if at some point during the computation $P_e(0)$, the real a appears on the output tape, define $|a|_e$ to be the least ordinal α such that $P_{e,\alpha}(0) = a$. Then define $|a|_{\text{acc}}$, the accidentally writable norm of a, to be

$$\min\{|a|_e \,:\, P_e(0) \text{ accidentally writes } a\}.$$

From Welch's Theorem 1.4.8 it follows easily that the accidentally writable norm of h is λ. In view of Theorem 2.1.13 it is equally clear that the accidentally writable norm of $\mathcal{O}^+$ is λ: once a program is seen to halt, write its computably isomorphic $\mathcal{O}^+$-twin on the output tape. Let us show that the eventual writing of $\mathcal{O}^+$ as described in the proof of Proposition 2.1.10 is maximally effective.

Observation 2.1.15. *Let $P_{e_0}(0)$ be some computation acting according to the algorithm described in the proof of Proposition 2.1.10. Then $|\mathcal{O}^+|_{e_0} = \lambda$.*

Proof. For an e such that $\mathbb{N} \subseteq \text{dom}(P_e)$, the supremum of the halting times of the computations $\{P_e(n)\}_{n\in\mathbb{N}}$ is dominated by a clockable ordinal: simulate all computations $\{P_e(n)\}_{n\in\mathbb{N}}$, then halt. It follows that the stage of the writing of $\mathcal{O}^+$ at which $\mathcal{O}^+\restriction n$ appears as part of the output tape is also dominated by a clockable ordinal. Hence, the writing of $\mathcal{O}^+$ as described in the proof of Proposition 2.1.10 takes at most $\gamma = \lambda$ many steps. q.e.d.

The height of $\mathcal{O}^+$. We now aim at proving that $|\mathcal{O}^+|$ is equal to λ, the supremum of the writable ordinals. That $|\mathcal{O}^+| \leq \lambda$ is an easy corollary of the proof of Proposition 2.1.10 where we eventually write $\mathcal{O}^+$.

Lemma 2.1.16. *For any $n \in \text{field}(\mathcal{O}^+)$, the set $\mathcal{O}^+ \restriction n$ is writable. In consequence, we have $|\mathcal{O}^+| \leq \lambda$.*

Proof. Start the procedure in the proof of Proposition 2.1.10 which eventually writes $\mathcal{O}^+$, and continue until n appears in the field of what is written on the output tape while the marker is on. Then continue until the next limit stage. At this stage we will have $\mathcal{O}^+ \restriction n$ included in what is written on the output tape. Now erase, in some systematic fashion, from the output tape all pairs $\langle k, m\rangle$ which are not in $\mathcal{O}^+ \restriction n$, then halt. This writes $\mathcal{O}^+ \restriction n$ on the output tape.

The second statement follows from the fact that $|\mathcal{O}^+|$ is defined as

$$\sup\{|n|_{\mathcal{O}^+} + 1 \,:\, n \in \text{field}(n)\}.$$

q.e.d.

Notice again that the proof of this lemma depends on the monotonicity of the eventual writing of $\mathcal{O}^+$ described in the proof of Proposition 2.1.10.

Now to the harder part, namely that of proving $\lambda \leq |\mathcal{O}^+|$. Our proof of this fact is inspired by the proof of the fact that the Turing computable ordinals embed into the constructive ordinals, as this proof is presented by Sacks in his [19, pp. 15-18]. We do, however, find our constructions to have more intuitive content than the corresponding constructions for classical $\mathcal{O}$; this is due to the enormous power that the infinite time Turing machine affords us.

The idea of the proof is, naturally, to embed writable reals coding well-orders into $\mathcal{O}^+$. This is done by first defining an addition operation on $\mathcal{O}^+$.

Proposition 2.1.17. *There is an* ITTM *computable function $\dot{+}$ such that*

(1) we have $m, n \in \text{field}(\mathcal{O}^+)$ if and only if $m \dot{+} n \in \text{field}(\mathcal{O}^+)$
(2) for all $m, n \in \text{field}(\mathcal{O}^+)$ we have $|m \dot{+} n|_{\mathcal{O}^+} = |m|_{\mathcal{O}^+} + |n|_{\mathcal{O}^+}$
(3) if $m, n \in \text{field}(\mathcal{O}^+)$ and $n \neq 1$, then $m <_{\mathcal{O}^+} m \dot{+} n$

Proof. The idea is to define $m \dot{+} n$ by recursion on n along the well-founded relation $<_{\mathcal{O}^+}$. If n is a successor element of $<_{\mathcal{O}^+}$, then it has the form 2^k for some k, so in that case we want $m \dot{+} n = 2^{m \dot{+} k}$. If n is a limit element of $<_{\mathcal{O}^+}$ then it has the form $3 \cdot 5^e$ for some e, and in that case we want $m \dot{+} n = 3 \cdot 5^d$, where $d \in \mathbb{N}$ codes an ITTM program such that $P_d(k) = m \dot{+} P_e(k)$. By standard use of The Recursion Theorem, one can see that there is an ITTM computable function $\dot{+}$ with these required properties. Some of the gory details are as follows.

Note first that there is an ITTM computable function which given $(e, m, d, n) \in \mathbb{N}^4$ outputs $P_e(m, P_d(n))$. Hence, by The s-m-n Theorem, there is an ITTM computable $h : \mathbb{N}^3 \to \mathbb{N}$ such that $P_{h(e,m,d)}(n) \simeq P_e(m, P_d(n))$. Further, there is an ITTM computable function $F \colon \mathbb{N}^3 \rightharpoonup \mathbb{N}$ such that

$$F(e, m, n) \simeq \begin{cases} m & \text{if } n = 1 \\ 2^{P_e(m,k)} & \text{if } n = 2^k \\ 3 \cdot 5^{h(e,m,d)} & \text{if } n = 3 \cdot 5^d \\ 7 & \text{otherwise.} \end{cases}$$

Another use of The s-m-n Theorem gives an $I\colon \mathbb{N} \to \mathbb{N}$ such that $P_{I(e)}(m,n) \simeq F(e,m,n)$. Let e be a fixed-point of I as given by The Recursion Theorem, then notice that

$$P_e(m,n) \simeq \begin{cases} m & \text{if } n = 1 \\ 2^{P_e(m,k)} & \text{if } n = 2^k \\ 3 \cdot 5^{h(e,m,d)} & \text{if } n = 3 \cdot 5^d \\ 7 & \text{otherwise.} \end{cases}$$

It is then clear that we can let $m \dot{+} n := P_e(m,n)$; note that for $m, 3 \cdot 5^d \in \mathcal{O}^+$ we get $m \dot{+} 3 \cdot 5^d = 3 \cdot 5^{h(e,m,d)}$ where $P_{h(e,m,d)}(k) \simeq m \dot{+} P_d(k)$. To verify that $\dot{+}$ satisfies the properties *(1)*, *(2)*, *(3)*, and even more, is then easy but a bit tedious. We omit the details. The skeptic may want to read the proof of Theorem I.3.4 in [19]. q.e.d.

The addition operation $\dot{+}$ just defined allows us to bound every writable subset of $\mathcal{O}^+$ by taking the infinite $\dot{+}$-sum over a. The bounding can be done in an effective manner via a function taking an index for the writable real in question to an element of $\mathcal{O}^+$ which lies above all elements of a.

Proposition 2.1.18. *There is a Turing computable function $F'\colon \mathbb{N} \to \mathbb{N}$ such that if $P_e(0)\!\downarrow = a$, then,*

if $a \subseteq \mathrm{field}(\mathcal{O}^+)$, then $F'(e) \in \mathrm{field}(\mathcal{O}^+)$, and for all $n \in a$, $|n|_{\mathcal{O}^+} < |F'(e)|_{\mathcal{O}^+}$.

Proof. Consider the following procedure. Given $(e,n) \in \mathbb{N}^2$, start simulating $P_e(0)$. If we find that $P_e(0)\!\downarrow = a$, then clear the input tape and write the set $\{2^m : m \in a\}$ on it in some systematic fashion. Now apply $\dot{+}$ recursively to the elements 2^m for $m \in a$ *in the order these appear on the input tape*, and output the result after having done this n times, that is, compute the function

$$\begin{array}{lcl} g_e(0) & := & \text{the first element appearing on the input tape} \\ g_e(k+1) & := & g_e(k) \dot{+} \text{ the } k+1\text{-th element appearing on the input tape,} \end{array}$$

and output $g_e(n)$. It is clear that some ITTM program P_d operates according to this procedure. By The s-m-n Theorem there is thus some Turing computable $I\colon \mathbb{N} \to \mathbb{N}$ such that $P_{I(e)}(n) = P_d(e,n)$. Now define $F'\colon \mathbb{N} \to \mathbb{N}$ by $F'(e) = 3 \cdot 5^{I(e)}$. Let us prove that F' has the asserted properties. Let $e \in \mathbb{N}$ be arbitrary with $P_e(0)\!\downarrow = a \subseteq \mathrm{field}(\mathcal{O}^+)$. Then, we have $P_{I(e)}(n) \simeq g_e(n)$ for the function g_e described above, and this function is just recursively applying $\dot{+}$ to elements of $\mathrm{field}(\mathcal{O}^+)$, and thus by an obvious induction, and using property *(1)* in Proposition 2.1.17, we will have $P_{I(e)}(n) \in \mathrm{field}(\mathcal{O}^+)$. Moreover, by property *(3)* in Proposition 2.1.17 we get $P_{I(e)}(k) <_{\mathcal{O}^+} P_{I(e)}(k+1)$ for all $k \in \mathbb{N}$. It follows that $3 \cdot 5^{I(e)} \in \mathrm{field}(\mathcal{O}^+)$ and that $P_{I(e)}(k) <_{\mathcal{O}^+} 3 \cdot 5^{I(e)}$ for all $k \in \mathbb{N}$. Hence, $F'(e) \in \mathrm{field}(\mathcal{O}^+)$; and, as for any $n \in a$ there is some $k \in \mathbb{N}$ such that $|n|_{\mathcal{O}^+} < |P_{I(e)}(k)|_{\mathcal{O}^+}$, it follows that for any $n \in \mathbb{N}$ we have $|n|_{\mathcal{O}^+} < |F'(e)|_{\mathcal{O}^+}$. q.e.d.

Corollary 2.1.19. *There is an* ITTM *computable function $F\colon 2^{\mathbb{N}} \rightharpoonup 2^{\mathbb{N}}$ with domain the writables, such that for all $a \in \mathrm{dom}(F)$ we have*

if $a \subseteq \mathrm{field}(\mathcal{O}^+)$, then $F(a) \in \mathrm{field}(\mathcal{O}^+)$, and for any $n \in a$, $|n|_{\mathcal{O}^+} < |F(a)|_{\mathcal{O}^+}$.

Proof. Let F' be the function asserted to exist by the previous proposition. An ITTM algorithm which computes F is then: given a search for an $e \in \mathbb{N}$ such that $P_e(0)\!\downarrow = a$. The first time such an e is found output $F'(e)$ and halt. q.e.d.

Applying the F of this corollary recursively we can embed every writable real coding a well-order into $\mathcal{O}^+$.

Proposition 2.1.20. *There is a Turing computable function $G\colon \mathbb{N} \to \mathbb{N}$ such that if $P_e(0)\!\downarrow = a$ and $a \in \mathrm{WO}$, then $P_{G(e)}(m) \in \mathrm{field}(\mathcal{O}^+)$ for all $m \in \mathrm{field}(a)$, and*

$$m <_a n \text{ implies } |P_{G(e)}(m)|_{\mathcal{O}^+} < |P_{G(e)}(n)|_{\mathcal{O}^+}.$$

Proof. Consider the following procedure. Given (e, n) start simulating $P_e(0)$. If and when it is found that $P_e(0)\!\downarrow = a$ for some $a \in 2^{\mathbb{N}}$, then clear the tapes while transferring a to the input tape. Now check whether a codes a linear ordering, and whether $n \in a$. If one of these do not hold output 7, otherwise go to the following recursive subprocedure. We suppose that the scratch tape is divided into two parts Scratch 1 and Scratch 2, and that all scratch work is done on the Scratch 2 tape. Let F be the ITTM computable function asserted to exist by Corollary 2.1.19

> ($\star$) Suppose a^* is written on the input tape and codes a linear ordering and that c is written on the Scratch 1 tape.
> Compute $F(c)$, and if this computation halts, then write $F(c)$ on the Scratch 1 tape. Then search for the a^*-least element. There are then three possibilities.
> (i) No a^*-least element is found. Then output 7.
> (ii) It is found that n is the a^*-least element. Then transfer the Scratch 1 tape to the output tape and halt.
> (iii) It is found that some element unequal to n is a^*-least. Then go back to ($\star$).

There is certainly an ITTM program P_d which operates according to this procedure. Use the s-m-n Theorem to get a $G\colon \mathbb{N} \to \mathbb{N}$ such that $P_{G(e)}(n) \simeq P_d(e, n)$. We prove that G has the required properties. So suppose $P_e(0)\!\downarrow = a$ and $a \in \mathrm{WO}$. We prove by induction on $<_a$ that $P_{G(e)}(m)\!\downarrow \in \mathrm{field}(\mathcal{O}^+)$ for every $m \in \mathrm{field}(a)$, and that $m <_a n$ implies $|P_{G(e)}(m)|_{\mathcal{O}^+} < |P_{G(e)}(n)|_{\mathcal{O}^+}$. Let $n \in \mathrm{field}(a)$, and suppose that $P_{G(e)}(m)\!\downarrow \in \mathrm{field}(\mathcal{O}^+)$ for all $m <_a n$. As a is writable, it follows that $\{P_{G(e)}(m) : m <_a n\} \subseteq \mathrm{field}(\mathcal{O}^+)$ is writable. But from the description of P_d it is clear that $P_{G(e)}(n) \simeq F(\{P_{G(e)}(m) : m <_a n\})$. Hence, from the properties of F, it follows that $P_{G(e)}(n)\!\downarrow \in \mathrm{field}(\mathcal{O}^+)$ and that $m <_a n$ implies $|P_{G(e)}(m)|_{\mathcal{O}^+} < |P_{G(e)}(n)|_{\mathcal{O}^+}$. q.e.d.

With the functions F and G at hand we can prove that

Theorem 2.1.21. $|\mathcal{O}^+| = \lambda$.

Proof. That $|\mathcal{O}^+| \leq \lambda$ is just Lemma 2.1.16.

To show that $\lambda \leq |\mathcal{O}^+|$, suppose $\alpha < \lambda$. There is then an $a \in \mathrm{WO}$ coding α and an $e \in \mathbb{N}$ such that $P_e(0)\!\downarrow = a$. Using P_e and $P_{G(e)}$, where G is the function of the previous Proposition 2.1.20, we can write a real $P_{G(e)}[a] := c \subseteq \mathrm{field}(\mathcal{O}^+)$ such that $|a| \leq \sup\{|n|_{\mathcal{O}^+} : n \in c\}$. Now apply the F of Corollary 2.1.19 to the real c. We have $F(c) \in \mathrm{field}(\mathcal{O}^+)$ and for all $n \in c$, $|n|_{\mathcal{O}^+} < |F(c)|_{\mathcal{O}^+}$. Hence, $\alpha = |a| \leq |F(c)|_{\mathcal{O}^+} < |\mathcal{O}^+|$. q.e.d.

We conclude with the observation that $\mathcal{O}^+$ as a tree is both fatter and taller than $\mathcal{O}$.

Corollary 2.1.22. *We have $\mathcal{O} \subseteq \mathcal{O}^+$, and the tree coded by $\mathcal{O}^+$ is both taller and fatter than that coded by $\mathcal{O}$; thus, in fact, $\mathcal{O} \subsetneq \mathcal{O}^+$.*

Proof. That $\mathcal{O} \subseteq \mathcal{O}^+$ follows from the fact that the set of Turing machine programs and the set of ITTM programs coincide. That the tree coded by $\mathcal{O}^+$ is taller than the one coded by $\mathcal{O}$ is just the fact that $\omega_1^{\mathrm{CK}} < \lambda$. By $\mathcal{O}^+$ being fatter than $\mathcal{O}$ we mean that for any limit ordinal $\alpha < \omega_1^{\mathrm{CK}}$, we have

$$\{n \in \mathbb{N} : |n|_{\mathcal{O}} = \alpha\} \subsetneq \{n \in \mathbb{N} : |n|_{\mathcal{O}^+} = \alpha\}.$$

To see why this is so, let $3{\cdot}5^e \in \mathrm{field}(\mathcal{O})$, and let g be a function which enumerates $0'$ in strictly increasing order. By induction on $|3 \cdot 5^e|_{\mathcal{O}}$, one shows that the composition $\{e\} \circ g$ is such that $\{e\}(g(n)) <_{\mathcal{O}} \{e\}(g(n+1))$ holds for all $n \in \mathbb{N}$. Furthermore, $\{e\} \circ g$ is ITTM computable, so there is some $d \in \mathbb{N}$ such that $P_d \simeq \{e\} \circ g$; but $\{e\} \circ g$ is not Turing computable (otherwise, g would be computable, contradicting the unsolvability of the halting problem); thus $3 \cdot 5^d \in \{n \in \mathbb{N} : |n|_{\mathcal{O}^+} = \alpha\} - \{n \in \mathbb{N} : |n|_{\mathcal{O}} = \alpha\}$. q.e.d.

Linear orders of height λ. As the successor of any writable ordinal is also writable, we know that λ, the supremum of the writable ordinals, is itself not writable. We have just seen that there is a "reasonably simple" set coding a well-founded relation of height λ – namely, the set $\mathcal{O}^+$. As the reader is aware, the relation coded by $\mathcal{O}^+$ is, however, not linear. It is very natural to ask whether there can be a similarly "reasonably simple" set coding a *well-order* of height λ. The parallel of this question for classical Kleene's $\mathcal{O}$ has a positive solution: call a set $a \subseteq \mathcal{O}$ such that $|a| = |\mathcal{O}|$ a path through $\mathcal{O}$ – Feferman and Spector proved in their [2] that there is a Π^1_1 path through $\mathcal{O}$; consequently, there is a Π^1_1 set $A \subseteq \mathbb{N}$ coding a well-order of height ω_1^{CK}. While the proof for this fact in the classical case is far from trivial, it is relatively easy to see that:

Proposition 2.1.23 (Hamkins & Lewis). *There is a set $\mathscr{C} \subseteq \mathbb{N}$ coding a well-order of height λ, and which is computably isomorphic to $\mathcal{O}^+$.*

Proof. Define $\mathscr{C}$ by letting $\langle d, e\rangle \in \mathscr{C}$ if and only if $P_d(0)$ halts at stage α, $P_e(0)$ halts at stage β, and we have either $\alpha < \beta$ or $\alpha = \beta$ and $d < e$. Then it is clear that $\mathscr{C}$ codes well-order, $<_{\mathscr{C}}$. We first show that $<_{\mathscr{C}}$ has height λ, and thereafter that it is computably isomorphic to h; the proposition then follows from the fact, proved in Theorem 2.1.13, that $\mathcal{O}^+$ and h are computably isomorphic.

By Welch's Theorem 1.4.8, it is clear that $<_{\mathscr{C}}$ has height at most $\omega \cdot \lambda = \lambda$. To prove that $<_{\mathscr{C}}$ has height equal to λ, it thus suffices to prove that each $\alpha < \lambda$ injects into $\mathscr{C}$. To that end, let $P_{\hat{n}}$ be the program which on input 0 writes a code a for α, searches through a as in The Count-Through Theorem 1.4.4, and halts if and when it finds $n \in \mathrm{field}(a)$. It is clear that if $m, n \in \mathrm{field}(a)$ and $m \neq n$, then $P_{\hat{m}}(0)$ and $P_{\hat{n}}(0)$ halt at different times. It follows that α injects into $\mathscr{C}$.

To show that $\mathscr{C} \equiv_1 h$ it suffices by Myhill's Isomorphism Theorem 1.2.3 to prove that $\mathscr{C} \leq_1 h$ and that $h \leq_1 \mathscr{C}$. For the first of these, it suffices to note that $\mathscr{C}$ is ITTM semi-decidable, for then it will be 1-reducible to h by the remark at

the end of the proof of Theorem 2.1.13. But it is clear that $\mathscr{C}$ is semi-decidable: to semi-decide whether $\langle d, e\rangle \in \mathscr{C}$ just run $P_d(0)$ and $P_e(0)$ to check whether there are α and β such that either $\alpha < \beta$ or both $\alpha = \beta$ and $d < e$ holds.

For the other direction, namely $h \leq_1 \mathscr{C}$, let $F\colon \mathbb{N} \to \mathbb{N}$ be a Turing computable function which takes any e and outputs the code of a program obtained from P_e by substituting all mention of the halting state in P_e with instructions for doing one more computation step before halting. That such an F exists is clear by Church's Thesis, and by the padding lemma we may suppose that F is injective. Moreover, we have

$$e \in h \Leftrightarrow \langle e, F(e)\rangle \in \mathscr{C},$$

thus, $h \leq_1 \mathscr{C}$. q.e.d.

Chapter 3

Eventual Computability and The Set $\mathcal{O}^{++}$

Recall that a computation $P_e(a)$, even if it does not halt, may stabilize with some constant value on the output tape. This is for instance the case with the computation $P_e(0)$ which is eventually writing h. In such a case we say that $P_e(a)$ eventually computes a value. In this section we define $\mathcal{O}^{++}$, the analogue of Kleene's $\mathcal{O}$ for the family of eventually computable functions. In section 3.1 we introduce this class of functions, and review some work of Welch's which will be of great use to us. In section 3.2 we define $\mathcal{O}^{++}$ and determine the height of the tree coded by $\mathcal{O}^{++}$ to be ζ, the supremum of the eventually writable ordinals. Then, in section 3.3 we review the theory of accidentally writable degrees and show that $\mathcal{O}^{++}$ is of the maximal such degree. This classification is then improved upon to a theorem stating that $\mathcal{O}^{++}$ is computably isomorphic to s, "the stabilization problem", that is, the set of those programs that eventually writes a real. Next, in section 3.4 we look at the notion of eventual computability in the setting of classical computability and observe that the analogue of Kleene's $\mathcal{O}$ for the resulting family of functions is just classical Kleene's $\mathcal{O}$ relativized to $0'$, both from the computability theoretic and the order theoretic viewpoint. In section 3.5 we note that, by work of Welch and Burgess, the classification of $\mathcal{O}^{++}$ as being computably isomorphic to s implies that $\mathcal{O}^{++}$ is computably isomorphic to many other naturally arising sets, and moreover, that it is complete with respect to the class of arithmetically quasi-inductive sets. The final section 3.6 then draws some speculative consequences from the results of section 3.5 and suggests that there is a parallel of hyperarithmetic theory in which $\mathcal{O}^{++}$ and the eventually writable reals play a major role.

3.1 Eventual Computability

It is convenient when working with eventual computability to only consider infinite time Turing machines that never enter the halting state.

ITTM programs without halting state. Let $\{Q_e\}_{e\in\mathbb{N}}$ be a Turing computable enumeration of all ITTM programs with no transitions to the halting state. We assume that Q_e is obtained from P_e by replacing all mentions of the halting state in P_e with instructions telling the machine to go into a loop without writing anything on the tapes.

Eventually computable functions. We let $Q_e(a)$ denote the computation of program Q_e on input a, that is, $Q_e(a)$ is an ω_1-sequence of ITTM snapshots

according to the program Q_e on input a. For $\alpha < \omega_1$, we let $Q_{e,\alpha}(a)$ denote the content of the output tape of the α-th snapshot in the sequence $Q_e(a)$. If there is some $\alpha < \omega_1$ and some $b \in 2^{\mathbb{N}}$, such that for all $\beta > \alpha$ we have $Q_{e,\beta}(a) = b$, then we write $Q_e(a)\!\uparrow = b$. We write $Q_e(a)\!\uparrow$ if there is a $b \in 2^{\mathbb{N}}$ such that $Q_e(a)\!\uparrow = b$, and $Q_e(a)\!\Uparrow$ if there is no such b. We say that $F\colon 2^{\mathbb{N}} \rightharpoonup 2^{\mathbb{N}}$ is eventually computable if there is an e such that $Q_e(a)\!\uparrow = F(a)$ for any $a \in \mathrm{dom}(F)$ and $Q_e(a)\!\Uparrow$ for any $a \notin \mathrm{dom}(F)$. In this case we say that Q_e eventually computes F. By abuse of notation, we sometimes write $Q_e\colon 2^{\mathbb{N}} \rightharpoonup 2^{\mathbb{N}}$ for the function eventually computed by Q_e. Thus, $Q_e(a)\!\uparrow = b$ and $Q_e(a) = b$ means the same thing, but they have slightly different connotations: in the former case we are thinking of the computation $Q_e(a)$, and thus of b as being the eventual value of this computation, whereas in the latter case we are only thinking of the function, as a set-theoretic object, eventually computed by Q_e.

It is evident from how the programs $\{Q_e\}_{e\in\mathbb{N}}$ are obtained from the programs $\{P_e\}_{e\in\mathbb{N}}$ that any ITTM computable function is also eventually computable. To see that eventual computability is strictly stronger than plain ITTM computability, consider the following algorithm. Given input $e \in \mathbb{N}$ start simulating $P_e(0)$. If this simulation is found to halt, then write 1 on the output and go into a loop. It is clear that an ITTM operating according to this algorithm eventually computes χ_h.

If we say that a set $A \subseteq 2^{\mathbb{N}}$ is eventually computable if its characteristic function is eventually computable, it is easy to see that all eventually computable sets are Δ^1_2. Similarly, we can see that all eventually semi-decidable sets are Δ^1_2.

The s-m-n- and Recursion Theorem. We note that The s-m-n Theorem and The Recursion Theorem holds for eventual computability.

Proposition 3.1.1 (The s-m-n Theorem for eventual computability). *Let $F\colon \mathbb{N}^m \times 2^{\mathbb{N}} \rightharpoonup 2^{\mathbb{N}}$ be eventually computable. Then, there is an injective total Turing computable $s\colon \mathbb{N}^m \to \mathbb{N}$ such that $Q_{s(\vec{k})}(a) \simeq F(\vec{k}, a)$ holds for all $a \in 2^{\mathbb{N}}$.*

Proof. Just like in the classical case. q.e.d.

Proposition 3.1.2 (The Recursion Theorem for eventual computability). *For any eventually computable $F\colon \mathbb{N} \to \mathbb{N}$, there is an e such that $Q_e \simeq Q_{F(e)}$.*

Proof. Just like in the classical case. q.e.d.

The ordinal production machine. One of the main techniques in proving that some function is eventually computable is to do approximations along an ordinal. For doing that, the following lemma will turn out to be very useful.

Lemma 3.1.3 (Welch). *There is an infinite time Turing machine with the property that when its first scratch cell shows a 0, then it has an ordinal written on its output tape. Moreover, arbitrarily large accidentally writable ordinals get written on its output tape, and from some point on only ordinals $\alpha \geq \zeta$ are written.*

Proof. The machine in question acts according to the following recursive procedure.

> Simulate, for each $e \in \mathbb{N}$, ω many steps of $Q_e(0)$. Store all current values of $Q_e(0)$ which code well-orders. Then, in some systematic

fashion compute the sum of all these well-orders and write the result on the output tape.

The first scratch cell is initially set to 1, and at each time a whole sum is written out, it is put to 0 and then to 1 again. Thus, when the first scratch cell has value 0, there is an ordinal written on the output tape. At compound limits there may be gibberish written on the output tape. By use of an indicator on the second scratch cell this is erased.

Now, from some point on, all computations $Q_e(0)$ that eventually write an ordinal have settled on this ordinal, so the sum of all current values of $Q_e(0)$ must be greater than or equal to ζ. q.e.d.

For ease of reference, we will call the machine asserted to exist by Lemma 3.1.3 the Ordinal Production Machine.

Welch's Lemma. The proof of Welch's Theorem 1.4.8 rests on a lemma (Corollary 3.1.6 below) which is also very interesting in its own right. It implies that any ITTM computation starts looping no later than at stage ζ. Below, in Observation 3.3.9, we will improve this result to the surprising fact that any non-halting ITTM computation starts looping either before stage λ or precisely at stage ζ.

Suppose we have an enumeration of all the cells on an infinite time Turing machine. We write $Q_{e,\alpha}(a)\lceil n\rceil$ for the value (which is either 0 or 1) of cell n at stage α in the computation $Q_e(a)$. For $(e, a, n, \alpha) \in \mathbb{N} \times 2^{\mathbb{N}} \times \mathbb{N} \times \omega_1$, we define the stabilization function δ by

$$\delta(e, a, n, \alpha) := \sup\{\beta < \alpha \,:\, \beta = 0 \text{ or } Q_{e,\beta}(a)\lceil n\rceil \neq Q_{e,\beta+1}(a)\lceil n\rceil\}.$$

We write $\delta^e_n(\alpha)$ for $\delta(e, 0, n, \alpha)$. For limit ordinals α, we say that the cell n has locally stabilized in the computation $Q_e(a)$ at stage α if $\delta(e, a, n, \alpha) < \alpha$. We say that the cell n has stabilized at α if $\delta(e, a, n, \beta) < \alpha$ for all $\beta \geq \alpha$. The next proposition, due to Welch ([23]), says that if the computation $Q_e(0)$ is locally stable at Σ, then it stabilized, in fact, before ζ.

Proposition 3.1.4 (Welch). *If $\delta^e_n(\Sigma) < \Sigma$, then $\delta^e_n(\Sigma) < \zeta$.*

Proof. Consider the following procedure. Given an ordinal α from the Ordinal Production Machine, compute $\delta^e_n(\alpha)$. If what is written on the output tape is either not in WO or codes an ordinal of length less than $\delta^e_n(\alpha)$, then write a code for $\delta^e_n(\alpha)$ on the output tape and wait for a new ordinal from the Ordinal Production Machine. In any other case, just wait for a new ordinal from the Ordinal Production Machine.

Now, at some point the Ordinal Production Machine will produce an ordinal α such that $\delta^e_n(\alpha) = \delta^e_n(\Sigma)$. The first time this happens $\delta^e_n(\Sigma)$ will get written on the output tape and will never be erased. Thus, $\delta^e_n(\Sigma)$ is eventually writable. q.e.d.

We now prove a converse of the previous Proposition 3.1.4, namely that if a cell is locally stable at ζ, then it will remain stable all the way up to Σ. In Corollary 3.1.6 we note how this implies that the cell in question remains stable to eternity.

Lemma 3.1.5. *If $\delta^e_n(\zeta) < \zeta$, then $\delta^e_n(\Sigma) = \delta^e_n(\zeta)$.*[1]

Proof. Suppose $\delta^e_n(\zeta) = \alpha < \zeta$, and let $Q_d(0)\!\uparrow = a$, a code for α. Consider the following recursive procedure, where η is the number of times the procedure has been repeated. ($\star$) Simulate ω many steps of the computation $Q_d(0)$, and let β be given by the OPM. Let $a^* = Q_{e,\omega\cdot\eta}(0)$ be the current approximation to a. Check whether $a^* \in$ WO and if so whether the ordinal α^* coded by a^* is less than β. If either of these do not hold, then return to ($\star$). If both hold, then simulate $Q_e(0)$ up to stage α^*, store the value $Q_{e,\alpha^*}(0)\lceil n\rceil$, and then continue simulating $Q_e(0)$ along β until it is found that the value in cell n changes. If such a stage is found, then write the ordinal at which this first happens on the output tape, if this ordinal is not already written on the output tape. In any case, wait for a new ordinal from the Ordinal Production Machine and go to ($\star$).

If the value of cell n changes between ζ and Σ, a machine acting according to the algorithm just described will eventually write an ordinal $\beta > \zeta$, a contradiction. Hence the value in cell n stays constant from stage $\delta^e_n(\zeta)$ all through to Σ, so $\delta^e_n(\Sigma) = \delta^e_n(\zeta)$. q.e.d.

Corollary 3.1.6. *The ζ-snapshot of the computation $Q_e(0)$ is the same as its Σ-snapshot, and the computation $Q_e(0)$ never escapes the loop in which it finds itself at stage ζ.*

Proof. At both stage ζ and stage Σ, the machine is in the limit state with the head set over the left-most cells. By Proposition 3.1.4, if $Q_{e,\Sigma}(0)\lceil n\rceil = 0$, then $Q_{e,\zeta}(0)\lceil n\rceil = 0$ as well. And by Lemma 3.1.5, if $Q_{e,\zeta}(0)\lceil n\rceil = 0$, then also $Q_{e,\Sigma}(0)\lceil n\rceil = 0$. It follows that the ζ-snapshot and the Σ-snapshot of the computation $Q_e(0)$ equal each other.

For the second assertion, note that if $Q_{e,\zeta}(0)\lceil n\rceil = 0$, then $Q_{e,\alpha}(0)\lceil n\rceil = 0$ for all $\alpha \geq \zeta$. It follows that the limit of the repeating ζ-snapshot of $Q_e(0)$ is again the same ζ-snapshot. q.e.d.

The proof of Welch's Theorem 1.4.8. Let us observe how Welch's Theorem 1.4.8 follows from Corollary 3.1.6.

Proof of Theorem 1.4.8 (Welch). We want to show that $\gamma = \lambda$. It is easy to see that any writable ordinal α is dominated by some clockable ordinal: write α, then erase α systematically as in the proof of The Count-Through Theorem 1.4.4, and halt. Hence $\lambda \leq \gamma$. For the other direction, note first that if $P_e(0)$ halts at α, and $\alpha < \Sigma$, then $\alpha < \lambda$: given an ordinal β from the OPM, run $P_e(0)$ along β, and output β if the simulation of $P_e(0)$ has halted before β. Thus, in view of Corollary 3.1.6, the computation $P_e(0)$ either halts before λ, or it goes into an infinite loop. Thus, $\gamma \leq \lambda$. q.e.d.

[1] A note on the origin of this Lemma is in order. The Lemma is necessary in establishing Corollary 3.1.6 below. Welch did not, however, prove it in his paper [23]. I have learned that Welch was recently made aware of this (admittedly, easily fixable) gap in his proof and that he has provided a proof of Lemma 3.1.5. The proof of Lemma 3.1.5 given here, however, was found independently by myself.

The Timing Theorem. If $Q_e(a)\!\uparrow = b$, then call the least ordinal α such that $Q_{e,\beta}(a) = b$ for all $\beta \geq \alpha$ the stabilization point of $Q_e(b)$. The following proposition – called the Timing Theorem by Hamkins & Lewis – extends Welch's result that $\lambda = \gamma$ to the case of eventual and accidental writability.

The Timing Theorem 3.1.7 (Welch, Hamkins & Lewis).

(1) *The computation $Q_e(0)$ accidentally writes a if and only if $Q_{e,\alpha}(0) = a$ for some $\alpha < \Sigma$.*
(2) *We have $Q_e(0)\!\uparrow$ if and only if the stabilization point of $Q_e(0)$ is less than ζ.*
(3) *If $Q_{e,\alpha}(0) = a$ for some $\alpha < \zeta$, then a is eventually writable.*

Proof. The *if* direction of *(1)* is trivial, and the *only if* direction is clear in view of Corollary 3.1.6.

For the *only if* direction of *(2)*, suppose that $Q_e(0)\!\uparrow = a$. We must show that there is some writable ordinal α such that $Q_{e,\alpha}(0) = a$. Consider the following recursive procedure where η is the number of times the procedure has been repeated. $(\star)$ Given an approximation $a^* = Q_{e,\omega\cdot\eta}(0)$ to a and an ordinal β from the Ordinal Production Machine, run the computation $Q_e(0)$ along β, and check whether a^* is final on the output tape in this computation. If so, then write on the output tape the ordinal from which a^* is final, given, as usual, that this ordinal is not already written there. Then, wait for a new (and better) approximation to a and a new ordinal and go to $(\star)$. Note that, by Corollary 3.1.6, we have $Q_{e,\alpha}(0) = a$ for all $\alpha \geq \zeta$. Hence, at the point where we are executing the procedure just described with the true a and with some ordinal greater than ζ, we will get the stabilization point of $Q_e(0)$ written on the output tape. Since from some point on the Ordinal Production Machine only produces ordinals greater than or equal to ζ it follows that this stabilization point is in fact eventually writable.

The *if* direction of *(2)* is clear in view of Lemma 3.1.5 and Corollary 3.1.6

For *(3)*, suppose that $Q_{e,\alpha}(0) = a$ for eventually writable α. Then, by computing approximations α^* to α and running $Q_e(0)$ along α^* writing $Q_{e,\alpha^*}(0)$ on the output tape if it is not already written there, we eventually write a. q.e.d.

Relativizing to eventually writable inputs. In the next propositions we generalize the foregoing to computations on eventually writable inputs. It should be noticed that these results depend on the special case of 0 inputs. If $P_e(0) = a$ and $P_{e,\beta}(0) = a^*$, then we call a^* a β-approximation to a.

Proposition 3.1.8. *Suppose b is eventually writable. If $\delta(e, b, n, \Sigma) < \Sigma$, then $\delta(e, b, n, \Sigma) < \zeta$.*

Proof. Do the proof of Welch's Proposition 3.1.4 but with $\delta(e, b^*, n, \Sigma)$, where b^* is a β-approximation to b for an accidentally writable ordinal β produced by the Ordinal Production Machine. By The Timing Theorem 3.1.7 *(2)* we will be computing with the true b once the Ordinal Production Machine is only producing ordinals $\beta \geq \zeta$. q.e.d.

Lemma 3.1.9. *Suppose b is eventually writable. If $\delta(e, b, n, \zeta) < \zeta$, then $\delta(e, b, n, \Sigma) = \delta(e, b, n, \zeta)$.*

Proof. Do the proof of Lemma 3.1.5, but with $Q_e(b^*)$, where b^* is a β-approximation to b, with β the ordinal gotten from the Ordinal Production Machine. q.e.d.

Corollary 3.1.10. *Suppose b be eventually writable. The ζ-snapshot of the computation $Q_e(b)$ is the same as its Σ-snapshot, and the computation $Q_e(b)$ never escapes the loop in which it finds itself at stage ζ.*

Proof. This follows from Proposition 3.1.8 and Lemma 3.1.9 in exactly the same way that Corollary 3.1.6 follows from Proposition 3.1.4 and Lemma 3.1.5. q.e.d.

The Relativized Timing Theorem 3.1.11. *Suppose b is eventually writable. Then*

(1) *The computation $Q_e(b)$ accidentally writes a if and only if $Q_{e,\alpha}(b) = a$ for some $\alpha < \Sigma$.*
(2) *We have $Q_e(b)\!\uparrow = a$ if and only if the stabilization point of $Q_e(b)$ is less than ζ.*
(3) *If $Q_{e,\alpha}(b) = a$ for some $\alpha < \zeta$, then a is eventually writable.*

Proof. Do the proof of The Timing Theorem 3.1.7 but using approximations to b as well. q.e.d.

It follows from this that the class of eventually computable functions computing on the eventually writable reals is closed under composition. A special case of this is proved in the Technical Lemma 3.2.4 below.

3.2 The set $\mathcal{O}^{++}$

We now define $\mathcal{O}^{++}$, the analogue of Kleene's $\mathcal{O}$ for the class of eventually computable functions. We show that $\mathcal{O}^{++}$ codes a tree of height ζ.

Definition 3.2.1. *Let $<_{\mathcal{O}^{++}}$ be the least binary relation on $\mathbb{N}$ satisfying the following closure conditions:*

(1)	$1 <_{\mathcal{O}^{++}} 2$	*(anchor)*
(2)	*If* $m <_{\mathcal{O}^{++}} n$, *then* $n <_{\mathcal{O}^{++}} 2^n$	*(successor)*
(3)	*If* $k <_{\mathcal{O}^{++}} m$ *and* $m <_{\mathcal{O}^{++}} n$, *then* $k <_{\mathcal{O}^{++}} n$	*(transitivity)*
(4)	*If* $\mathrm{dom}(Q_e) = \mathbb{N}$, *and we have* $Q_e(n) <_{\mathcal{O}^{++}} Q_e(n+1)$ *for all* $n \in \mathbb{N}$, *then* $Q_e(n) <_{\mathcal{O}^{++}} 3 \cdot 5^e$ *holds for all* $n \in \mathbb{N}$.	*(limit)*

The set $\mathcal{O}^{++}$ is the subset of $\mathbb{N}$ coding $<_{\mathcal{O}^{++}}$.

All the remarks made concerning $\mathcal{O}^{+}$ after Definition 2.1.9 carry over to $\mathcal{O}^{++}$. Thus, for instance, $<_{\mathcal{O}^{++}}$ is a well-founded tree with initial segments $\mathcal{O}^{++}\!\restriction n$ coding well-orders. We let $|n|_{\mathcal{O}^{++}}$ denote the height of the well-order $\mathcal{O}^{++}\!\restriction n$, and $|\mathcal{O}^{++}| := \sup\{|n|_{\mathcal{O}^{++}} + 1 : n \in \mathrm{field}(\mathcal{O}^{++})\}$.

The height of $\mathcal{O}^{++}$. We show that $|\mathcal{O}^{++}| = \zeta$. To this end, we note

Proposition 3.2.2. *If $n \in \mathrm{field}(\mathcal{O}^{++})$, then $\mathcal{O}^{++}\!\restriction n$ is eventually writable.*

Proof. The main idea behind a computation eventually writing $\mathcal{O}^{++}\!\restriction n$ is the following. On some part of the scratch tape we can make approximations a^* to $\mathcal{O}^{++}$ constructed with the help of all current values $\{Q_{e,\omega\cdot\eta}(k)\}_{e,k\in\mathbb{N}}$. If

$n \in \text{field}(a^*)$, then we write $a^* \restriction n$ on the output tape, given that this real is not already written there. The point is that if $n \in \text{field}(\mathcal{O}^{++})$, then all the computations $\{Q_e(k)\}_{k\in\mathbb{N}}$ for $3 \cdot 5^e <_{\mathcal{O}^{++}} n$ will finally stabilize, so that, finally, for any approximation a^* we will have $a^* \restriction n = \mathcal{O}^{++} \restriction n$. Thus, we will be eventually writing $\mathcal{O}^{++}$.

A brief description of how this works goes as follows. We think of the scratch tape as divided into infinitely many scratch tapes. On the Scratch 0 tape, we will be writing approximations to $\mathcal{O}^{++}$. Start by writing $\langle 1, 2\rangle$ on the Scratch 0 tape. As in the eventual writing of $\mathcal{O}^{+}$, we use a marker. At the η-th limit stage there are two possibilities.

Marker off. Turn marker on, then close what is written on the Scratch 0 tape off under $\mathcal{O}^{++}$-successor and transitivity. At the same time, simulate ω many new computation steps of $Q_e(k)$ for all $e, k \in \mathbb{N}$. Then, in ω many steps, for every e such that $Q_{e,\omega\cdot\eta+k}(m) \neq Q_{e,\omega\cdot\eta+k+1}(m)$ for some $k \in \mathbb{N}$, erase from the Scratch 0 tape all pairs $\langle k, m\rangle$ such that $\langle 3 \cdot 5^e, m\rangle$ is written on the Scratch 0 tape, and all pairs $\langle k, 3 \cdot 5^e\rangle$.

Marker on. Turn the marker off, then search for the least e such that $\{\langle Q_{e,\omega\cdot\eta}(k), Q_{e,\omega\cdot\eta}(k+1)\rangle\}_{k\in\mathbb{N}}$ is written on the Scratch 0 tape but such that no $\langle m, 3 \cdot 5^e\rangle$ is. If such an e is found then write $\{\langle Q_{e,\omega\cdot\eta}(k), 3 \cdot 5^e\rangle\}_{k\in\mathbb{N}}$ on the Scratch 0 tape. Let a^* be the current content of the Scratch 0 tape. If $n \in \text{field}(a^*)$, then write $a^* \restriction n$ on the output tape, given that it is not already written there.

q.e.d.

Corollary 3.2.3. $|\mathcal{O}^{++}| \leq \zeta$.

Proof. By the previous proposition we have $|n|_{\mathcal{O}^{++}} \leq \zeta$ for every $n \in \text{field}(\mathcal{O}^{++})$.
q.e.d.

Having this established, we venture to show that all eventually writable ordinals embed into $\mathcal{O}^{++}$, thus showing the converse of Corollary 3.2.3. The proof follows that of the writable case, with some extra complications. The first thing to establish is that we have a sum operation on $\mathcal{O}^{++}$. For that we need the following

Technical Lemma 3.2.4.

(1) There is an eventually computable function F which on any input $(e, m, d, n) \in \mathbb{N}^4$ such that for some $b \in 2^{\mathbb{N}}$, $Q_d(n)\uparrow = b$ and $Q_e(m, b)\uparrow = a$, outputs a. Let $h : \mathbb{N}^3 \to \mathbb{N}$ be such that $Q_{h(e,m,d)}(n) \simeq F(e, m, d, n)$.

(2) There is an eventually computable $G\colon \mathbb{N}^3 \to \mathbb{N}$ such that

$$G(e, m, n) \simeq \begin{cases} m & \text{if } n = 1 \\ 2^{Q_e(m,k)} & \text{if } n = 2^k \\ 3 \cdot 5^{h(e,m,d)} & \text{if } n = 3 \cdot 5^d \\ 7 & \text{otherwise.} \end{cases}$$

Proof. (1) Suppose we are given $(e, m, d, n) \in \mathbb{N}^4$ such that $Q_d(n) = b$ and $Q_e(m, b) = a$. Consider the following. Let β be produced by the Ordinal

Production Machine. Compute $Q_{d,\beta}(n) = b^*$, and then $Q_{e,\beta}(b^*) = a^*$, both along β, and write a^* on the output tape if it is not already written there. From The Relativized Timing Theorem 3.1.11 *(2)*, it follows that a machine according to this algorithm will eventually write a.

(2) Given $(e, m, n) \in \mathbb{N}^3$, check n. If $n \neq 2^k$, write the required output on the output tape. If $n = 2^k$, then start approximating $Q_e(m, k)$. If b^* is such an approximation, check whether $b^* \in \mathbb{N}$. If so, then write 2^{b^*} if it is not already written there. q.e.d.

Proposition 3.2.5. *There is an eventually computable function $\hat{+}$ such that*

(1) $m, n \in \text{field}(\mathcal{O}^{++})$ *if and only if* $m \hat{+} n \in \text{field}(\mathcal{O}^{+})$
(2) *for all* $m, n \in \text{field}(\mathcal{O}^{++})$ *we have* $|m \hat{+} n|_{\mathcal{O}^+} = |m|_{\mathcal{O}^+} + |n|_{\mathcal{O}^+}$
(3) *if* $m, n \in \text{field}(\mathcal{O}^{++})$ *and* $n \neq 1$, *then* $m <_{\mathcal{O}^{++}} m \hat{+} n$

Proof. Let F be the function asserted to exist by the Technical Lemma 3.2.4 *(1)*. The s-m-n Theorem for eventual computability, Proposition 3.1.1, gives us a computable function $h\colon \mathbb{N}^3 \to \mathbb{N}$ such that $Q_{h(e,m,d)}(n) \simeq F(e, m, d, n)$. Using the s-m-n Theorem for eventual computability on the G of Lemma 3.2.4 *(2)*, we get an $I\colon \mathbb{N} \to \mathbb{N}$ such that $Q_{I(e)}(m, n) \simeq G(e, m, n)$. The Recursion Theorem for eventual computability then gives us the function $\hat{+}$ just like in the writable case. q.e.d.

As in the writable case, a sum operation on $\mathcal{O}^{++}$ gives a way of bounding every eventually writable subset of $\mathcal{O}^{++}$, namely: take the infinite sum over a.

Proposition 3.2.6. *There is a Turing computable function $F'\colon \mathbb{N} \to \mathbb{N}$ such that if $Q_e(0)\!\uparrow = a$, then*

$$\text{if } a \subseteq \text{field}(\mathcal{O}^{++}), \text{ then } F'(e) \in \text{field}(\mathcal{O}^{++}), \text{ and for all } n \in a, \\ |n|_{\mathcal{O}^{++}} < |F'(e)|_{\mathcal{O}^{++}}.$$

Proof. This is only a small adjustment to the writable case, Proposition 2.1.18. Given input $(e, n) \in \mathbb{N}^2$. $(\star)$ Wait for an ordinal β from the OPM, and compute an approximation $Q_{e,\omega\cdot\eta}(0) = a^*$. Then write this a^* on the input tape and apply $\hat{+}$ recursively to the the elements of the input tape, computing along β. After having applied $\hat{+}$ thus n times, write the result on the output tape if it is not already written there, and go to $(\star)$.

There is an ITTM without halting state Q_d according to this algorithm. The s-m-n Theorem for eventual computability gives an $I\colon \mathbb{N} \to \mathbb{N}$ such that $Q_{I(e)}(n) \simeq Q_d(e, n)$. Let $F'(e) := 3 \cdot 5^{I(e)}$. The rest of the proof now goes as in the writable case, Proposition 2.1.18. q.e.d.

Corollary 3.2.7. *There is an eventually computable function $F\colon 2^{\mathbb{N}} \rightharpoonup 2^{\mathbb{N}}$ such that for any eventually writable a we have*

$$\text{if } a \subseteq \text{field}(\mathcal{O}^{++}), \text{ then } F(a) \in \text{field}(\mathcal{O}^{++}), \text{ and for any } n \in a, \\ |n|_{\mathcal{O}^{++}} < |F(a)|_{\mathcal{O}^{++}}.$$

Proof. Given input a. $(\star)$ Wait for an ordinal β from the Ordinal Production Machine, then simulate all the computations $\{Q_e(0)\}_{e\in\mathbb{N}}$ along β. Let F' be the function of the previous Proposition 3.2.6. Output $F'(e)$ for the least e such that a is final in $\{Q_{e,\alpha}\}_{\alpha<\beta}$ and if a is final in $\{Q_{d,\alpha}(0)\}_{\alpha<\beta}$, then the stabilization point of $Q_e(0)$ is less than or equal to that of $Q_d(0)$ (given, as always, that this $F'(e)$ is not already written on the output tape). Go to $(\star)$. q.e.d.

Proposition 3.2.8. *There is a Turing computable function* $G\colon \mathbb{N} \to \mathbb{N}$ *such that if* $Q_e(0)\!\uparrow = a$ *and* $a \in \mathrm{WO}$*, then* $Q_{G(e)}(m) \in \mathrm{field}(\mathcal{O}^+)$ *for all* $m \in \mathrm{field}(a)$*, and*

$$m <_a n \text{ implies } |Q_{G(e)}(m)|_{\mathcal{O}^+} < |Q_{G(e)}(n)|_{\mathcal{O}^+}.$$

Proof. This requires the familiar adjustments to the writable case, Proposition 2.1.20: compute along some ordinal given by the Ordinal Production Machine. We omit the details. q.e.d.

Having all this established, we get

Theorem 3.2.9. $|\mathcal{O}^{++}| = \zeta$

Proof. That $|\mathcal{O}^{++}| \leq \zeta$ is established by Corollary 3.2.3.

For the other direction, let $a \in \mathrm{WO}$ be eventually writable. There is then an $e \in \mathbb{N}$ such that $Q_e(0)\!\uparrow = a$. Using Q_e and $Q_{G(e)}$, where G is the function of the previous Proposition 3.2.8, we can eventually write a real $Q_{G(e)}[a] := c \subseteq \mathrm{field}(\mathcal{O}^{++})$ such that $|a| \leq \sup\{|n|_{\mathcal{O}^{++}} : n \in c\}$. Now apply the F of Corollary 3.2.7 to the real c. We have $F(c) \in \mathrm{field}(\mathcal{O}^{++})$ and for all $n \in c$, $|n|_{\mathcal{O}^{++}} < |F(c)|_{\mathcal{O}^+}$. Hence, $|a| \leq |F(c)|_{\mathcal{O}^{++}} < |\mathcal{O}^{++}|$. q.e.d.

We end this section by observing how $\mathcal{O}^{++}$ naturally extends $\mathcal{O}^+$.

Corollary 3.2.10. *We have* $\mathcal{O}^+ \subseteq \mathcal{O}^{++}$*, and the tree coded by* $\mathcal{O}^{++}$ *is both taller and fatter than that coded by* $\mathcal{O}^+$*; thus, in fact,* $\mathcal{O}^+ \subsetneq \mathcal{O}^{++}$*.*

Proof. This is very similar to the proof of Corollary 2.1.22, and we omit the details. q.e.d.

3.3 The Accidentally Writable ITTM Degrees

In this section we first establish the ITTM degree of $\mathcal{O}^{++}$; this result is then much improved upon to a theorem stating that $\mathcal{O}^{++}$ is computably isomorphic to "the stabilization problem". As preliminaries we present some elementary facts and constructions from [5] and review thereafter the theory of accidentally writable degrees, developed by Welch, Hamkins, and Lewis. The set $\mathcal{O}^{++}$ will be seen to have the maximal such degree.

The weak jump. Define the weak jump $\cdot^\nabla$ on $\wp(2^{\mathbb{N}})$ by

$$A^\nabla := \{e \in \mathbb{N} : P_e^A(0)\!\downarrow\} \oplus A,$$

where $\oplus$ is some Δ_0-definable operation coding disjoint unions. That $A < A^\nabla$ then follows easily from the facts that A is computable from A, but h^A is not ITTM A-decidable. The set $A \subseteq 2^{\mathbb{N}}$ is explicitly included in the weak jump of A to make sure that A^∇ computes A; for cardinality reasons, some sets of reals A are namely too complex to be computed by any real, in particular too complex to be computed by $h^A := \{e \in \mathbb{N} : P_e^A(0)\!\downarrow\}$. It is easy to see, however, that for $a \in 2^{\mathbb{N}}$, we have $a \leq_m h^a$ via the function which takes $e \in \mathbb{N}$ to the program which asks whether e is in the oracle, halts if it is, goes into a loop if it is not. It follows that $a^\nabla \equiv_\infty h^a$.[2]

[2]There is also a strong jump, $\cdot^\blacktriangledown$, corresponding to the boldface halting problem H, discussed in Footnote 5. It is defined by setting $A^\blacktriangledown := \{(e, a) \in \mathbb{N} \times 2^{\mathbb{N}} : P_e^A(a)\!\downarrow\}$. The strong jump can be seen to jump much higher than the weak jump (see the Jump Iteration Theorem 6.12 of [5]). We will not have anything to say about the strong jump in this thesis.

Given a real $a \in \mathrm{WO}$ coding α, we define the iterate $A^{\nabla_a^\beta}$ for $\beta \leq \alpha$ inductively by

$$\begin{array}{lcl} A^{\nabla_a^{\beta+1}} & := & (A^{\nabla_a^\beta})^\nabla \\ A^{\nabla_a^\beta} & := & \{(n,b) \in \mathbb{N} \times 2^{\mathbb{N}} : \exists \gamma < \beta \, (b \in A^{\nabla_a^\gamma} \,\&\, |n|_a = \gamma)\}, \\ & & \text{for limit ordinals } \beta. \end{array}$$

Often we suppress mention of the real a coding α, thus only writing A^{∇^α}. As is common in degree theory, we denote the empty set by 0, thus writing, for instance, 0^∇ for h.

The accidentally and eventually writable degrees. An ITTM degree $[A]_{\equiv_\infty}$ is called accidentally/eventually writable if there is an accidentally/eventually writable real a such that $a \in [A]_{\equiv_\infty}$. Let $\mathbb{A}$ denote the set of accidentally writable degrees ordered by $<_\infty$, and let $\mathbb{E}$ denote the set of eventually writable degrees ordered by $<_\infty$. In their [6], Hamkins & Lewis characterizes the structures $\mathbb{A}$ and $\mathbb{E}$. We report here some of their results.

Classifying $\mathbb{A}$ and $\mathbb{E}$. We recall the notion of the accidentally writable norm mentioned earlier. If $Q_e(0)$ accidentally writes a, then we said that $|a|_e$ is to be the least ordinal α such that $Q_{e,\alpha}(0) = a$. For convenience, say that $|a|_e := \Sigma$ if $Q_e(0)$ does not accidentally write a. For any $a \in 2^{\mathbb{N}}$, we then define $|a|_{\mathrm{acc}} := \min\{|a|_e : e \in \mathbb{N}\}$. Note that by Proposition 3.1.7 we have $|a|_e < \Sigma$ if and only if $Q_e(0)$ accidentally writes a.

Proposition 3.3.1 (Hamkins & Lewis). *Let a and b be accidentally writable reals. If there are $d, e \in \mathbb{N}$ such that $|a|_d \leq |b|_e$, then $a \leq_\infty b$. In less precise prose: if a appears before b, then $a \leq_\infty b$.*

Proof. Suppose $|a|_d \leq |b|_e < \Sigma$. Note that $|b|_e$ is b-clockable: simply run $Q_e(0)$ until b appears, then halt. Hence, by Welch's Theorem 1.4.8, $|b|_e$ is b-writable, so as there are no gaps in the b-writable ordinals, $|a|_d$ is b-writable. But then, $a = Q_{d,|a|_d}(0)$ is b-writable as well. q.e.d.

Corollary 3.3.2 (Hamkins & Lewis). *The structures $\mathbb{A}$ and $\mathbb{E}$ are well-orders.*

Proof. We show that the set of accidentally writable reals under $\leq_\infty$ is a pre-well-order. From this it follows that $\mathbb{A}$ is a well-order, and – as every eventually writable real is also accidentally writable – that $\mathbb{E}$ is a well-order. Given accidentally writable reals a, b, there are $d, e \in \mathbb{N}$ such that $Q_d(0)$ and $Q_e(0)$ accidentally writes a and b respectively. Trivially, $|a|_d$ and $|b|_e$ are $\leq$-comparable, so a and b are $\leq_\infty$-comparable. Now consider a set A of accidentally writable reals. There is some $a \in A$ such that $|a|_{\mathrm{acc}} = \min\{|b|_{\mathrm{acc}} : b \in A\}$. By the previous proposition we then get $a \leq_\infty b$ for all $b \in A$. q.e.d.

The following proposition is both interesting in itself and helpful in deciding the height of the well-order $\mathbb{E}$.

Proposition 3.3.3 (Hamkins & Lewis). *If α is eventually writable, then 0^{∇^α} is eventually writable.*

Proof. The argument is a sort of priority argument. In parallel, eventually write α and 0^{∇^α} as follows. For any given approximation α^* to α, start writing $0^{\nabla^{\alpha^*}}$ by writing each slice 0^{∇^β}, for $\beta < \alpha^*$ in parallel. We give highest priority to the writing of α, so for any new approximation α^* to α we have to start anew with the writing of $0^{\nabla^{\alpha^*}}$. Further, for $\gamma < \beta \leq \alpha^*$, the writing of 0^{∇^γ} has higher priority than the writing of 0^{∇^β}, so that if some new element is added in the writing of 0^{∇^γ} all writings of 0^{∇^β} for $\beta > \gamma$ start anew.

Finally, we will be doing this with the true α, and then, after a while, the true 0^∇ will appear, then 0^{∇^2}, *etc.*, at last giving the whole 0^{∇^α}. q.e.d.

Proposition 3.3.4 (Hamkins & Lewis). *The well-order* $\mathbb{IE}$ *has order type* ζ.

Proof. Let $|\mathbb{IE}|$ denote the height of $\mathbb{IE}$. We want to show that $|\mathbb{IE}| = \zeta$. Let us first prove $\zeta \leq |\mathbb{IE}|$. If α is eventually writable, then, by the previous Proposition 3.3.3, $\{0^{\nabla^\beta} : \beta \leq \alpha + 1\}$ is a set of eventually writable reals which forms a well-order of height $\alpha + 1$ under $<_\infty$. Thus, we can not have $|\mathbb{IE}| < \zeta$.

By Proposition 3.3.1, $\mathbb{IE}$ embeds into the structure

$$(\{|a|_{\text{acc}} : a \text{ is eventually writable}\}, <)$$

which has height at most ζ by the Timing Theorem 3.1.7 *(2)*. q.e.d.

This classifies $\mathbb{IE}$. We know that $\mathbb{IE}$ is an initial segment of $\mathbb{A}\setminus$. From the next few propositions it follows that we only need to add one element to $\mathbb{IE}$ to get $\mathbb{A}\setminus$. In particular, $\mathbb{A}\setminus$ has a maximal element.

Let $Q_{e_0}(0)$ be the ITTM computation which simulates all computations $\{Q_d(0)\}_{d\in\mathbb{N}}$ on the scratch tape in parallel using some part Output d of the output tape as the simulated output tape of the computation $Q_d(0)$. Thus, for any limit ordinal α, $Q_{e_0,\alpha}(0)$ is a real coding all the reals $\{Q_{d,\alpha}(0)\}_{d\in\mathbb{N}}$.

Definition 3.3.5. *Let* $Q_{e_0}(0)$ *be the computation just described. Define* $z := Q_{e_0,\zeta}(0)$.

Lemma 3.3.6 (Welch, Hamkins & Lewis). *We have* $|z|_{e_0} = \zeta$*, and the least ordinal* $\alpha > \zeta$ *such that* $Q_{e_0,\alpha}(0) = z$ *is* Σ*. Moreover,* z *is not eventually writable.*

Proof. Notice that the range of the norm $|\cdot|_{\text{acc}}$ restricted to eventually writable reals is unbounded below ζ: for any $\alpha < \zeta$ we can eventually write a real a such that $|a|_{\text{acc}} > \alpha$ by eventually writing α, and then, by diagonalization, eventually writing a real a not in the list $\{Q_{d,\alpha}(0)\}_{d\in\mathbb{N}}$. As z computably codes all eventually writable reals (plus something more) it can thus not appear before stage ζ *in any computation*; in particular $|z|_{e_0} = \zeta$. Moreover, z can not be eventually writable in view of the Timing Theorem 3.1.7 *(2)*.

It remains to prove that z does not reappear in the computation $Q_{e_0}(0)$ before stage Σ. Note first that, by Lemma 3.1.5 and the fact that $|z|_{e_0} = \zeta$, any ω-sequence of the repeating ζ-snapshot of the computation $Q_{e_0}(0)$ is again the same ζ-snapshot. Now suppose that z reappeared before Σ. As Σ has certain elementary closure properties (in particular $\alpha < \Sigma$ implies $\alpha \cdot \omega < \Sigma$) it follows that there is an ω-sequence of repeating ζ-snapshots below Σ. Hence, the computation $Q_{e_0}(0)$ loops before Σ, and ζ is the stage at which it starts looping. But then we can use the Ordinal Production Machine to run $Q_{e_0}(0)$ along accidentally writable ordinals, and recognize ζ as the stage where the computation starts looping, and thus write ζ, a contradiction. q.e.d.

Lemma 3.3.7 (Welch, Hamkins & Lewis). *If $a \in 2^{\mathbb{N}}$ is accidentally writable and there is an $e \in \mathbb{N}$ such that $|a|_e \geq \zeta$, then $a \equiv_\infty z$.*

Proof. By Proposition 3.3.1 we get $z \leq_\infty a$ immediately.

As z does not reappear on the output tape in the computation $Q_{e_0}(0)$ before Σ it follows that Σ is z-clockable: just run Q_{e_0} until z appears for the second time. So by Welch's Theorem 1.4.8 and the fact that $|a|_e < \Sigma$ it follows that $|a|_e$ is z-writable. But then a is z-writable as well. Hence, a is ITTM z-decidable. q.e.d.

Corollary 3.3.8 (Welch, Hamkins & Lewis). *The well-order $\mathbb{A}$ has height $\zeta + 1$.*

Proof. Let a be accidentally writable. If $|a|_{\text{acc}} < \zeta$, then a is eventually writable and so is a member of some degree in $\mathbb{E}$; and if $|a|_{\text{acc}} \geq \zeta$, then $a \equiv_\infty z$. In particular, there is only one accidentally writable degree which is not eventually writable. q.e.d.

This completes the classification of the structures $\mathbb{E}$ and $\mathbb{A}$.

An aside. Let us pause to note a surprising result we anticipated above. The result follows from the proof of Lemma 3.3.6. Denote by $\|Q_e(a)\|$ the ordinal at which the computation $Q_e(a)$ starts looping. We write $\|Q_e\|$ for $\|Q_e(0)\|$.

Observation 3.3.9 (Welch, Hamkins). *Either $\|Q_e\| < \lambda$ or $\|Q_e\| = \zeta$.*

Proof. Suppose $\lambda \leq \|Q_e\| < \zeta$ for some $e \in \mathbb{N}$. Then, the stage at which the computation $Q_e(0)$ repeats is also less than ζ, so using the Ordinal Production Machine we can write $\|Q_e\|$, a contradiction. q.e.d.

This observation does of course not contradict the fact that the set

$$\{|a|_{\text{acc}} : a \text{ is accidentally writable}\}$$

is unbounded in ζ; it only implies that if the output tape of the computation $Q_e(0)$ does not stabilize before λ, then $Q_e(0)$ does not enter into a loop before stage ζ. The moral is that for a computation to stabilize on a constant output tape is very different from it entering a loop.

Classifying $\mathcal{O}^{++}$. We now show that $\mathcal{O}^{++}$ has the maximal accidentally writable degree. We do that by showing that $\mathcal{O}^{++}$ is accidentally but not eventually writable.

Lemma 3.3.10. *Using a code for ζ as an oracle, we can write $\mathcal{O}^{++}$.*

Proof. By The Relativized Timing Theorem 3.1.11, if we have ζ available, then we can record those $e \in \mathbb{N}$ such that $\mathbb{N} \subseteq Q_e$ by simply simulating all computations $\{Q_e(k)\}_{e,k\in\mathbb{N}}$ for ζ many steps, and then record those e's for which $Q_e(k)$ has stabilized before ζ for all $k \in \mathbb{N}$.

After having done this, we can write $\mathcal{O}^{++}$ up to height ζ as follows. First write $\{\langle 2{\uparrow\uparrow}k, 2{\uparrow\uparrow}(k+1)\rangle\}_{k\in\mathbb{N}}$ on the output tape. Then go into the following recursive subprocedure, each part of which can be done in ω many steps: ($\star$) Let a^* be the current approximation to $\mathcal{O}^{++}$ written on the output tape. Find the e's for which $\{Q_e(k)\}_{k\in\mathbb{N}}$ forms an increasing cofinal sequence in a^*; then, for

these e's write $\{\langle Q_e(k), 3 \cdot 5^e \rangle\}_{k \in \mathbb{N}}$ on the output tape, and close off under $\mathcal{O}^{++}$-successor and transitivity. In the limit go back to $(\star)$. Repeat this procedure ζ many times.

It is clear that this procedure will write $\mathcal{O}^{++}$ up to height $\omega \cdot \zeta = \zeta$. But by Corollary 3.2.3, the height of $\mathcal{O}^{++}$ is restricted by ζ, so in fact, this procedure writes the true $\mathcal{O}^{++}$. q.e.d.

Proposition 3.3.11. *The set $\mathcal{O}^{++}$ is accidentally writable.*

Proof. For any ordinal β given by the Ordinal Production Machine, do the construction in Lemma 3.3.10 with ζ replaced by β. We may assume that the Ordinal Production Machine at some point produces ζ, and at that point we will be accidentally writing $\mathcal{O}^{++}$ on the output tape. q.e.d.

Proposition 3.3.12. *The set $\mathcal{O}^{++}$ is not eventually writable.*

Proof. Note first that by using an algorithm similar to that described in the proof of The Count Through Theorem 1.4.4, with $\mathcal{O}^{++}$ as an oracle we can clock an ordinal $\alpha \geq \zeta$. Thus, by Welch's Theorem 1.4.8, α is $\mathcal{O}^{++}$-writable; thus, ζ is $\mathcal{O}^{++}$-writable. But then, if $\mathcal{O}^{++}$ were eventually writable, ζ would be eventually writable as well, and that would be a contradiction. q.e.d.

Corollary 3.3.13. *The set $\mathcal{O}^{++}$ has the maximal accidentally writable degree.*

Proof. By the two previous propositions and The Timing Theorem 3.1.7, $\mathcal{O}^{++}$ must have accidentally writable norm somewhere between ζ and Σ. Thus, by Lemma 3.3.7, $\mathcal{O}^{++}$ has the maximal accidentally writable degree. q.e.d.

To get an even better result, consider

$$s := \{e \in \mathbb{N} : Q_e(0)\!\uparrow\},$$

that is, the set of programs that eventually writes a real, the set we have been calling the "stabilization problem". The set s is thus the "halting" problem of eventual computability. Just as $\mathcal{O}^{+} \equiv_1 h$, we now prove that $\mathcal{O}^{++} \equiv_1 s$.

Lemma 3.3.14. *We have $s \leq_1 \mathcal{O}^{++}$.*

Proof. Define a function $G : \mathbb{N}^2 \rightharpoonup \mathbb{N}$ by

$$G(e, n) := \begin{cases} 2\!\uparrow\!\uparrow n & \text{if } e \in s \\ \uparrow & \text{if } e \notin s \end{cases}$$

It is easy to see that G is eventually computable, so just as in the case of $\mathcal{O}^{+}$ and h, Theorem 2.1.13, we can use the s-m-n Theorem to obtain a Turing computable total injective $F\colon \mathbb{N} \to \mathbb{N}$ such that

$$e \in s \Leftrightarrow \langle 1, 3 \cdot 5^{F(e)} \rangle \in \mathcal{O}^{+},$$

Hence, s is 1-reducible to $\mathcal{O}^{++}$ via the function $n \mapsto \langle 1, 3 \cdot 5^{F(n)} \rangle$. q.e.d.

To show the other direction we introduce a class of new families of functions that generalizes the family of eventually computable functions.

We say that a computation $Q_e(a)$ is locally stable at a limit ordinal α if there is some $b \in 2^{\mathbb{N}}$ and some $\beta < \alpha$ such that $Q_{e,\gamma}(a) = b$ for all γ with $\beta < \gamma < \alpha$, that is, if b is final in the sequence $\{Q_{e,\beta}(a)\}_{\beta<\alpha}$.

Now, for any ordinal $\alpha < \omega_1$ and $e \in \mathbb{N}$, let $Q_{e,<\alpha}$ be defined by setting $Q_{e,<\alpha}(a) := b$ if and only if the computation $Q_e(a)$ is locally stable at α with $Q_{e,\alpha}(a) = b$, and by letting $Q_{e,<\alpha}(a)$ be undefined if $Q_e(a)$ is not locally stable at α. For any $\alpha < \omega_1$, we let $\mathcal{Q}^\alpha$ be the family $\{Q_{e,<\alpha}\}_{e\in\mathbb{N}}$.

Now recall that $\mathcal{O}^{\mathcal{Q}^\alpha}$ is the set obtained by replacing clause *(4)* in Definition 1.3.1 by

(4) *If* $\mathbb{N} \subseteq \mathrm{dom}(Q_{e,<\alpha})$*, and we have* $Q_{e,<\alpha}(n) <_{\mathcal{O}^{\mathcal{Q}^\alpha}} Q_{e,<\alpha}(n+1)$ *for all* $n \in \mathbb{N}$*, then* $Q_{e,<\alpha}(n) <_{\mathcal{O}^{\mathcal{Q}^\alpha}} 3 \cdot 5^e$ *holds for all* $n \in \mathbb{N}$*.*

Define $(\mathcal{O}^{++})_\alpha := \mathcal{O}^{\mathcal{Q}^\alpha}$.

Lemma 3.3.15. *Given a code for* $\alpha < \omega_1$ *we can write* $(\mathcal{O}^{++})_\alpha$*.*

Proof. Given a code for α, we can compute all the values $\{Q_{e,<\alpha}(k)\}_{e,k\in\mathbb{N}}$; and with all these values at hand it is straightforward how to write $\mathcal{O}^{\mathcal{Q}^\alpha}$. q.e.d.

Lemma 3.3.16. *For any ordinal* β *such that* $\zeta \leq \beta < \Sigma$*, we have*

$$\mathcal{O}^{++} = \bigcap_{\beta<\alpha<\Sigma} (\mathcal{O}^{++})_\alpha.$$

Proof. Let β be such that $\zeta \leq \beta < \Sigma$. In view of The Relativized Timing Theorem 3.1.11 it is clear that $\mathcal{O}^{++} = (\mathcal{O}^{++})_\zeta \subseteq (\mathcal{O}^{++})_\alpha$ for all α such that $\beta < \alpha < \Sigma$.

Now suppose that $\langle 1, 3 \cdot 5^e\rangle \in (\mathcal{O}^{++})_\alpha$ for every α such that $\beta < \alpha < \Sigma$. This means that all the computations $\{Q_e(k)\}_{k\in\mathbb{N}}$ are locally stable at Σ. By an argument very similar to that in the proof of Lemma 3.1.4 we can show that all the computations $\{Q_e(k)\}_{k\in\mathbb{N}}$ must in fact have stabilized before ζ. Thus $\langle 1, 3 \cdot 5^e\rangle \in \mathcal{O}^{++}$. It follows that $\bigcap_{\beta<\alpha<\Sigma} (\mathcal{O}^{++})_\alpha \subseteq \mathcal{O}^{++}$. q.e.d.

Theorem 3.3.17. *The sets* $\mathcal{O}^{++}$ *and* s *are computably isomorphic.*

Proof. By Myhill's Isomorphism Theorem 1.2.3 and Lemma 3.3.14 it suffices to prove $\mathcal{O}^+ \leq_1 s$.

To that end, consider the following algorithm. Given input $(n, a) \in \mathbb{N} \times 2^{\mathbb{N}}$ start the Ordinal Production Machine. $(\star)$ For any ordinal α given by the Ordinal Production Machine write $(\mathcal{O}^{++})_\alpha$ on some part of the scratch tape. This is possible by Lemma 3.3.15. Then, if $n \in (\mathcal{O}^{++})_\alpha$ just go back to $(\star)$. If, however, $n \notin (\mathcal{O}^{++})_\alpha$, then flash a flag on the output tape, that is, in some cell of the output tape write a 1 and then a 0; then go to $(\star)$.

Surely, some ITTM program Q_e without halting state works according to this algorithm. The s-m-n Theorem then gives some Turing computable total injective $F\colon \mathbb{N} \to \mathbb{N}$ such that $Q_{F(n)}(a) = Q_e(a)$. Further, at stage ζ, the Ordinal Production Machine goes into a loop and from that point on it is only producing ordinals lying between β and Σ for some $\beta \geq \zeta$. It is then easy to see that

$$n \in \bigcap_{\beta_0<\alpha<\Sigma} (\mathcal{O}^{++})_\alpha \Leftrightarrow F(n) \in s,$$

for β_0 the supremum of these β's. Together with Lemma 3.3.16 this implies that $\mathcal{O}^{++} \leq_1 s$. q.e.d.

This completes our classification of $\mathcal{O}^{++}$.

Linear orders of height ζ. Before ending this section, let us note that, in analogy with the writable case, we have

Proposition 3.3.18. *There is a set $\mathscr{Z} \subseteq \mathbb{N}$ coding a well-order of height ζ, and which is computably isomorphic to $\mathcal{O}^{++}$.*

Proof. Define $\langle d, e\rangle \in \mathscr{Z}$ if and only if $Q_d(0)$ stabilizes at stage α, $Q_e(0)$ stabilizes at stage β, and we have either $\alpha < \beta$ or $\alpha = \beta$ and $d < e$. By The Timing Theorem 3.1.7, the set $\mathscr{Z}$ then codes a well-order $<_{\mathscr{Z}}$ of height at most ζ. To show that the height of $<_{\mathscr{Z}}$ equals ζ it suffices to show that every $\alpha < \zeta$ embed into $\mathscr{Z}$. To that end, let $\alpha < \zeta$. The priority method as used in the proof of Proposition 3.3.3 showing that 0^{∇^α} is eventually writable can then be used to show that there are α many different stabilization ordinals. Thus, we have $|\mathscr{Z}| = \zeta$.

Using Myhill's Isomorphism Theorem 1.2.3 it remains to show that $\mathscr{Z}$ and s 1-reduce to each other. Now, it is straightforward to see that the function $1\restriction \mathscr{Z}$ is eventually computable. Use of the s-m-n Theorem then shows that there is a Turing computable $F\colon \mathbb{N} \to \mathbb{N}$ such that $e \in \mathscr{Z}$ if and only if $Q_{F(e)}(0)$ stabilizes; thus $\mathscr{Z} \leq_1 s$. For the other direction, let $F\colon \mathbb{N} \to \mathbb{N}$ be a Turing computable function taking any $e \in \mathbb{N}$ to a code $F(e)$ for a program $Q_{F(e)}$ which stabilizes on input 0 only if $Q_e(0)$ stabilizes, and in that case the computation $Q_{F(e)}(0)$ stabilizes at some stage after $Q_e(0)$ has stabilized. Then $e \in s \Leftrightarrow \langle e, F(e)\rangle \in \mathscr{Z}$; thus $s \leq_1 \mathscr{Z}$. q.e.d.

3.4 Finite Time $\mathcal{O}^{++}$

Before continuing our investigations in infinite time computability, we lower ourselves down to the realm of finite time. We start out in this lower realm by noticing that eventual computability makes sense also for finite time Turing machines. We call the notion of computability that this gives rise to for finite time eventual computability. As will be clear to the computability theorist, this is no novel notion of computability – in fact, by Shoenfield's Limit Lemma (see 3.4.2 below), finite time eventual computability is just $0'$-computability. In this section we observe the consequences of this fact for $\widehat{\mathcal{O}}$, the analogue of Kleene's $\mathcal{O}$ for the family of finite time eventually computable functions. We shall see that $\widehat{\mathcal{O}}$, even though it is fatter than $\mathcal{O}$ reaches no higher than $\mathcal{O}$. As the height of infinite time eventual computability $\mathcal{O}$ – that is, the height of $\mathcal{O}^{++}$ – is much higher than that of plain infinite time computability $\mathcal{O}$ – that is, the height of $\mathcal{O}^{+}$ – this can be taken to show a certain divergence between finite time and infinite time computability.

Finite time eventual computability. We use the same enumeration $\{Q_e\}_{e\in\mathbb{N}}$ of programs without halting state as before. For $n, k \in \mathbb{N}$, recall that $Q_{e,k}(n)$ denotes the content of the output tape of the computation $Q_e(n)$ at stage k. Recall further that the computation $Q_e(k)$ is said to be locally stable at ω if

there is some $k \in \mathbb{N}$ such that for every $m \in \mathbb{N}$ with $m \geq k$ we have $Q_{e,m}(n) = Q_{e,k}(n)$. We then say that a partial function $F\colon \mathbb{N} \rightharpoonup \mathbb{N}$ is finite time eventually computable if there is an $e \in \mathbb{N}$ such that for all $n \in \mathrm{dom}(F)$, the computation $Q_e(n)$ is locally stable at ω and $Q_{e,\omega}(n) = F(n)$, and for all $n \notin \mathrm{dom}(F)$ the computation $Q_e(n)$ is not locally stable at ω. Finite time eventual computability thus captures the idea of a (finite time) Turing machine computation which never halts, but which nevertheless has a constant output tape from some point onwards.

We notice that, in the notation of the last section, the family of finite time eventually computable functions is just the family $\mathfrak{Q}^\omega$. We denote by $\hat{Q}_e$ the partial function finite time eventually computed by the program Q_e.

Computable approximations. The notion of computable approximations was introduced by Shoenfield in his [20] , and has become a standard of classical computability theory. A sequence $\{F_k\}_{k\in\mathbb{N}}$ of partial functions on $\mathbb{N}$ is said to be an approximation to $F\colon \mathbb{N} \rightharpoonup \mathbb{N}$, written $\lim F_k = F$, if for all $n \in \mathrm{dom}(F)$, there is an $m \in \mathbb{N}$ such that for all $k \geq m$ we have $F_k(n) \simeq F(n)$, and for all $n \notin \mathrm{dom}(F)$, there is no $m \in \mathbb{N}$ such that for all $k \geq m$ we have $F_k(n) = F_m(n)$. The sequence $\{F_k\}_{k\in\mathbb{N}}$ is said to be computable if there is some Turing computable $G\colon \mathbb{N}^2 \rightharpoonup \mathbb{N}$ such that $G(k,n) \simeq F_k(n)$ for all $k, n \in \mathbb{N}$.

As is intuitively clear, finite time eventual computability is the same as having a computable approximation.

Observation 3.4.1. *Let $F\colon \mathbb{N} \rightharpoonup \mathbb{N}$. Then F is finite time eventually computable if and only it has a computable approximation. Moreover, there are injective computable functions f' and g' such that $\hat{Q}_e(k) = \lim_n \{g'(e)\}(n,k)$ and $\lim_n \{e\}(n,k) = \hat{Q}_{f'(e)}(k)$.*

Proof. Let $F\colon \mathbb{N} \rightharpoonup \mathbb{N}$. Suppose first that F is finite time eventually computable, computed by Q_e. Define $G : \mathbb{N}^2 \to \mathbb{N}$ by

$$(k,n) \mapsto Q_{e,k}(n).$$

Then G is Turing computable: given (k,n), run the computation $Q_e(n)$ for k steps, then halt. If we set $F_k(n) := G(k,n)$, it is then clear that $\lim_k F_k = F$.

Now suppose that F has a computable approximation $\{F_k\}_{k\in\mathbb{N}}$, where $F_k(n) = G(k,n)$ for some Turing computable G. An algorithm to finite time eventually compute F is then: given n compute in parallel all $\{G(k,n)\}_{k\in\mathbb{N}}$. If and when it is found that $G(k,n)\downarrow$, write $G(k,n)$ on the output tape if it is not already written there. As $\{F_k\}_{k\in\mathbb{N}}$ is approximating F, there is some point from which the execution of this algorithm has a constant output tape, namely $F(n)$.

The existence of the functions f' and g' follows easily from the uniformity in the proof just given, by standard use of s-m-n. q.e.d.

Shoenfield's Limit Lemma. We assume some familiarity with the notion of oracle Turing computations, and the related notion of relativized computability. Let $\{\tilde{e}\}^A$ denote the partial function $F\colon \mathbb{N} \rightharpoonup \mathbb{N}$ computed by the e-th oracle Turing machine on oracle $A \subseteq \mathbb{N}$. A set $A \subseteq \mathbb{N}$ is said to B-computable if $\chi_A \simeq \{\tilde{e}\}^B$ for some $e \in \mathbb{N}$. We say that $F\colon \mathbb{N} \rightharpoonup \mathbb{N}$ is B-computable if $F \simeq \{\tilde{e}\}^B$ for some $e \in \mathbb{N}$. Shoenfield proved in his [20] that

Shoenfield's Limit Lemma 3.4.2. *Let $F\colon \mathbb{N} \rightharpoonup \mathbb{N}$. Then F has a computable approximation if and only if F is $0'$-computable. Moreover, there are injective computable functions f and g such that $\{\tilde{e}\}^{0'}(k) = \lim_n \{f(e)\}(n,k)$, and $\lim_n \{e\}(n,k) = \{\tilde{g(e)}\}^{0'}(k)$.*

Proof. It is surely enough to prove the existence of the functions f and g. For that, we follow the proof given in Soare's book (see [21] Theorem 3.3), but make explicit the uniformity that is needed to get the required functions f and g. Let us first show the existence of f.

Fix an index e_0 such that $\{e_0\}$ is a computable approximation to $0'$, that is, for each n, $\{e_0\}(n)$ codes a finite subset of $0'$, $m < n$ implies $\{e_0\}(n) \subseteq \{e_0\}(m)$, and $\cup_n \{e_0\}(n) = 0'$. Then consider a function

$$(e,n,k) \mapsto \begin{cases} \{\tilde{e}\}_n^{\{e_0\}(n)}(k) & \text{if } \{e_0\}(n)_n(k)\downarrow \\ 0 & \text{otherwise} \end{cases}$$

Applying The s-m-n Theorem here then gives the required f.

To show the existence of g, consider the following algorithm. Given (e,k), start producing natural numbers $n = 0, 1, 2, \ldots$. Halt with output $\{e\}(n,k)$ for the first n found such that there is no $m > n$ with $\{e\}(m,k) \neq \{e\}(m+1,k)$. By the fact that $0'$ is a complete Σ^0_1 set, and the relativized Church's Thesis, there is a $0'$-computable function acting according to this algorithm. The s-m-n Theorem now gives the required g. q.e.d.

Thus, finite time eventual computability coincides with $0'$-computability. Let us notice that no similar state of affairs obtains in the infinite time setting.

Observation 3.4.3. *There is no real $a \in 2^{\mathbb{N}}$ such that a function $F\colon \mathbb{N} \rightharpoonup \mathbb{N}$ is* ITTM *a-computable if and only if F is eventually computable.*

Proof. Let $a \in 2^{\mathbb{N}}$ be arbitrary. With the danger of insulting the reader: either a is eventually writable, or it is not. If a is not eventually writable, then χ_a, the characteristic function of a, is not eventually computable; but χ_a is surely a-computable. If a, on the other hand, is eventually writable, then so is $a^\triangledown$ by Proposition 3.3.3; but $\chi_{a^\triangledown}$ is not a-computable. q.e.d.

The hyperdegrees and Spector's Theorem. Recall that $\Sigma^i_n[A]$ denotes the class of sets definable by a Σ^i_n formula possibly using $A \subseteq \mathbb{N}$ as a parameter; similarly for $\Pi^i_n[A]$. If $A \subseteq \mathbb{N}$ is definable by both a $\Sigma^1_1[B]$-formula and a $\Pi^1_1[B]$-formula, then A is said to be hyperarithmetic in B, written $A \leq_{\mathrm{HYP}} B$. By the relativized Kleene's Normal Form Theorem, we have $A \leq_{\mathrm{T}} B$ if and only if A is both $\Sigma^0_1[B]$-definable and $\Pi^0_1[B]$-definable; hence the relation $\leq_{\mathrm{HYP}}$ is a weakening of the relation $\leq_{\mathrm{T}}$.

It is not difficult to see that $\leq_{\mathrm{HYP}}$ is a transitive relation. As $\leq_{\mathrm{HYP}}$ is clearly reflexive, setting $A \equiv_{\mathrm{HYP}} B$ if and only if $A \leq_{\mathrm{HYP}} B$ and $B \leq_{\mathrm{HYP}} A$ we obtain yet another equivalence relation on $2^{\mathbb{N}}$. The equivalence classes under $\equiv_{\mathrm{HYP}}$ are called the hyperdegrees.

For any set $A \subset \mathbb{N}$, we define $\mathcal{O}^A$ to be the analogue of Kleene's $\mathcal{O}$ for the family of A-computable functions. The jump on the hyperdegress, the hyperjump, is then defined to be the function $A \mapsto \mathcal{O}^A$.

The ordinal ω_1^A is defined to be the supremum of all the A-computable ordinals. Kleene's proof that $|\mathcal{O}| = \omega_1^{\mathrm{CK}}$ then generalizes straightforwardly to a proof showing that $|\mathcal{O}^A| = \omega_1^A$. Spector proved in his [22] that

Theorem 3.4.4 (Spector). *If $A \leq_{\mathrm{HYP}} B$, then $\omega_1^A \leq \omega_1^B$*

Proof. See Theorem I.7.3. in Sacks' book [19]. q.e.d.

Theorem 3.4.4 has the surprising consequence that

Corollary 3.4.5. $\sup\{\alpha : \alpha \text{ is coded by some } a \in \Delta_1^1\} = \omega_1^{\mathrm{CK}}$.

Proof. Just note that for any $A, B \in \mathrm{HYP} = \Delta_1^1$ we have $A \equiv_{\mathrm{HYP}} B$. q.e.d.

Kleene's $\mathcal{O}$ for finite time eventual computability. Let $\widehat{\mathcal{O}}$ denote the analogue of Kleene's $\mathcal{O}$ for finite time eventual computability, that is, $\widehat{\mathcal{O}} := (\mathcal{O}^{++})_\omega$ in the notation introduced above. As finite time eventual computability and $0'$-computability are extensionally equivalent notions, we expect that $\mathcal{O}^{0'}$ and $\widehat{\mathcal{O}}$ are in some sense equal. We make this precise by showing that $\widehat{\mathcal{O}}$ and $\mathcal{O}^{0'}$ are both computably isomorphic and order-isomorphic.

Proposition 3.4.6. *The sets $\widehat{\mathcal{O}}$ and $\mathcal{O}^{0'}$ are computably isomorphic.*

Proof. We argue that both $\mathcal{O}^{0'} \leq_1 \widehat{\mathcal{O}}$ and $\widehat{\mathcal{O}} \leq_1 \mathcal{O}^{0'}$ hold; then the proposition follows from Myhill's Isomorphism Theorem. We restrict ourselves to showing that $\mathcal{O}^{0'} \leq_1 \widehat{\mathcal{O}}$, – the proof of which is an exercise in the use of the Recursion Theorem – the other direction being completely analogous. Let f and f' be injective recursive functions such that $\{\tilde{e}\}^{0'}(k) = \lim_n\{f(e)\}(n,k)$ and $\lim_n\{e\}(n,k) = \hat{Q}_{f'(e)}(k)$. Then $f' \circ f$ is an injective Turing computable function such that $\{\tilde{e}\}^{0'} \simeq \hat{Q}_{f'\circ f(e)}$. Now let $h\colon \mathbb{N} \to \mathbb{N}$ be an injective Turing computable function such that $\hat{Q}_{h(e,d)}(k) \simeq \{e\}(\hat{Q}_{(f'\circ f)(d)}(k))$. The existence of such an h is immediate from the s-m-n Theorem. Then, define a partial recursive function by

$$(e,n) \mapsto \begin{cases} 1 & \text{if } n = 1 \\ 2^{\{e\}(k)} & \text{if } n = 2^k \\ 3 \cdot 5^{h(e,d)} & \text{if } n = 3 \cdot 5^d \\ 7n & \text{otherwise.} \end{cases}$$

The s-m-n and Recursion Theorem then gives a Turing computable function $F\colon \mathbb{N} \rightharpoonup \mathbb{N}$ with index e_0 such that

$$F(n) \simeq \begin{cases} 1 & \text{if } n = 1 \\ 2^{F(k)} & \text{if } n = 2^k \\ 3 \cdot 5^{h(e_0,d)} & \text{if } n = 3 \cdot 5^d \\ 7n & \text{otherwise.} \end{cases}$$

One can argue by induction that this F is, in fact, total. Similarly, one can argue by induction on m that $m < n$ implies $F(m) \neq F(n)$. By $<_{\mathcal{O}^{0'}}$-induction on n one sees that $n \in \mathrm{field}(\mathcal{O}^{0'})$ and $m <_{\mathcal{O}^{0'}} n$ implies $F(n) \in \mathrm{field}(\widehat{\mathcal{O}})$ and $F(m) <_{\widehat{\mathcal{O}}} F(n)$. Let us do the induction step where $n = 3 \cdot 5^d$. Suppose $m <_{\mathcal{O}^{0'}} 3 \cdot 5^d$. We then have $\{\hat{d}\}^{0'}(k) <_{\mathcal{O}^{0'}} \{\hat{d}\}^{0'}(k+1)$ for all $k \in \mathbb{N}$ and $m <_{\mathcal{O}^{0'}} \{d\}(k)$

for some $k \in \mathbb{N}$. Using the property of h and the induction hypothesis we find that $3 \cdot 5^{h(e_0,d)} \in \text{field}(\widehat{\mathcal{O}})$ and $F(m) <_{\widehat{\mathcal{O}}} 3 \cdot 5^{h(e_0,d)} = F(n)$. To complete the proof, it only remains to show that $F(n) \in \text{field}(\widehat{\mathcal{O}})$ and $F(m) <_{\widehat{\mathcal{O}}} F(n)$ implies $n \in \text{field}(\mathcal{O}^{0'})$ and $m <_{\mathcal{O}^{0'}} n$. This is done, similarly, by induction on $<_{\widehat{\mathcal{O}}}$, using the property of h for the limit case. q.e.d.

Corollary 3.4.7. *We have $\mathcal{O} \subseteq \widehat{\mathcal{O}}$, and the tree coded by $\widehat{\mathcal{O}}$ is fatter than that coded by coded by $\mathcal{O}$; thus, in fact, $\mathcal{O} \subsetneq \widehat{\mathcal{O}}$.*[3]

Proof. That $\mathcal{O} \subseteq \widehat{\mathcal{O}}$ is clear from the way we have coded Turing machine programs without halting state. And, as the strictly increasing function enumerating $0'$ is $0'$-computable, the proof of Corollary 2.1.22 showing that $\mathcal{O}^+$ is fatter than $\mathcal{O}$ also shows that $\widehat{\mathcal{O}}$ is fatter than $\mathcal{O}$. q.e.d.

The height of $\widehat{\mathcal{O}}$. In the proof of Proposition 3.4.6, the function F witnessing $\mathcal{O}^{0'} \leq_1 \widehat{\mathcal{O}}$ was also proved to be an embedding of $(\mathbb{N}, <_{\mathcal{O}^0})$ into $(\mathbb{N}, <_{\widehat{\mathcal{O}}})$. Similarly, one can show that the function witnessing $\widehat{\mathcal{O}} \leq_1 \mathcal{O}^{0'}$ is an embedding of $(\mathbb{N}, <_{\widehat{\mathcal{O}}})$ into $(\mathbb{N}, <_{\mathcal{O}^{0'}})$. This, of course, implies that

Proposition 3.4.8. $|\widehat{\mathcal{O}}| = |\mathcal{O}^{0'}|$*; in consequence,* $|\widehat{\mathcal{O}}| = \omega_1^{\text{CK}}$.

Proof. As $\mathcal{O}^{0'}$ and $\widehat{\mathcal{O}}$ embed into each other, none of them can be higher than the other. The second claim follows from Spector's Theorem 3.4.4. q.e.d.

Thus, $\widehat{\mathcal{O}}$ is fatter than $\mathcal{O}$, but not taller.

3.5 Putting $\mathcal{O}^{++}$ into context.

The main point of this section is to show that there are, in fact, many naturally defined sets which are computably isomorphic to $\mathcal{O}^{++}$. All the new results of this section will follow from the fact of Theorem 3.3.17 that $\mathcal{O}^{++}$ and s are computably isomorphic, and from the work of Welch ([23], [24], [25]) and Burgess ([1]), some parts of which will be presented here.

The reader will notice that in this section some of our proofs will consist of a bit more hand-waving than what is the case in other parts of this thesis. This is especially true when we introduce Gödel's constructible hierarchy below. But as the results presented here are treated elsewhere, and as the reader has only finite time on his hand, we found it legitimate to be slightly hand-wavy.

Lightface versions of WO. For $e \in \mathbb{N}$, define

$$R_e := \{(m,n) \in \mathbb{N}^2 \,:\, \{e\}(\langle m,n\rangle) = 1\},$$

[3] We should point out that with the way we have defined finite time eventual computable functions here, we will not have $\widehat{\mathcal{O}} \subseteq \mathcal{O}^{++}$. This is because, for instance, some programs Q_e are such that for all $k \in \mathbb{N}$, $Q_e(k)$ is locally stable at ω with $Q_{e,\omega}(k) = 2{\uparrow\uparrow}k$, but, still, $Q_e(k){\Uparrow}$, so $3 \cdot 5^e \notin \text{field}(\mathcal{O}^{++})$. If we defined finite time eventual computability by requiring that $Q_e(k)$ be fully stabilized by stage ω, then we would get a version of $\mathcal{O}$ included in $\mathcal{O}^+$ – we did not like, however, the idea of defining a notion concerning *finite* time computability using an ordinal greater than ω.

the binary relation coded by $\{e\}$. Markwald introduced in his [14] the set W of indices e such that R_e is a well-order. Spector then proved in his [22] that W is a complete Π^1_1 set. The set W is often viewed as a lightface version of WO.

Certainly, for any family $\mathcal{F} = \{F_e\}_{e\in\mathbb{N}}$ of functions on the natural numbers, we can define the set $W^{\mathcal{F}}$ of indices such that F_e codes a well-ordering. Following the typography of this thesis, we let W^+ be the set of indices $e \in \mathbb{N}$ such that $P_e(0)\downarrow = a$ and $a \in \mathrm{WO}$, and we let W^{++} be the set of indices $e \in \mathbb{N}$ such that $Q_e(0)\uparrow = a$ and $a \in \mathrm{WO}$.

Proposition 3.5.1. *The sets W^{++} and s are computably isomorphic, as are the sets W^+ and h.*

Proof. We do the case for W^{++} and s, the other being very similar, and even simpler. To show that $s \leq_1 W^{++}$, consider, for any $e \in \mathbb{N}$, the machine $Q_{\hat{e}}$ which on input 0 does: $(\star)$ for an α given by the Ordinal Production Machine, check whether the computation $Q_e(0)$ is locally stable at α; if the check is positive, then write on the output tape the least ordinal γ such that $Q_{e,\beta} = Q_{e,\alpha}$ for all β with $\gamma \leq \beta < \alpha$; if the check is negative, then flash the first cell of the output tape. Thereafter, in both cases, wait for a new ordinal from the Ordinal Production Machine and go back to $(\star)$. Standard use of s-m-n now implies that $s \leq_1 W^{++}$ (in the future we will often suppress mention of s-m-n usage).

For the other direction, that is $W^{++} \leq_1 s$, consider, for any $e \in \mathbb{N}$, the machine $Q_{\check{e}}$ which on input 0 simulates $Q_e(0)$ and at any limit stage η of the simulation checks whether $Q_{e,\eta}(0) \in \mathrm{WO}$. If that check is positive, then the machine just continues the simulation, but if the check is negative the machine flashes the first cell of the output tape before it continues the simulation. q.e.d.

Corollary 3.5.2. *The sets $\mathcal{O}^{++}$ and W^{++} are computably isomorphic, as are the sets $\mathcal{O}^+$ and W^+.*

Proof. The eighth Clay Millennium Problem. q.e.d.

Arithmetically quasi-inductive sets. We look at Burgess' generalization of inductive definitions, so-called quasi-inductive definitions. For η a limit ordinal and $\{A_\alpha\}_{\alpha<\eta}$ a sequence of sets, define

$$\liminf\{A_\alpha\}_{\alpha<\eta} := \bigcup_{\alpha<\eta}\bigcap_{\alpha<\beta<\eta} A_\beta.$$

A sequence $\{A_\alpha\}_{\alpha<\omega_1}$ such that

$$A_\eta = \liminf\{A_\alpha\}_{\alpha<\eta}$$

holds for all limit ordinals η, is called a quasi-inductive definition. Notice that any $\subseteq$-monotone sequence $\{A_\alpha\}_{\alpha<\eta}$, is a quasi-inductive definition; thus, the notion of a quasi-inductive definition generalizes that of an inductive definition. The set $\liminf\{A_\alpha\}_{\alpha<\omega_1}$ is said to be the stable point of the quasi-inductive definition $\{A_\alpha\}_{\alpha<\omega_1}$. An operator $\Gamma\colon \wp(\mathbb{N}) \to \wp(\mathbb{N})$ induces a quasi-inductive definition by iteration and inferior limits as follows

$$\begin{array}{lcl} \Gamma^0 & := & \emptyset \\ \Gamma^{\alpha+1} & := & \Gamma(\Gamma^\alpha) \\ \Gamma^\eta & := & \liminf\{\Gamma^\alpha\}_{\alpha<\eta}, \quad \text{for limit ordinals } \eta. \end{array}$$

In this case we denote by $\Gamma^{<\infty}$ the set $\bigcup_\alpha \bigcap_{\alpha<\beta} \Gamma^\beta$. By a cofinality argument one sees that there is some $\eta < \omega_1$ such that $\Gamma^{<\infty} = \Gamma^\eta$; hence, $\Gamma^{<\infty}$ is the stable point of the quasi-inductive definition $\{\Gamma^\alpha\}_{\alpha<\omega_1}$. Define $\|\Gamma\|$, the stabilization ordinal of Γ, to be the least ordinal η such that $\Gamma^\eta = \Gamma^{<\infty}$. For monotone Γ, the ordinal $\|\Gamma\|$ is often called the closure ordinal of Γ.

The quasi-inductive definition induced by an operator Γ is said to be arithmetical if Γ is an arithmetical operator. A set $A \subseteq \mathbb{N}$ is then said to be arithmetically quasi-inductive if A is m-reducible to the stable point of an arithmetical quasi-inductive definition. The class of arithmetically quasi-inductive sets thus extends the class of arithmetically inductive sets.

Recall the language L of arithmetic introduced above. Let L^+ be the extension of L with the unary predicate Tr. Let L_0^+ denote the set of *first-order* sentences of L^+, that is, the set of L^+-formulas without free variables (of any sort) and with no function quantifiers. For any $A \subseteq \mathbb{N}$, define the L^+-structure $\mathfrak{N} \cup A := (\mathbb{N} \cup 2^{\mathbb{N}}, +, \times, \leq, A, 0, 1)$, the standard structure of arithmetic extended with the set A. Suppose we have a Turing computable coding of L^+, and denote by $\ulcorner \varphi \urcorner$ the code of φ. Define an operator $j \colon \wp(\mathbb{N}) \to \wp(\mathbb{N})$ by

$$j(A) := \{\ulcorner \varphi \urcorner \in \mathbb{N} : \varphi \in L_0^+ \ \& \ \mathfrak{N} \cup A \models \varphi\}.$$

Hence, $j(A)$ is the set of (codes for) first-order L^+-sentences true in the structure $\mathfrak{N} \cup A$. By Tarski's theorem on the undefinability of truth, the operator j is not arithmetical. It follows, however, easily from well-known facts that j is a Δ_1^1 operator.

Call the quasi-inductive definition induced by j for the N-sequence, and let $N_{<\infty}$ be the stable point of the N-sequence.[4] The N-sequence arises in Herzberger's so-called "naive" truth-predicate semantics (see [7]), where the set $N_{<\infty}$ is called the set of stably true sentences of arithmetic. Burgess proved that

Theorem 3.5.3 (Burgess). *The set $N_{<\infty}$ is complete with respect to the class of arithmetically quasi-inductive sets.*

Proof. The proof proceeds by first showing that the quasi-inductive definition induced by j can be simulated modulo m-reducibility by an *arithmetical* quasi-inductive definition; this proves that $N_{<\infty}$ is arithmetically quasi-inductive.

For the other direction Burgess proves that from an arithmetical $\Gamma \colon 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ one can uniformly devise a first-order L^+-formula $\varphi(x)$ such that

$$\Gamma^\alpha(\emptyset) = \{n \in \mathbb{N} : \ulcorner \varphi(\overline{n}) \urcorner \in N_\alpha\},$$

showing that $\Gamma^{<\infty}$ is 1-reducible to $N_{<\infty}$.

The reader is referred to the proof of Proposition 13.1 in Burgess' paper [1] for more details. q.e.d.

For readers familiar with Kripke's inductive truth-predicate semantics (see [12]) we notice the following fact. Let (T^+, T^-) be the least fixed-point for the predicate Tr over $\mathfrak{N}$ with respect to Kleene's strong three-valued logic. Then T^+ is a complete Π_1^1 set, and hence T^+ is complete with respect to the arithmetically inductive sets (for a proof of this fact, see Theorem 6.1 in Burgess'

[4] The nomenclature here comes from Burgess, who also calls the N-sequence *the negative sequence*, the sequence "that never gives a sentence the benefit of the doubt." ([1], p. 673)

paper [1]). Thus, both for the class of arithmetically inductive sets and for the class of arithmetically quasi-inductive sets there is a "truth-set" with respect to which the class is complete; and just as the arithmetically quasi-inductive sets generalize the arithmetically inductive sets so does Herzberger's "naive" truth-predicate semantics generalize Kripke's inductive truth-predicate semantics. Below we will see that similar remarks apply to $\mathcal{O}$ and $\mathcal{O}^{++}$.

The next result is due to Welch,[5] but the proof given here originates with me.

Proposition 3.5.4 (Welch). *The sets $N_{<\infty}$ and s are computably isomorphic.*

Proof. To show that $N_{<\infty} \leq_1 s$ we provide, uniformly, for each $n \in \mathbb{N}$ a program index $\hat{n}$ such that $Q_{\hat{n}}(0)$ stabilizes if and only if $n \in N_{<\infty}$. To that end, notice that as the operator j is Δ^1_1, there is an ITTM program which given A, can write $j(A)$. The only problem, then, with simulating the quasi-inductive definition induced by j, that is, the N-sequence, on an infinite time Turing machine, is the limit rule: quasi-inductive definitions use the lim inf rule, while infinite time Turing machines use the lim sup rule. It may come as no surprise on the reader that this problem has a solution: we will now describe an algorithm the execution of which simulates the N-sequence. We think of the scratch tape as divided into two infinite parts, Scratch 0 and Scratch 1, none of which includes the first cell. On the Scratch 0 tape, we will compute the iterations of j. Let Output 0 denote the part of the Scratch 0 tape at which we will be writing the iterations j^α, and let Cell c be the n-th cell on the Output 0 tape.

($\star$) Suppose j^α is written on the Output 0 tape and the first cell of the scratch tape shows a 0. Then, compute $j^{\alpha+1}$, write the value of Cell c on the first cell of the output tape, and flash the first cell of the scratch tape. Thereafter, transfer the Output 0 tape to the Scratch 1 tape by representing all 1's on the Output 0 tape by a pair 01, and all 0s on the Output 0 tape by a pair 10. Then go back to ($\star$).

Now suppose we are in the limit state and the first cell of the scratch tape shows a 1. This means that the procedure ($\star$) has been repeated some limit ordinal η many steps. Now we transfer the Scratch 1 tape to the Output 0 tape as follows. All pairs 01 are translated to 1, all pairs 10 to 0, and, most importantly, all pairs 11 are translated to 0. This makes sure that $\liminf\{j^\alpha\}_{\alpha<\eta}$ gets written on the Output 0 tape. Write the value of Cell c on the first cell of the real output tape and go to ($\star$).

It is then straightforward to verify that $n \in N_{<\infty}$ if and only if $\hat{n} \in s$, and we omit the details. In fact, the algorithm just described generalizes to show that any set which is the stable point of a quasi-inductive definition induced by an ITTM computable operator 1-reduces to s.

To prove that $s \leq_1 N_{<\infty}$ it suffices by Burgess' Theorem 3.5.3 to show that s is arithmetically quasi-inductive. And for that it suffices to show that there is an infinite time Turing machine $Q^{\inf}_d$ using the lim inf limit rule such that $s = \liminf\{Q^{\inf}_{d,\alpha}(0)\}_{\alpha<\eta}$ for all sufficiently large ordinals η. We now describe such a machine $Q^{\inf}_d$. The machine simulates all computations $\{Q_e(0)\}_{e\in\mathbb{N}}$ *using the* lim sup *limit rule*. This is possible, for instance, by representing, as above, all 0's by the pair 10, and all 1's by the pair 01; then, at any simulated limit stage all pairs 00 are transformed to 01 before the simulation continues. Now,

[5] It follows as an easy corollary of some of his and Burgess' work that we will present below.

at any successor stage $\alpha+1$ of this simulation do the following. For any $e \in \mathbb{N}$, if $Q_{e,\alpha+1}(0) = Q_{e,\alpha}(0)$, then write a 1 in the eth cell of the real output tape; and if $Q_{e,\alpha+1}(0) \neq Q_{e,\alpha}(0)$, then write a 0 in the eth cell of the real output tape. It is then clear that if, and only if, the computation $Q_e(0)$ stabilizes, then the eth cell of the output tape of $Q_d^{\inf}$ will stabilize to 1. This completes our proof. q.e.d.

Recall from Proposition 1.3.2 that classical Kleene's $\mathcal{O}$ is complete with respect to the class of arithmetically inductive sets. We now get that

Corollary 3.5.5. *The set $\mathcal{O}^{++}$ is complete with respect to the class of arithmetically quasi-inductive sets.*

Proof. Immediate from Theorem 3.5.3, Theorem 3.3.17 and Proposition 3.5.4. q.e.d.

Connections with constructibility theory From the viewpoint of the set theorist, the deeper reason why the sets $\mathcal{O}^{++}$, s, and $N_{<\infty}$ are all computably isomorphic is that, firstly, they all have a Σ_2 definition inside $\mathbf{L}_\zeta$, the ζ-th level of Gödel's constructible hierarchy, and secondly, they all code the Σ_2 theory of the structure $(\mathbf{L}_\zeta, \in)$. Let us make this more precise.

Let L_{ST} be the language of set theory, that is the first-order language whose sole non-logical symbol is $\in$. Let $\mathbf{V}$ denote the universe of sets. The definable power set operation $\mathscr{D}$ is defined as follows. For any $A \in \mathbf{V}$, say that $B \in \mathscr{D}(A)$ if and only if there is a formula $\varphi(x)$ of L_{ST} possibly using parameters from A such that $v \in B \Leftrightarrow (A, \in) \models \varphi[v]$. The constructible universe $\mathbf{L}$ is then $\bigcup_\alpha \mathbf{L}_\alpha$, where

$$\begin{array}{lcl} \mathbf{L}_0 & := & \emptyset \\ \mathbf{L}_{\alpha+1} & := & \mathscr{D}(\mathbf{L}_\alpha) \\ \mathbf{L}_\eta & := & \bigcup_{\alpha<\eta} \mathbf{L}_\alpha, \quad \text{for limit ordinals } \eta. \end{array}$$

A formula $\varphi(\vec{x})$ of L_{ST} is Δ_0 if all quantifiers appearing in φ are bounded; the formula φ is Σ_1 (Π_1) if it is of the form $\exists x\, \psi(x, \vec{y})$ ($\forall x\, \psi(x, \vec{y})$) with ψ a Δ_0 formula; finally, the formula $\varphi(\vec{x})$ is Σ_{n+1} (Π_{n+1}) if it is of the form $\exists x\, \psi(x, \vec{y})$ ($\forall x\, \psi(x, \vec{y})$) with ψ a Π_n (Σ_n) formula. We suppose a natural number coding $\ulcorner \cdot \urcorner$ of all formulas of L_{ST}. Define

$$\mathrm{Th}_{\Sigma_n}(A) := \{\ulcorner \varphi \urcorner : \varphi \text{ is a } \Sigma_n \text{ sentence of } L_{\mathrm{ST}} \,\&\, (A, \in) \models \varphi\}.$$

If $A, B \in \mathbf{V}$ we write $A \prec_{\Sigma_n} B$ if for any Σ_n formula $\varphi(\vec{x})$ and any tuple $\vec{v} \in A^n$ we have $A \models \varphi[\vec{v}] \Leftrightarrow B \models \varphi[\vec{v}]$.

For any $v \in \mathbf{V}$, let $\mathrm{trcl}(v)$ denote the transitive closure of v. To set up a connection between set theory and ITTM theory, notice that any set $v \in \mathbf{V}$ whose transitive closure is countable can be coded as a real: let $\{v_n\}_{n\in\mathbb{N}}$ be an enumeration of $\mathrm{trcl}(\{v\})$, and let E be the unique relation on $\mathbb{N}$ such that

$$mEn \Leftrightarrow v_m \in v_n$$

holds for all $m, n \in \mathbb{N}$. If $a \in 2^{\mathbb{N}}$ codes this relation E, then a is said to be a code for v. As $v \neq w$ implies $(\mathrm{trcl}(\{v\}), \in) \ncong (\mathrm{trcl}(\{w\}), \in)$, it follows that if a codes a set, then it codes a unique set. Using the coding of sets $v \in \mathbf{V}$ as reals we can transfer notions from ITTM theory to set theory, and say, for instance, that a set $v \in \mathbf{V}$ is writable if it has a writable code.

This coding of sets as reals was used to great effect already by Hamkins & Lewis in their first paper [5]. There they showed, for instance, that given a code for α, an infinite time Turing machine can write a code for $\mathbf{L}_\alpha$; they also noted that, given reals coding sets, any Δ_0 fact about these coded sets is ITTM decidable. Using these facts and standard application of the Ordinal Production Machine, it is easy to see that $\mathbf{L}_\zeta \prec_{\Sigma_1} \mathbf{L}_\Sigma$.

Welch proved in his [24] that

Proposition 3.5.6 (Welch). *The sets s and $\mathrm{Th}_{\Sigma_2}(\mathbf{L}_\zeta)$ are computably isomorphic.*

Proof. To prove that $s \leq_1 \mathrm{Th}_{\Sigma_2}(\mathbf{L}_\zeta)$, recall the stabilization function δ defined above. Using the Timing Theorem 3.1.7 it is clear that $e \in s$ if and only if

$$\forall k \in \mathbb{N}\, \exists \alpha < \zeta\, \forall \beta < \zeta (\alpha < \beta \to \delta^e_k(\beta) = \alpha).$$

Thus, by absoluteness considerations, the function which sends $n \in \mathbb{N}$ to an L_{ST} formula expressing $\forall k \in \omega\, \exists \alpha \forall \beta > \alpha\, (\delta^e_k(\beta) = \alpha)$ witnesses that $s \leq_1 \mathrm{Th}_{\Sigma_2}(\mathbf{L}_\zeta)$.

To show the other direction, namely that $\mathrm{Th}_{\Sigma_2}(\mathbf{L}_\zeta) \leq_1 s$, we will for any formula φ of L_{ST} describe a program $Q_{\hat{\varphi}}$ which stabilizes on input 0 if and only if $\ulcorner \varphi \urcorner \in \mathrm{Th}_{\Sigma_2}(\mathbf{L}_\zeta)$. For any given φ, the program $Q_{\hat{\varphi}}$ works as follows.

If φ is not a Σ_2 sentence, then we let $Q_{\hat{\varphi}}$ be a program which (on any input) keeps on writing and erasing $\ulcorner \varphi \urcorner$ on the output tape to eternity.

If φ is a Σ_2 sentence, and thus of the form $\exists x \forall y\, \psi(x, y)$ for $\psi(x, y)$ a Δ_0 formula, then the program $Q_{\hat{\varphi}}$ works as follows on input 0. Do a double simulation of the Ordinal Production Machine; let us call these simulations OPM1 and OPM2. ($\star$) Given an ordinal α from OPM1, then write $\mathbf{L}_\alpha$ and ($\star\star$) wait for an ordinal β from OPM2. Then, given such a β, check whether $\alpha < \beta$. If the check is negative, then just go back to ($\star\star$); if the check is positive, then do the following. First, write $\mathbf{L}_\beta$; thereafter check whether

$$(i) \quad \exists v \in \mathbf{L}_\alpha \forall w \in \mathbf{L}_\beta\, \psi[v, w].$$

holds. If (i) holds, then erase β and $\mathbf{L}_\beta$ and go to ($\star\star$). If, on the other hand, (i) does not hold, then flash the first cell of the output tape, erase all of α, β, $\mathbf{L}_\alpha$, and $\mathbf{L}_\beta$ and go to ($\star$). This completes the description of the program $Q_{\hat{\varphi}}$.

Now suppose that $\ulcorner \varphi \urcorner \in \mathrm{Th}_{\Sigma_2}(\mathbf{L}_\zeta)$ and that φ is of the form $\exists x \forall y\, \psi(x, y)$ for a Δ_0 formula $\psi(x, y)$. Then there is an $\alpha < \zeta$ such that $\exists v \in \mathbf{L}_\alpha\, \forall w \in \mathbf{L}_\zeta\, \psi[v, w]$. As $\mathbf{L}_\zeta \prec_{\Sigma_1} \mathbf{L}_\Sigma$ we get $\exists v \in \mathbf{L}_\alpha\, \forall w \in \mathbf{L}_\Sigma\, \psi[w, v]$. It is thus clear that $Q_{\hat{\varphi}}(0)$ stabilizes in this case.

Finally suppose that $Q_{\hat{\varphi}}(0)$ stabilizes. Then φ is of the form $\exists x \forall y\, \psi(x, y)$ for a Δ_0 formula ψ, and there must be some $\alpha < \Sigma$ such that (i) holds for all β with $\alpha < \beta < \Sigma$. Using by now standard methods it is easy to see that this α is, in fact, eventually writable; hence $\ulcorner \varphi \urcorner \in \mathrm{Th}_{\Sigma_2}(\mathbf{L}_\zeta)$. q.e.d.

For any set $A \in \mathbf{V}$ say that $B \in \mathbf{V}$ is $\Sigma_n(A)$ if there is a Σ_n formula φ of L_{ST} such that

$$v \in B \Leftrightarrow A \models \varphi[v].$$

Notice that φ is not allowed to have parameters from A.

It is a well-known fact that

Proposition 3.5.7 (Folklore). *The arithmetically inductive sets are precisely the sets in $\Sigma_1(\mathbf{L}_{\omega_1^{\mathrm{CK}}})$.*

Proof. We prove that $\Pi^1_1 = \Sigma_1(\mathbf{L}_{\omega_1^{\mathrm{CK}}})$. As already noted, the set W of indices for computable well-orders is Π^1_1-complete; hence, to prove $\Pi^1_1 \subseteq \Sigma_1(\mathbf{L}_{\omega_1^{\mathrm{CK}}})$ it suffices to show that W is $\Sigma_1(\mathbf{L}_{\omega_1^{\mathrm{CK}}})$. Recall that R_e denotes the binary relation coded by $\{e\}$. Then notice that $e \in W$ if and only if there is an $\alpha < \omega_1^{\mathrm{CK}}$ and an embedding $F\colon R_e \hookrightarrow \alpha$. Now, if such an emdedding F exists, then there certainly exists one in $\mathbf{L}_{\omega_1^{\mathrm{CK}}}$, thus

$$e \in W \Leftrightarrow \mathbf{L}_{\omega_1^{\mathrm{CK}}} \models \exists\alpha\exists F\,(F\colon R_e \hookrightarrow \alpha).$$

For the other direction, suppose that $A \subseteq \mathbb{N}$ is $\Sigma_1(\mathbf{L}_{\omega_1^{\mathrm{CK}}})$. Then we have $n \in A$ iff

$$(i) \quad \mathbf{L}_{\omega_1^{\mathrm{CK}}} \models \exists\alpha\exists v \in \mathbf{L}_\alpha\, \psi[v, n].$$

But (i) is a Π^1_1 statement as it is equivalent to the statement that there is an $e \in W$ with $\{e\}$ coding α, say, and such that for all $B \subseteq \mathbb{N}$ which looks like $\mathbf{L}_\alpha$ (this statement is Π^1_1 in B and α) we have "$B \models \exists x\psi(x, n)$" (this statement is arithmetical in B and n). q.e.d.

Closely related to this proposition is the fact that

Proposition 3.5.8 (Folklore). *We have*

$$\sup\{\|\Gamma\| : \Gamma \textit{ is } \subseteq\textit{-monotone and arithmetical}\} = \omega_1^{\mathrm{CK}}.$$

Proof. It is not hard to see that if Γ is arithmetical, then Γ^α is hyperarithmetical for all $\alpha < \omega_1^{\mathrm{CK}}$. Thus, as $\mathcal{O}$ is the least fixed-point of an arithmetical monotone operator and is a complete Π^1_1 set, it follows that $\sup\{\|\Gamma\| : \Gamma \text{ is } \subseteq\text{-monotone and arithmetical}\} \leq \omega_1^{\mathrm{CK}}$. On the other hand, one can show that $\Gamma^{\omega_1^{\mathrm{CK}}}$ is not hyperarithmetical; using this fact and Aczel's Stage Comparison Theorem (see [15]) the proposition follows. q.e.d.

Say that an ordinal α is Σ_n extendible if there is an ordinal $\beta > \alpha$ such that $\mathbf{L}_\alpha \prec_{\Sigma_n} \mathbf{L}_\beta$. Burgess generalized Proposition 3.5.7 to the case of quasi-inductive definitions by proving that

Theorem 3.5.9 (Burgess). *The arithmetically quasi-inductive sets are precisely the sets in $\Sigma_2(\mathbf{L}_\eta)$, where η is the least Σ_2 extendible ordinal.*

Proof. The proof is rather long-winded, so we refer the reader to Theorem 14.1 in Burgess' paper [1]. q.e.d.

Corollary 3.5.10. *Let η be the least Σ_2 extendible ordinal. Then, the set $\mathrm{Th}_{\Sigma_2}(\mathbf{L}_\eta)$ is complete with respect to the class of arithmetically quasi-inductive sets.*

Proof. By Theorem 3.5.9, any $A \subseteq \mathbb{N}$ which is arithmetically quasi-inductive has a Σ_2 definition $\varphi_A(x)$ over $\mathbf{L}_\eta$. Thus the function $n \mapsto \ulcorner\varphi_A(\overline{n})\urcorner$ witnesses that $A \leq_1 \mathrm{Th}_{\Sigma_2}(\mathbf{L}_\eta)$. q.e.d.

The parallel of Proposition 3.5.8 also obtains.

Proposition 3.5.11 (Burgess). *We have*

$$\sup\{\|\Gamma\| : \Gamma \textit{ is arithmetical}\} = \eta,$$

where η is the least Σ_2 extendible ordinal.

Proof. Let η be the least Σ_2 extendible ordinal. The first part of Burgess' proof of his Theorem 14.1 in [1] proves that any arithmetical quasi-inductive definition starts looping no later than η. On the other hand, if the N-sequence started looping before η, then we would get $N_{<\infty} \in \mathbf{L}_\eta$ which would contradict the completeness of $N_{<\infty}$ with respect to $\Sigma_2(\mathbf{L}_\eta)$. q.e.d.

Welch connected Burgess' work on quasi-inductive definitions with ITTM computability[6] by proving that

Theorem 3.5.12 (Welch). *The supremum of the accidentally writable orindals, ζ, is the least Σ_2 extendible ordinal. Furthermore, the supremum of the accidentally writable ordinals, Σ, is the least ordinal α greater than ζ such that $\mathbf{L}_\zeta \prec_{\Sigma_2} \mathbf{L}_\alpha$*

Proof. We first prove that $\mathbf{L}_\zeta \prec_{\Sigma_2} \mathbf{L}_\Sigma$; this implies that ζ is Σ_2 extendible. Let $\varphi(v)$ be $\exists x \forall y\, \psi(v, x, y)$, where ψ is Δ_0 and $v \in \mathbf{L}_\zeta$. We notice that $v \in \mathbf{L}_\zeta$ implies $v \in \mathbf{L}_\alpha$ for some $\alpha < \zeta$, so v is eventually writable.

Suppose first that $\mathbf{L}_\zeta \models \varphi$. Then there is a $w \in \mathbf{L}_\zeta$ such that $\mathbf{L}_\zeta \models \forall x\, \psi(v, w, x)$. As $\mathbf{L}_\zeta \prec_{\Sigma_1} \mathbf{L}_\Sigma$, we get $\mathbf{L}_\Sigma \models \varphi$.

Now suppose that $\mathbf{L}_\Sigma \models \varphi$. Recall the algorithm $Q_{\hat{\varphi}}$ described in the proof of Proposition 3.5.6. We slightly modify that by telling the machine to write the ordinal α given from the simulation OPM1 on the output tape (if it is not already written there), and compute approximations v^* to v such that we check

$$(i^*) \quad \exists u \in \mathbf{L}_\alpha \forall w \in \mathbf{L}_\beta\, \psi[v^*, u, w].$$

instead of the statement (i). It is then straightforward to see that a machine executing this modified algorithm will eventually write the least ordinal α such that for all β with $\alpha < \beta < \Sigma$ we have $\exists u \in \mathbf{L}_\alpha\, \forall w \in \mathbf{L}_\beta\, \psi[v, u, w]$; in consequence, $\exists u \in \mathbf{L}_\zeta\, \forall w \in \mathbf{L}_\zeta\, \psi[v, u, w]$, that is $\mathbf{L}_\zeta \models \varphi$.

To show that ζ and Σ are least with the required properties, suppose that γ and η are such that $\mathbf{L}_\gamma \prec_{\Sigma_2} \mathbf{L}_\eta$. Let $\varphi_{e,k}$ be the formula $\exists \alpha \forall \beta > \alpha\, (\delta^e_k(\beta) = \alpha)$, where δ is the stabilization function defined above. Then

$$\mathbf{L}_\gamma \models \varphi_{e,k} \Leftrightarrow \mathbf{L}_\eta \models \varphi_{e,k}$$

holds for every $e, k \in \mathbb{N}$. It follows from this, as in Corollary 3.1.6, that any ITTM computation will be in a loop at stage γ, and that the γ-th snapshot of the computation will be the same as the η-th snapshot. As there are ITTM computations which do not start looping before stage ζ and which do not repeat again before stage Σ, it follows that $\zeta \leq \gamma$ and $\Sigma \leq \eta$. q.e.d.

Thus, a set $A \subseteq \mathbb{N}$ is arithmetically quasi-inductive if and only if it has a Σ_2 definition over $\mathbf{L}_\zeta$.

To conclude this section, we note that Welch in his [25] proved that the set of "ultimate truths" of arithmetic in Field's truth-predicate semantics (see [4]) is computably isomorphic to $\mathrm{Th}_{\Sigma_2}(\mathbf{L}_\zeta)$, thus putting yet another naturally arising (and conceptually interesting) set in the same isomorphism class as $\mathcal{O}^{++}$.

[6] Löwe was, with his [13], perhaps the first to notice that there is a relation between quasi-inductive definitions (or "revision sequences", which is a generalized form of quasi-inductive definitions) and ITTM computability.

3.6 Hyperinductive Theory

Hyperarithmetic theory, developed by Kleene, Spector, Kreisel, and others (see Part A of Sacks' book [19] for an overview), knows certain basic objects and relations. The basic objects are the class of hyperarithmetic sets, the class of Π^1_1 sets, the ordinal ω_1^{CK} and Kleene's $\mathcal{O}$, Turing reducibility and the Turing jump, and, finally, hyperreducibility ($\leq_{\mathrm{HYP}}$) and the hyperjump. As we saw in the previous section, behind this lies the class $\Sigma_1(\mathbf{L}_{\omega_1^{\mathrm{CK}}})$ of sets Σ_1 definable (without parameters) over $\mathbf{L}_{\omega_1^{\mathrm{CK}}}$.

In this concluding section we want to argue that there should be a parallel of hyperarithmetic theory at the level of $\Sigma_2(\mathbf{L}_\zeta)$. For now we want to call such a theory (if such there be) *hyperinductive* theory, as it concerns certain classes of sets lying "slightly" above the arithmetically inductive sets. We want to point out that whereas classical hyperarithmetic theory seems to be closely tied up with the notion of monotonic processes (*i.e.*, inductive definitions), its parallel hyperinductive theory should be seen to be closely tied to the notion of non-monotonic processes. In the following we are going to propose for each of the basic objects and relations of classical hyperarithmetic theory its hyperinductive counterpart. Given a statement of hyperarithmetic theory we should then be able to provide a twin statement of hyperinductive theory by replacing all mention of the basic hyperarithmetic objects with their hyperinductive counterparts. We would, rather boldly, like to think of this section as encouraging the research program of finding out how the theorems of classical hyperarithmetic theory transfer to hyperinductive theory.

We commence by admitting that $\mathcal{O}^{++}$ is going to play the role of $\mathcal{O}$, and ζ the role of ω_1^{CK}. In many results of hyperarithmetic theory, it is not the full $\mathcal{O}$ which is used, but rather a Π^1_1 path going through $\mathcal{O}$ – this is because one is interested in having for each ordinal $\alpha < \omega_1^{\mathrm{CK}}$ a unique notation in $\mathbb{N}$. We have seen in Proposition 3.3.18 above that there is a system $\mathscr{Z}$ of unique notations for ordinals $\alpha < \zeta$, and which is computably isomorphic to $\mathcal{O}^{++}$; hence, when a set of unique notations is wanted, we suggest that $\mathscr{Z}$ be used.

In the previous section, we saw that whereas $\mathcal{O}$ is complete with respect to the class of Π^1_1 sets of integers (or, equivalently, the arithmetically inductive sets), $\mathcal{O}^{++}$ is complete with respect to the class of arithmetically quasi-inductive sets. Moreover, we saw that Π^1_1 equals $\Sigma_1(\mathbf{L}_{\omega_1^{\mathrm{CK}}}) \cap \wp(\mathbb{N})$ and that arithmetically quasi-inductive equals $\Sigma_2(\mathbf{L}_\zeta) \cap \wp(\mathbb{N})$. There is thus good support for letting the arithmetically quasi-inductive sets play the role of the Π^1_1 sets.

Let us note the following apparent discrepancy. In addition to being complete for the class of arithmetically inductive sets, Kleene's $\mathcal{O}$ is itself the fixed-point of an arithmetical inductive definition. We know that $\mathcal{O}^{++}$ is m-reducible to the stable point of an arithmetical quasi-inductive definition, but we have not proved that $\mathcal{O}^{++}$ is in fact the stable point of such a definition. Certainly, in our Definition 3.2.1 of $\mathcal{O}^{++}$, we obtain $\mathcal{O}^{++}$ as the stable point of a quasi-inductive definition; but, as noted, that inductive definition is not arithmetical; indeed, it is easy to see that there can be no arithmetical inductive definition with $\mathcal{O}^{++}$ as a fixed-point. These considerations give rise to the following

Question 1. Can the set $\mathcal{O}^{++}$ be obtained as the stable point of an arithmetically quasi-inductive definition?

More generally, Löwe has asked whether all arithmetically quasi-inductive

sets are the stable point of some arithmetical quasi-inductive definition. By a result of Hinman (see his [8, p. 130]), the parallel of this does not hold for the class of arithmetically inductive sets; that is, there are arithmetically inductive sets (in particular, Δ^1_1 Cohen generic reals) which are not the fixed-point of any arithmetical inductive definition.

Let us move on to the question of what is going to play the role of the hyperarithmetic sets in hyperinductive theory. We proceed in a *brute force* manner and imitate within the infinite time setting the classical definition of the hyperarithmetic sets and see what we get. Define the E-sets by recursion on $\mathcal{O}^{++}$ as follows.

$$\begin{array}{lcll} E_1 & := & \emptyset & \\ E_{2^n} & := & E_n^{\nabla} & \text{for } n \in \text{field}(\mathcal{O}^{++}) \\ E_{3\cdot 5^e} & := & \{\langle x, Q_e(n)\rangle : x \in J_{Q_e(n)}\} & \text{for } 3\cdot 5^e \in \text{field}(\mathcal{O}^{++}) \end{array}$$

Temporarily, say that a set $A \subseteq \mathbb{N}$ is infinite time hyperarithmetic if and only if there is an $n \in \text{field}(\mathcal{O}^{++})$ such that $A \leq_\infty E_n$.

Observation 3.6.1. *A set $A \subseteq \mathbb{N}$ is infinite time hyperarithmetic if and only if it is eventually writable.*

Proof. For the *only if* direction it is enough to prove that any E-set is eventually writable. This is proved by induction on $<_{\mathcal{O}^{++}}$. The successor case is easy, as the weak jump of any eventually writable real is again eventually writable. For the limit case, recall that, by Proposition 3.2.2, $\mathcal{O}^{++}\restriction 3\cdot 5^e$ is an eventually writable code for an ordinal α. Thus, to eventually write $E_{3\cdot 5^e}$ we can follow the priority algorithm in the proof of Proposition 3.3.3 eventually writing 0^{∇^α} on some part of the scratch tape, but in addition use approximations to $\{\langle Q_e(k), Q_e(k+1)\rangle\}_{k\in\mathbb{N}}$ to single out the slices of 0^{∇^α} that we write on the output tape, giving lowest priority to the computations $\{Q_e(k)\}_{k\in\mathbb{N}}$.

For the *if* direction, note first that $m <_{\mathcal{O}^{++}} n$ implies $E_m <_\infty E_n$. Moreover, there are paths $b \subseteq \mathcal{O}^{++}$ of height ζ. Thus $\{E_n : n \in \text{field}(b)\}$ under $\leq_\infty$ forms a well-order of height ζ. Thus, if there were some eventually writable A not reducible to any E-set, we would get $E_n <_\infty A$ for all $n \in \text{field}(b)$, so IE would have height at least $\zeta + 1$, contradicting Proposition 3.3.4. q.e.d.

The contention that the eventually writable reals should play the role of the hyperarithmetic sets thus has something going for it. In fact, it has a lot going for it:

Proposition 3.6.2 (Folklore). *Let $A \subseteq \mathbb{N}$.*

(i) We have $A \in \mathbf{L}_{\omega_1^{\text{CK}}}$ if and only if A is hyperarithmetic.

(ii) We have $A \in \mathbf{L}_\zeta$ if and only if A is eventually writable.

Proof. For the *only if* direction of *(i)*, notice that if $n \in \text{field}(\mathcal{O})$ and $|n|_{\mathcal{O}} = \alpha$, then $H_n \in \mathbf{L}_\alpha$; it follows that any hyperarithmetic set is a member of $\mathbf{L}_{\omega_1^{\text{CK}}}$. The *if* direction of *(i)* follows from the fact that any set in $\mathbf{L}_{\omega_1^{\text{CK}}}$ is coded by a hyperarithmetic real (see chapter VII.1 in Sacks' book [19] for details).

For part *(ii)* notice that if $A \in \mathbf{L}_\zeta$, then $A \in \mathbf{L}_\alpha$ for some eventually writable α, so by results of Hamkins & Lewis ([5]) it follows that A is eventually writable. On the other hand, if A is eventually writable then there are $e \in \mathbb{N}$ and $\alpha < \zeta$ such that $Q_{e,\alpha}(0) = A$. But we can run the computation $Q_e(0)$ inside $\mathbf{L}_\zeta$, so we will get $A \in \mathbf{L}_\zeta$. q.e.d.

In the jargon of α-recursion theory (see Part C of Sacks' book [19]), this proposition says that just as the hyperarithmetic sets are exactly the ω_1^{CK}-finite sets, so are the eventually writable reals exactly the ζ-finite sets.

With Observation 3.6.1 and Proposition 3.6.2 on our hands we let the eventually writable reals play the role of the hyperarithmetic reals. We will, of course, continue calling the eventually writable reals *the eventually writable reals* and not the hyperinductive reals.

From the way we have put things up now, the rest follows automatically. Given how the eventually writable reals can be obtained by iterating the weak jump $\cdot^{\triangledown}$ along $\mathcal{O}^{++}$ and then closing off under the $\leq_\infty$ reducibility, we will let standard ITTM reducibility, $\leq_\infty$, play the role of Turing reducibility and we will let the weak jump $\cdot^{\triangledown}$ play the role of the Turing jump. Towards giving the parallel of $\leq_{\mathrm{HYP}}$, consider the relation "A is eventually computable from B", denoted by $A \leq_{\mathrm{EV}} B$: we say that A is eventually computable from B if there is an infinite time Turing machine with oracle B which eventually computes χ_A. This definition makes perfect sense for $A, B \subseteq 2^{\mathbb{N}}$, but as elsewhere in this thesis, we are interested in sets $A, B \subseteq \mathbb{N}$. Further, we let $s^A : \{e \in \mathbb{N} : Q_e^A(0)\uparrow\}$. Naturally then, we decide that the relation $\leq_{\mathrm{EV}}$ will play the role of $\leq_{\mathrm{HYP}}$, whereas the function $A \mapsto s^A$ will play the role of the hyperjump. We will call s^A the *zeta*-jump of A and denote it by A°. Note that A° and $(\mathcal{O}^{++})^A$ are computably isomorphic – the zeta-jump of A is thus the exact parallel of the hyperjump given the way we have set things up. We prefer, however, our formulation using s, as that makes the zeta-jump more concrete, that is, more like the standard Turing jump involving a halting problem. We should note that Welch in his [24] has studied both the relation $\leq_{\mathrm{EV}}$ (it is there denoted by $\leq_{e\infty}$) and the zeta-jump. Let us now sum up our decisions in the following diagram.

$\Sigma_1(\mathbf{L}_{\omega_1^{\mathrm{CK}}})$	Π^1_1	Hyper-arithmetic	$\mathcal{O}$	ω_1^{CK}	$\leq_{\mathrm{T}}$	$\leq_{\mathrm{HYP}}$
$\Sigma_2(\mathbf{L}_\zeta)$	Arithmetically Quasi-Inductive	Eventually Writable	$\mathcal{O}^{++}$	ζ	$\leq_\infty$	$\leq_{\mathrm{EV}}$

Figure 3.1: The Characters of Hyperinductive Theory

It is our belief then that

Conjecture 2. Many theorems from classical hyperarithmetic theory carry over to hyperinductive theory.

As a very small beginning we observe that Spector's Theorem 3.4.4 easily carries over.

Observation 3.6.3. *Let $A, B \subseteq \mathbb{N}$. If $A \leq_{\mathrm{EV}} B$, then $\zeta^A \leq \zeta^B$.*

Proof. If α is eventually writable from A and A is eventually writable from B, then it is seen using by now standard methods that α is eventually writable from B. q.e.d.

Notice, by the way, that Corollary 3.4.5 saying that the supremum of the hyperarithmetic ordinals equals ω_1^{CK} carries over trivially: the ordinal ζ is by definition the supremum of the eventually writable ordinals.

Notice also how Observation 3.6.3 implies

Observation 3.6.4. *The zeta-jump is well-defined on the $\equiv_{\mathrm{EV}}$-degrees; that is, if $A \equiv_{\mathrm{EV}} B$, then $s^A \equiv_{\mathrm{EV}} s^B$.*

Proof. It suffices to show that $A \leq_{\mathrm{EV}} B$ implies $s^A \leq_{\mathrm{EV}} s^B$, so suppose the former. Then, notice that since $\zeta^A \leq \zeta^B$ holds, and since ζ^B is s^B-writable, it follows that ζ^A is s^B-writable; but then, s^A is s^B-writable as well; thus, we get $s^A \leq_{\mathrm{EV}} s^B$. q.e.d.

Possible directions for further research. Having thus peopled hyperinductive theory, we want to conclude this thesis by raising some questions which seem natural given the parallels we have drawn up. These questions can also be seen as giving content to Conjecture 2 above.

Question 3 (Spector boundedness). Spector proved that if $A \subseteq \mathbb{N}$ is Σ^1_1 and $A \subseteq \mathcal{O}$, then there is an $n \in \mathrm{field}(\mathcal{O})$ such that $|k|_{\mathcal{O}} < |n|_{\mathcal{O}}$ for all $k \in A$. Does the same hold for co-arithmetically-quasi-inductive sets and $\mathcal{O}^{++}$; that is, if $A \subseteq \mathcal{O}^{++}$ and A is the complement of an arithmetically quasi-inductive set, is there an $n \in \mathrm{field}(\mathcal{O}^{++})$ such that $|k|_{\mathcal{O}^{++}} < |n|_{\mathcal{O}^{++}}$ for all $k \in A$?

Question 4 (Pre-well-ordering property). For a class of sets $\mathcal{A}$ with $A \in \mathcal{A}$, say that $\rho\colon A \to \mathrm{ON}$ is an $\mathcal{A}$-norm on A if the relations $\leq^*_\rho$ and $<^*_\rho$, defined by

$$\begin{array}{lcl} v \leq^*_\rho w & \Leftrightarrow & v \in A \;\&\; (w \in A \to \rho(v) \leq \rho(w)) \\ v <^*_\rho w & \Leftrightarrow & v \in A \;\&\; (w \in A \to \rho(v) < \rho(w)) \end{array}$$

both are in $\mathcal{A}$. It is well-known that the class of Π^1_1 sets of integers has the pre-well-ordering property – does the class of arithmetically quasi-inductive sets have the the pre-well-ordering property?

Question 5 (Quasi-induction on acceptable structures). Say that an inductive definition $S = \{A_\alpha\}_{\alpha < \omega_1^{\mathrm{CK}}}$ is elementary over a structure $\mathfrak{M}$ if S is generated by an operator Γ which is first-order definable over $\mathfrak{M}$. Moschovakis in his [15] developed a rich theory of elementary inductive definitions on acceptable structures. Is there a similarly rich theory of elementary quasi-inductive definitions on acceptable structures?

Before presenting the last question, let us introduce some notation and concepts. Let L^{RA} be the language L of arithmetic, referred to above, extended with *ranked* function-variables of the form f_α, for $\alpha < \omega_1$. An L^{RA} formula φ is said to be ranked if all function variables occurring in φ are of the form f_α for some $\alpha < \omega_1$. Say that φ is an L^{RA}_β formula if φ is ranked and all function variables occurring in φ have rank less than β. We define simultaneously by recursion on the ordinals a sequence of sets of reals $\{\mathcal{M}_\alpha\}_{\alpha<\omega_1}$ and the relation

$$(\mathfrak{N}, \bigcup_{\alpha<\beta} \mathcal{M}_\alpha) \models \varphi,$$

for φ an L^{RA}_β sentence, as follows. Let φ be an L^{RA}_β sentence. The relation $(\mathfrak{N}, \bigcup_{\alpha<\beta} \mathcal{M}_\alpha) \models \varphi$ is defined by recursion on the complexity of φ; the only

non-standard clause is where φ has the form $\exists \mathsf{f}_\eta \psi(\mathsf{f}_\eta)$ for $\eta < \beta$; in that case we say

$$(\mathfrak{N}, \bigcup_{\alpha<\beta} \mathcal{M}_\alpha) \models \exists \mathsf{f}_\eta \psi(\mathsf{f}_\eta) \Leftrightarrow \text{ there is an } a \in \mathcal{M}_\eta \text{ such that } (\mathfrak{N}, \bigcup_{\alpha<\beta} \mathcal{M}_\alpha) \models \psi[a].$$

The set $\mathcal{M}_\beta$ is defined as follows. Say that $a \in 2^{\mathbb{N}}$ is L^{RA}_β definable over $(\mathfrak{N}, \bigcup_{\alpha<\beta} \mathcal{M}_\alpha)$ if there is an L^{RA}_β formula $\varphi(x)$ such that

$$n \in a \Leftrightarrow (\mathfrak{N}, \bigcup_{\alpha<\beta} \mathcal{M}_\alpha) \models \varphi(\overline{n}).$$

The set $\mathcal{M}_\beta$ is then the set of reals which are L^{RA}_β definable over $(\mathfrak{N}, \bigcup_{\alpha<\beta} \mathcal{M}_\alpha)$. The sequence of sets $\{\mathcal{M}_\alpha\}_{\alpha<\omega_1}$ is called the ramified analytic hierarchy.

Question 6 (Ramified analytic hierarchy). Kleene proved that the set $\mathcal{M}_{\omega_1^{\mathrm{CK}}}$ equals the set of hyperarithmetic reals. It is, however, well-known that $\mathcal{M}_{\omega_1^{\mathrm{CK}}} \subsetneq \mathcal{M}_{\omega_1^{\mathrm{CK}}+1}$, so the ramified analytic hierarchy $\{\mathcal{M}_\alpha\}_{\alpha<\omega_1}$ does not stop growing at ω_1^{CK}. Does it continue growing all the way up to ζ, and is $\mathcal{M}_\zeta$ the set of eventually writable reals?

Can one, using the set $\mathscr{Z}$ of unique notations for ζ (see Proposition 3.3.18 above), extend Cohen forcing in the sense of Feferman,[7] or extend Sacks forcing[8] to the language L^{RA}_ζ? What can be said about the generic reals for either notion of forcing?

[7] See Feferman's [3] or chapter IV.3 of Sacks' book [19].

[8] See either the original paper [18] or the textbook presentation in chapter IV.4 of [19].

Bibliography

[1] Burgess, J. P.: The Truth is Never Simple, *Journal of Symbolic Logic* **51** (1986), pp. 663-681.

[2] Feferman, S., Spector, C.: Incompleteness Along Paths in Progression of Theories, *Journal of Symbolic Logic* **27** (1962), pp. 383-390.

[3] Feferman, S.: Some Applications of the Notion of Forcing and Generic Sets, *Fundamenta Mathematicae* **56** (1965), pp. 325-345.

[4] Field, H.: A Revenge Immune Solution to the Semantic Paradoxes, *Journal of Philosophical Logic* **32** (2003), pp. 139-177.

[5] Hamkins, J., Lewis, A.: Infinite Time Turing Machines, *Journal of Symbolic Logic* **65** (2000), pp. 567-604.

[6] Hamkins, J., Lewis, A.: Post's Problem for Supertasks has both Positive and Negative Solutions, *Archive for Mathematical Logic* **42** (2002), pp. 507-523.

[7] Herzberger, H.: Notes on Naive Semantics, *Journal of Philosophical Logic* **11** (1982), pp. 61-102.

[8] Hinman, P. G.: *Recursion-Theoretic Hierarchies*, Springer Verlag, Heidelberg, 1978.

[9] Kleene, S. C.: On Notation for Ordinal Numbers, *Journal of Symbolic Logic* **3** (1938), pp. 150-155.

[10] Kleene, S. C.: Predicates in the Theory of Constructive Ordinals, *American Journal of Mathematics* **77** (1955), pp. 405-428.

[11] Kleene, S. C.: Arithmetical Predicates and Function Quantifiers, *Transactions of the American Mathematical Society* **79** (1955), pp. 312-340.

[12] Kripke, S.: Outline of a Theory of Truth, *The Journal of Philosophy* **72** (1975), pp. 690-716.

[13] Löwe, B.: Revision Sequences and Computers with an Infinite Amount of Time, *Journal of Logic and Computation* **11** (2001), pp. 25-40.

[14] Markwald, W.: Zur Theorie der Konstruktiven Wohlordnungen, *Mathematische Annalen* **127**, pp. 135-149.

[15] Moschovakis, Y.: *Elementary Induction on Abstract Structures*, North-Holland, Amsterdam, 1974.

[16] Myhill, J.: Creative Sets, *Zeitschrift für Matematische Logik und Grundlagen der Mathematik* **1** (1955), pp. 97-108.

[17] Rogers, H.: *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, New York, 1967.

[18] Sacks, P. G.: Forcing With Perfect Closed Sets, *Axiomatic Set Theory. Proceedings of Symposia in Pure Mathematics. American Mathematical Society* **13** (1971), pp. 331-335.

[19] Sacks, P. G.: *Higher Recursion Theory*, Springer Verlag, Heidelberg, 1990.

[20] Shoenfield, J. R.: On Degrees of Unsolvability, *The Annals of Mathematics* **69** (1959), pp. 644-653.

[21] Soare, R. I.: *Recursively Enumerable Sets and Degrees*, Springer Verlag, Heidelberg 1987.

[22] Spector, C.: Recursive Well-Orderings, *Journal of Symbolic Logic* **20** (1955), pp. 151-163.

[23] Welch, P. D.: The Length of Infinite Time Turing Machine Computations, *Bulletin of the London Mathematical Society* **32** (2000), pp. 129-136.

[24] Welch, P. D.: Eventually Infinite Time Turing Machine Degrees: Infinite Time Decidable Reals, *Journal of Symbolic Logic* **67** (2002), pp. 1141-1152.

[25] Welch, P. D.: Ultimate Truth *vis à vis* Stable Truth, *Journal of Philosophical Logic*, to appear.