Characteristics of discrete transfinite time Turing machine models: halting times, stabilization times, and Normal Form theorems

P.D.Welch,
School of Mathematics,
University of Bristol,
England

June 2, 2008

1 Introduction

We give an account of the basic determinants of the courses of computation of an *Infinite Time Turing machine* (ITTM), a model of computation which allows for transfinitely many steps of computation. One such basic theorem relates the ordinal codes capable of being output on halting computations to those capable of being produced eventually by some non-terminating computation. This so-caled "λ-ζ-Σ"-theorem is reproven here, and it is newly related to a theorem of Friedman & Harrington on the levels of the Gödel Constructible hierarchy of sets.

In a second part we provide further new results: (i) a Normal Form Theorem (corresponding to Kleene's Normal Form Theorem for standard Turing machines) in which a code for a course of computation can be produced, uniformly on all inputs, in the lengths of time close to that of the original computation; (ii) a characterization of which ordinals start gaps in the class of ordinal halting times; (iii) an example of divergence between halting points of one-tape and three-tape ITTM architecture answering a question of Hamkins and Seabold [12].

Prerequisites

The reader will be assumed to be familar with the paper of Hamkins and Lewis [10], for the basic notions (although we have outlined most of the basic notions above). They will need some knowledge of ordinal numbers for which see, *e.g.* [6], and some of the Gödel hierarchy of constructible sets L for which [5], or the early parts of [4] can be consulted. For notions associated with standard Turing Machine theory, such as *recursive ordinal*, *Turing degree*, *etc.* then see [27] or [25]. For the theory of admissible sets and ordinals, *vide* [1].

Structure of the paper

The next subsection provides the general setting for the *Infinite Time Turing Machines* (ITTM's) concept. This is a general introduction serving as a less specialised motivation for the specialised results that follow. Section 1.2 relates the ITTM context to that of other work: to Kleene's work on an equational calculus for computation in higher types, and to higher type recursion theory generally. We mention that the degree theory on the integers that emerges from ITTM theory is more similar to that of hyperdegrees or Δ^1_2-degrees than Turing degrees. We mention other connections to *circular definitions* coming from the Gupta-Belnap *revision theory*. We finally indicate some points of connection with ordinal analysis in proof theory.

In Section 1.3 we summarise the contents of Section 2 and the new results of this paper. The reader familiar with ITTM-lehre may wish to go straight there to learn the precise statements of what is appearing here.

Acknowledgements: We should like to thank various people who have contributed questions, queries and discussion of the notions involved and struggled with the incomprehensibility of some of our previous proofs. We mention here: Sy Friedman (in particular for some comments and arguments on the points raised here), Joel Hamkins, Benedikt Löwe, Robert Lubarsky (again in particular for working through a draft of this paper), Joost Winter, *inter alia* and should like to take this opportunity of thanking them.

1.1 The General Context

Infinite Time Turing Machines (ITTM's) are a model of discrete computation based on the standard Turing machine, but where the "time" or stages of computation are allowed to transcend the finite. They were invented by Hamkins and Kidder in the 90's but first appeared in print in a paper authored by Hamkins and Lewis [10] in 2000. For a detailed description of the machine the reader is urged to consult that paper. We shall be assuming the reader has done so in order to fully understand this paper, but nevertheless we give a brief account here: the *hardware* of such a machine is that of a standard one-way tape of a strip of cells numbered $\langle C_i | i \in \mathbb{N}\rangle$; a *read/write* head moves in each stage one cell to the left or right (unless it is viewing C_0 in which case it may of course only move to C_1) upon reading and possibly changing the cell symbol, which we shall restrict here to being from an alphabet consisting of "0" and "1". (This symbol, in cell C_i say, will be called the *cell value*, and if the cell value at time α is $j \in \{0, 1\}$ we shall set $C_i(\alpha) = j$.) The *software* remains the standard Turing machine program, as given for example, by a transition table. The definition of [10] introduces a new state symbol q_L to add to the usual finitely many states. This symbol acts simply as one new kind of state, the *limit state*.

The novelty is in *defining* a behaviour for the machine after it has possible been through stages for every finite time n. We consider stages of "time" to be given by ordinal numbers α. This is done by fiat: at time α the head is inspecting a cell, in a certain state and its action is entirely determined by the transition table - in short it behaves literally as a standard Turing machine. However at time μ where μ is a *limit ordinal* we must specify its action which we do as follows: the head returns to the starting cell C_0, the machine enters state q_L; lastly we specify the cell values at limit time μ, $C_i(\mu)$, by setting

$$C_i(\mu) =_{\mathrm{df}} \liminf\langle C_i(\alpha) \mid \alpha < \mu\rangle = \bigcup_{\alpha<\mu} \bigcap_{\alpha<\beta<\mu} C_\beta(i).$$

To paraphrase: if the cell value *stabilizes* by stage μ then the value $C_i(\mu)$ at that time is that stabilized value. Otherwise it is set to 0. (On the right equality we are using the convention that the ordinals are the von Neumann ordinals, in particular that $0 =_{\mathrm{df}} \varnothing$ and $1 =_{\mathrm{df}} \{0\}$.)

Variants on this straightforward model are possible: one may specify that the value of a cell is *blank* at a limit time μ, whenever the cell value had changed value unboundedly in the time μ (and thus have an alphabet of three symbols); one may dispense with the special state q_L for limit stages μ, and demand that the machine enter into the state q_i where i is the liminf of the state numbers of q_j at times less than μ (this has the attractive computational feature of putting the machine, at limit stages of time, at the beginning of the outermost loop it was cycling through before μ). Finally, and this we shall adopt for this paper, just as the authors of [10] originally did, we allow for multiple tapes. We shall allow for three tapes *input*, *scratch*, and *output* arranged side by side, with the the read/write head viewing a triple of cells, one from each tape, simultaneously, and allowed to write likewise to all three such cells, again simultaneously, in a single step. (It is one of the differences between such 1 tape and 3 tape models that we shall investigate in this paper.) We continue to enumerate the cells in the same way however as $\langle C_i | i \in \mathbb{N} \rangle$.

It is quite natural then to ask "*to what extent do properties of the standard Turing machine model transfer to the ITTM model?*".

The first obvious feature is that ITTM's not only subsume standard TM's in terms of the times they are allowed, they are in essence computing at a higher type: they can read and write infinite strings of 0's and 1's which we consider as elements of Cantor space $2^{\mathbb{N}}$. We identify such $x, y \in 2^{\mathbb{N}}$ with *real numbers*. (If we wish to compute simply on integer input, we can code an integer n of course on the input tape as a 1 entered into the first n cells of the input tape, and 0's in the others.) As the programs (or transition tables *etc.*) are still clearly enumerable we let P_e be the e'th progam. If this program halts on input x (and with y on the output tape), we write $P_e(x) \downarrow$ or ($P_e(x) \downarrow y$).

Entirely appropriate questions are:

Q1 *What is:* (a) $\{e \mid P_e(0) \downarrow\}$; (b) $\{\langle e, x \rangle \mid P_e(x) \downarrow\}$; (c) $\{y \in 2^{\mathbb{N}} \mid P_e(x) \downarrow y\}$?

Q1a) and b) here are versions of the *halting problem* for such machines, a) of course is just b) restricted to the bottom type of integer inputs. Here c) raises the question of what *reals* can be computed by such machines.

Definition 1 *[10] A real $y \in 2^{\mathbb{N}}$ is called* writable *if $\exists e P_e(0) \downarrow y$. We say that an ordinal number α (or a set A) is* writable *if a code for α, or A, is writable.*

(By a *code for α* we mean a $y \in 2^{\mathbb{N}}$ so that if we set $n <_y m \leftrightarrow y(\langle n, m \rangle) = 1$ then $\langle \omega, <_y \rangle \cong \langle \alpha, < \rangle$; in this case we write "$\|y\| = \alpha$". A code for a *transitive set A* is any $z \in \mathbb{N} \times \mathbb{N}$ so that $\langle \mathbb{N}, z \rangle \cong \langle A, \in \rangle$.) Clearly only countable ordinals and countable sets in the above sense can be writable. We let WO stand for the set of reals that code wellorderings. We may ask then:

Q2 *What are the writable reals? What are the writable ordinals?*

Definition 2 *[10] $\lambda =_{\mathrm{df}} \sup\{\alpha \mid \alpha$ is a writable ordinal$\}$;*
(ii) $H(\lambda) = \{A \mid A$ is a writable set$\}$.

It is easy to see that if α is writable, then so is any $\beta < \alpha$. (If y codes α then for some n $\|y \restriction n\| = \beta$, where $y \restriction n$ gives value 1 only to those pairs $\langle k, m \rangle$ where $k <_y m <_y n$; we then modify the program that outputs y to output $y \restriction n$ instead.) The writable ordinals thus form an initial segment of the countable ordinals, and $\lambda \subseteq H(\lambda)$. It can also be shown that $H(\lambda)$ is a transitive class of sets.

Considering further the computational process, we may next consider the *halting times* of computations $P_e(x) \downarrow y$. Hamkins and Lewis partly analysed such halting times on integer inputs. They defined:

Definition 3 [10, §3] *(i) α is a* clockable *ordinal, if for some e $P_e(0) \downarrow^{\alpha}$, where the latter indicated that the computation halted in exactly α steps, that is its action at time α is to go into a halting state.*
(ii) $\gamma =_{\mathrm{df}} sup\{\alpha \mid \alpha$ is clockable$\}$.

Hamkins and Lewis proved in [10] two basic facts: (Thm 3.4) that not all ordinals below γ are clockable; (Thm 3.8) that $\lambda \leq \gamma$.

Q3 *What are the clockable ordinals? Does $\lambda = \gamma$? That is, are all clockables writable?*

Hamkins and Lewis pursued the Turing machine model by analysing appropriate notions of ITTM-*degree* corresponding to Turing degrees, and provided a wealth of results. It is the view (with the benefit of hindsight!) taken here that the emphasis is better placed on looking not at *halting* computations and their outputs, but by analysing what is the essential feature of these machines: that *any* computation stabilizes (we can regard a halting computation as a 'degenerate' case of stabilization). Let us make the following definition:

Definition 4 *Let $S(\alpha) = \langle C_i(\alpha) | i \in \mathbb{N}\rangle$ be the sequence of cell values at time α; let us call the* snapshot of the computation $P_e(x)$ at time α *to be the sequence $S(\alpha)$ together with any other appropriate specificatory information, such as the current position of the head, and the current line of the transition table.*

As there are only continuum many possible snapshots of any given computation, we see that as α increases through the ordinals there must be some stage ξ so that a snapshot $S(\xi)$ reappears at some later stage, and indeed the computation must at some stage enter an infinite repeating loop. Indeed for some computations of the form $P_e(x)$ although the machine has not formally halted, the contents y of the output tape remain unaltered from some point on ("the output tape has stabilized"). Such a real y is, in a very concrete sense, "*eventually computable*". Hamkins and Lewis's term for this was *eventually writable*.

Definition 5 [10, §3] *(i) If there is some point in time δ so that for all later δ' the content of the output tape of $P_e(0)$ is fixed with content $y \in 2^{\mathbb{N}}$ ("$P_e(0) \uparrow y$") we say* *that y is* eventually writable. *(ii) For the least such δ occurring in (i) (if it exists), we shall write $P_e(0) \uparrow^{\delta} y$. (iii) $\zeta =_{\mathrm{df}} \sup\{\beta \mid \beta$ has an eventually writable code$\}$. (iy) $H(\zeta)$ is the class of sets A with eventually writable code.*

Again, any $\zeta' < \zeta$ has eventually writable code, and $H(\zeta)$ is a transitive class of sets. We may then ask:

Q4 *What are ζ and $H(\zeta)$?*

An important point to note is that any computation that starts to cycle will in fact do so after a *countable* number of stages (a simple Löwenheim-Skolem argument shows this, or it can be done directly (see [10, Thm 1.1])). The ordinal ζ (and the set $H(\zeta)$) are thus necessarily countable.

Lastly we define a third class of sets: those whose code appears on the output tape of some computation, although such appearance may be evanescent:

Definition 6 [10, §3] *(i) A real $y \in 2^{\mathbb{N}}$ is* accidentally writable *if there is some e so that y appears on one of the computation tapes of $P_e(0)$ at some point in time. (ii) $\Sigma =_{\mathrm{df}} \sup\{\beta \mid \beta$ has an accidentally writable code$\}$; (iii) $H(\Sigma)$ is the class of sets with accidentally writable codes.*

There is the obvious version of Q4 for Σ and for accidentally writable sets. In the papers [32] and [33] we answered the above questions Q1-Q4 (and Q5 below). The first paper [33] was primarily motivated with answering the "clockables = writables?" query, and missed out on the fundamental relationship (that we called the λ-ζ-Σ-Theorem), which was published in [32]. We take the opportunity here in the first part of the paper to give a cleaner account of these arguments and recast the above theorem. We shall also use this to give a machine-theoretic proof of a theorem of Friedman and Harrington.

Before doing this we give one definition that is begging to be made.

Definition 7 *(i) $x \subseteq \mathbb{N}$ is said to be* decidable *iff there is some $e \in \mathbb{N}$ so that $\forall n \in \mathbb{N} P_e(n) \downarrow$ and $\forall n(n \in x \leftrightarrow P_e(n) \downarrow 1)$.*
(ii) $A \subseteq 2^{\mathbb{N}}$ is said to be decidable *iff there is some $e \in \mathbb{N}$ so that $\forall x \in 2^{\mathbb{N}} P_e(x) \downarrow$ and $\forall x \in 2^{\mathbb{N}}(x \in A \leftrightarrow P_e(x) \downarrow 1)$.*

We then may ask:

Q5 *What are the ITTM decidable sets of integers, or reals?*

1.2 Relationships to other work

In the immediate ITTM neighbourhood, we have indicated the pioneering paper of Hamkins and Lewis [10] where all the basic architecture, definitions, and results are stated. The same authors pursue the Turing machine analogy in [11] where they investigate Post's problem for both integer and real computation using ITTM's. Hamkins and Seabold looked at the single tape model in [12] as alluded to, and for most purposes their results show that the 1 tape model is sufficient to replace the 3 tape model.

It is one of our themes that proper comparison of ITTM degree and computational structure should not be that with ordinary standard Turing degrees, but with that obtained from *generalized recursion theory* or *higher type* recursion theory. We have pursued this analogy in [31] (the former appearing, by the vagaries of the publication processes even before [10]!) and [34]. It is our view that ITTM's provide a *model of computation* that is stronger than that of Kleene's Recursion (see [17]). However the resulting degree structure is more akin to that of *hyperdegrees* or even Δ^1_2-*degrees*. (Hamkins and Lewis's negative result to Post's problem in ITTM theory, as opposed to Post's positive solution for standard Turing degrees, points in this direction, and the results on complete sets in degrees of the *ITTM jump hierarchy* being recursively isomorphic to codes of levels of the Gödel hierarchy in [32] confirm this.) In order to view ITTM's as providing a *model of recursion* however, one needs the λ-ζ-Σ-Theorem characterisation of the relevant ordinals, and *then* at that point one can obtain a "Normal Form Theorem" (Corollary 5). This then links ITTM theory *qua* a *recursion theory* with the generalised theory of Spector classes in Higher Type Recursion Theory. Further discussion of this would take us too far afield.

Whatever one's view on the connection with higher recursion theory, one does seem to need to analyse the theory at a more abstract level (meaning not at the level of machine programs) and see the connections with lower level set theory, mainly that of the constructible sets, and in particular with that of *theory of admissible sets* (see [1]). Kripke and Platek formulated the theory KP which is a weakening of the Zermelo-Fraenkel axioms of set theory so that, in brief, we have instances of Δ_1-Separation and Σ_1-Replacement (meaning that the relevant schemes are restricted to apply to formulae in the mentioned classes). Models of KP are then called *admissible sets*.

As Kleene Recursion is tied up intimately with a higher type recursion at the level of such sets (see for example [16]), so ITTM Recursion is tied up with models of an enhanced theory of KP (namely Σ_2-KP obtained by enhancing the Replacement scheme to require Σ_2-Replacement) which enjoy an extendability property (namely that between L_ζ and L_Σ). Just as the higher type Kleene

Recursion is at the level of admissible sets, so our contention is that ITTM computation on *sets* of reals, is at the level of "Σ_2-extendable sets". However we have to make use of these sets even when analysing computation on sets of integers, *i.e.* at one type down.

This particular "liminf" inductive constructive has its parallels in other areas. (1) Löwe was the first to point out the similarities of the ITTM computation sequences with certain so-called *revision sequences* in the theory of truth of Herzberger. This was pursued in [22], [21], and [23]. In fact we now see that formally these structures are mutually interpretable: any computation can be coded into a revision sequence, so that the eventual output/tape values *etc.*, are recursive in the stability set of a Herzberger revision sequence. (2) Kreutzer [20] used essentially a revision rule akin to Herzberger's when defining a new *partial fixed point semantics*. (3) Burgess [2] abstracted from Herzbergerian revision semantics the notion of a *(arithmetically) quasi-inductive definition*. This is the mathematical analogue appropriate to procedures involving infinitary liminf rules, of (monotone) inductive definitions (and operators). Questions then relating to one area from ITTM's, revision sequences, and quasi-inductive definitions have their formal equivalents in either of the the other two. One question of foundational interest is: "How much of analysis is needed to show that ITTM computations halt or enter a loop?" One can formulate this as a statement of second order number theory, and it can be shown that this is provable in Π^1_3-CA_0 but not Π^1_2-CA_0 (see Simpson [29] for the relevant notions here).

There are connections with ordinal analysis is proof theory. Rathjen ([26]) has given an ordinal analysis of Π^1_2-CA_0 which is tied up with analysing chains of Σ_1 extendible levels of the L-hierarchy. The presumed ordinal analysis of Π^1_3-CA_0 will involve chains of Σ_2 extendibles in the L-hierarchy. Analysis of computability on integers here involves statements concerning the first link in such a chain (see Theorem 1 below).

Lastly there is the question of what these virtual machines are "good for". One aspect is that, as Hamkins and Lewis say in their introduction, is that the machines have a *defined* behaviours at limit stages: we do not have to agonize over any physically imposed constraints or configurations. We may then think of ITTM's as a mathematical laboratory for analysing other modes of infinitary or transfinite computational model making. For example, the *purely physical* models of Hogarth [13],[14], and Etesi-Németi [7] concerning possible levels of computation in various spacetimes can be simulated on ITTM's. Insights gained from the "conceptual" ITTM models can be used to feed back to those putative physical models. (It turns out that the Hogarth models can be considerably "amplified" see [30]. One should perhaps not make too much of this, since ITTM's are themselves subject to a particular infinitary rule at limit times, that might conceivably *not* be compatible with some yet-to-be-devised physical models. However in general the physical GR-models are in the case of Etesi-Németi, at the level of Δ^0_2 or "trial and error predicates" and can be modelled just on an ITTM with just a single limit point in time; the Hogarth model of [14] decides arithmetical statements, and these can be done in ω^2 many steps on an ITTM and hence in an arrangement using just infinitely many limit points - or as Hogarth does - an ω-sequence of standard Turing machines in a particular space-time arrangement that allows appropriate communication between the devices. The "amplification" referred to above, is to see that Hogarth's set-up can sustain hyperarithmetical queries, and thus can be simulated on an ITTM working through the recursive ordinals. One could go further, but the physical interpretation starts to stretch incredulity (if it has not done so already!). We do not investigate these matters further here.)

1.3 A detailed outline and a description of new results

Section 2 continues with some basic definitions and facts mostly taken from [10]. We then proceed to look at cell stabilization times Lemma 1. That the height of the ordinals ζ, Σ gives the first repeating snapshots in the universal machine program is established in Lemma 2. The main theorem characterising in terms of the notion of Σ_2-extendability of levels of the L-hierarchy, is Theorem 1.

The statement connecting all of λ, ζ, Σ is at Corollary 2 of the main Theorem 1, and the "clockables are writable" question is answered at Corollary 3. From this we can derive the *Normal Form Theorem* for ITTM computation (Corollary 5), by way of analogy with Kleene's Normal Form Theorem for Turing computation, see *e.g.* [27, §14 Thm III] or more precisely [18, §53 Thm.IX]. The last three theorems of this section are cited from other work to illustrate the relations with degree theories such as hyperdegrees, as they establish *Spector criteria* for the degree orderings (see [27, §16, Cor. XXXVI] or [28, II.Thm 7.6]). This indicates the nature of the ITTM degrees on sets of integers.

S-D Friedman asked, given the relationship between ITTM's and Gödel's L-hierarchy (as embodied in the λ-ζ-Σ-Theorem, Cor. 2, whether the ITTM viewpoint enabled any new proofs about the lower end of this hierarchy. (This hierarchy is so well studied that one could not expect that there be new set theoretical *results*, but new proofs or insights could perhaps be obtained). In Section 2 we give what could perhaps be called a purely "machine-theoretic" proof of a theorem of his and Harrington, where they in turn answered a question of Richter. Richter's question was whether the first two levels of the L hierarchy with the same Σ_2-theories (meaning the same set Σ_2-sentences in the language of set theory were true at both levels) was also the first pair with this Σ_2-extendability property. Friedman and Harrington answered affirmatively, and although one might automatically expect that the answer be positive, it required some argument. Friedman's original proof involved finestructural considerations and utilised an argument showing that in this region of the L-hierarchy one has uniform Σ_2-skolem functions (see [8]). For us the argument drops out of the fact that between $S(\zeta)$ and $S(\Sigma)$, the snapshots of the *universal machine* are those of a permanently looping cycle (in tandem with the fact that codes for levels of the L_α-hierarchy are accidentally writable for levels $\alpha < \Sigma$). This is at Lemma 8. We let the reader decide what new insight this gives, if any, on the Friedman-Harrington result.

In the second part of this paper (Sections 3 and 4) we consider a number of open questions. In Section 3 we answer:

Q6 *If* $P_e(0) \downarrow^\alpha$ *then how fast can one write down a code for* α *itself?*

We show (Lemma 9) that if α is a halting time of a program, then in fact there is another program that halts in time $\leq \alpha$ with a real code $y \in \mathrm{WO}$ on its output tape with $||y|| = \alpha$. This allows us to claim that in the Normal Form Theorem (Cor 5) we may always find a code for a course of a computation very close to the same time as the length of the original computation (uniformly on all inputs). A more precise form of this is stated at Theorem 7.

We characterise those ordinals that initiate gaps in the clockable ordinals. Hamkins and Lewis had shown that if α is admissible then no computation $P_e(n)$ halts in precisely α steps, and admissible α either start gaps in the halting times of such computations, or are already interior to one. The converse question arises:

Q7 *If* α *starts a gap in the clockable ordinals, is it admissible?*

We answer affirmatively at Theorem 8.

In Section 4 we discuss some differences between the 'standard' 3 tape machine of [10] and a 1 tape machine of [12]. The class of computable functions $f : 2^{\mathbb{N}} \longrightarrow \mathbb{N}$ is identical for both models, but there are slight differences if one wishes to consider functions $f : 2^{\mathbb{N}} \longrightarrow 2^{\mathbb{N}}$. As Hamkins and Seabold demonstrated, this arises just out of small technicality: a final stage of simulating a 3 tape computation of a function $f : 2^{\mathbb{N}} \longrightarrow 2^{\mathbb{N}}$ on a 1 tape machine cannot halt without an additional piece of, either hardware (an extra reserved cell, or extra piece of information stored in the r/w head)) or changing the alphabet (a 3 symbol alphabet allows the storage of a bit more information). In discussing the 1 tape machine Hamkins and Seabold again looked at halting times of such machines; one expects some vagaries of difference here between the two models. They catalogued many halting times, but left open whether any halting time of a 3 tape machine which was a simple limit ordinal could be the halting time of a 1 tape machine. We give a counterexample to this suggestion, Theorem 9, by showing that if α is the least Π_3-reflecting ordinal then $\alpha + \omega$ is the halting time of a 3 tape machine but not of a 1 tape version.

2 The basic λ-ζ-Σ relation.

We start our investigation of this relationship by first listing some basic facts of the ITTM concept. 2.1-2.4 are due to Hamkins and Lewis and are in [10]:

Fact 2.1 ([10, Cor. 2.3]) *Any Π^1_1 predicate of reals is decidable by some P_e: meaning that if $A \in \Pi^1_1$ then there is $e \in \omega$ so that $\forall x[P_e(x)\downarrow \wedge (x \in A \longleftrightarrow P_\varepsilon(x)\downarrow 1)]$*

Hence it is possible, given a standard method for coding hereditarily countable sets, for a machine to verify if $y \in 2^{\mathbb{N}}$ is such a code.

It is an exercise in dovetailing (see [35] and the argument at (1) of Theorem 9 below) to see that there is an algorithm so that if $B \in 2^{\mathbb{N}}$ is written on the input tape (equivalently the scratch tape) portion then after ω many steps the characteristic function of the complete Σ^B_2 set is written on the output tape (meaning that $k \in B''$ iff the k'th cell on the output tape contains a 1). This operation $B \longrightarrow B''$ is characteristic of such machines.

Fact 2.2 *$\lambda < \zeta < \Sigma$ and $H(\lambda)$, $H(\zeta)$ are transitive admissible sets which are unions of such.* (*cf* [10] 8.1, 8.2, 8.5, 8.6). *λ and ζ are highly closed ordinals: they are admissible limits of admissibles (and more).*

There is a universal machine $\mathcal{U}$ which we often consider as running simultaneously all the computations $P_e(0)$ for $e \in \omega$. We organise this of course by dividing up the scratch and output tapes of $\mathcal{U}$ recursively into infinitely many other virtual 'tapes'.

Definition 8 (cf. [10] proof of 8.6.) *We let $\sigma(\nu)$ denote the "grand sum" function that adds together all the ordinals that are coded on $\mathcal{U}$'s 'tapes' at time ν.*

We may thus think of $\sigma(\nu)$ as a temporary 'approximation' to Σ, but not in any sense that it is monotonically increasing to that limit: once $\nu > \zeta$ then $\sigma(\nu)$ remains in the interval (ζ, Σ) in an oscillatory fashion. It is easy to see also that (by the same reasons for 2.2):

Fact 2.3 *$H(\Sigma)$ is a transitive set.*

Although it turns out not to be admissible ([33, Cor 3.4]). In [10] it is shown that if one has a real $y \in \mathrm{WO}$ which is code for the ordinal α on a tape then it is easy to arrange a computation that computes, by iterating the definition of the constructible L_γ-hierarchy along the ordering coded by y, a code for L_α. (This is also done in detail in [9].) This construction proceeds formally by an induction along y and can be done inside any sufficiently closed set containing the real y. Very weak closure conditions are needed for this. Closure under the rudimentary functions (see [4]) suffices here. As Fact 2.2 asserts that $H(\lambda)$ and $H(\zeta)$ are unions of admissible sets (and are 'sufficiently closed' in the above sense), it is then straightforward to see that:

Fact 2.4 ([10, Thm. 8.6]) $L_\lambda \subseteq H(\lambda)$, $L_\zeta \subseteq H(\zeta)$.

We may thus view the above as saying that the machines as also capable of producing codes for initial segments of the L-hierarchy, at least up to ζ.

However an easy argument shows that:

Fact 2.5 (cf [33, 3.4]) Σ *is either an admissible ordinal or is a union of such; further* $L_\Sigma \subseteq H(\Sigma)$.

Proof: We give the argument in this case as we shall build on it in the sequel. We may consider variations on a universal machine $\mathcal{U}$ that pause automatically at fixed intervals of time and given any accidentally writable $y \in 2^{\mathbb{N}}$ appearing on some 'tape' of $\mathcal{U}$ at such a stage, it initiates simulations of the standard turing machines to compute the ranks $\|\{e\}^y\|$ for those $e \in \omega$, so that $\{e\}^y \in \mathrm{WO}$. (It is unproblematic for a machine to check the Π^1_1-question as to whether $\{e\}^y \in \mathrm{WO}$ by Hamkins and Lewis' result [10, 2.3] mentioned at Fact 2.1 above). Hence if $y \in H(\Sigma)$ then so are all the ordinals of the form $\|\{e\}^y\|$, and then so all sets of the form $L_{||e||^y}[y]$, from which (i) follows. **Q.E.D.**

However it is readily seen that the ITTM construction and processes are highly absolute. Thus any computation $P_e(0)$ for example can be performed inside any rudimentary closed set $\mathcal{M}$, and the computation sequence is seen to be absolute for as many stages as there are ordinals in $\mathcal{M}$. We thus have:

Fact 2.6 ([33] 3.3, 3.5) $L_\lambda \supseteq H(\lambda)$, $L_\zeta \supseteq H(\zeta)$ *and* $L_\Sigma \supseteq H(\Sigma)$ *and thus with Facts 2.4, 2.5 above we have equality as well.*

However at this stage we have not identified these ordinals in any way, nor have we any hold as to what the ITTM-decidable sets of integers may be.

It is a small but important point to note that one may have identical snapshots at different points in time $\nu < \nu'$ (or even at points $\nu_n < \nu_{n+1}$ for

$n < \omega$) without the computation being in a final infinite loop (perhaps $C_i(\nu_n) = 1 = C_i(\nu_{n+1})$, but for some $\nu_n < \xi_n < \nu_{n+1}$ we may have $C_i(\xi_n) = 0$; if $\widetilde{\nu} = \sup_n\{\nu_n\}$ we shall have $C_i(\widetilde{\nu}) = 0$). To get a permanently repeating loop in the snapshot sequence one needs to know the above scenario is not occurring: one needs to know that there are no such ξ_n times as in the example given: then we know there are no switches in value and any limit point snapshot of an apparent looping sequence really is a final, repeating loop. Such snapshots then reappear on a *closed* and unbounded class of ordinal stages.

We now prove a lemma on the stabilization points of cells C_i during a computation $P_e(0)$. We make the following definition:

$\delta_i(\sigma) \simeq \inf\{\delta < \sigma | \forall \delta' \in [\delta, \sigma)\, C_i(\delta') = C_i(\delta)\ \}$ *if the latter set is non-empty; otherwise* $\delta_i(\sigma)$ *is undefined.*

Thus $\delta_i(\sigma) < \sigma$ when it is defined.

Lemma 1 *Let* $P_e(0)$ *be any program, and let the cell values be* $C_i(\nu)$ *at time* ν*. Then* $C_i(\Sigma) = C_i(\zeta)$ *for any* $i < \omega$*. In particular:*
Either $\delta_i(\zeta)$*,* $\delta_i(\Sigma)$ *are both defined and equal, or they are both undefined.*

Proof: We let $P_f(0)$ be the program that (i) computes 'grand sum' ordinals $\sigma(\nu)$ as above; (ii) calculates $\delta_i(\sigma(\nu))$, by simulating a run of $P_e(0)$ and looking at cell values in that run.

(iii) it then writes the value of $\delta_i(\sigma(\nu))$ (if it is defined) to a reserved area of tape, R_1, but only *after* inspecting the current contents of R_1, and if that contents codes an ordinal δ_1 say, it checks that $\delta_1 < \delta_i(\sigma(\nu))$. Thus, if δ_1 is not a smaller ordinal than $\delta_i(\sigma(\nu))$, for whatever reason, then δ_1 is left on R_1, unchanged. Otherwise the code to hand for $\delta_i(\sigma(\nu))$ replaces that for δ_1 on R_1. If now $\exists \nu' < \Sigma \forall \nu \in (\nu', \Sigma) C_i(\nu') = C_i(\nu)$, that is, if $\delta_i(\Sigma)$ is defined (and so is $< \Sigma$) once $\sigma(\nu) > \nu'$ a code for an ordinal $\nu_0 \geq \nu' \geq \delta_i(\sigma(\nu))$ will be present on the segment R_1 at the end of this stage, never to be overwritten. Thus ν_0 is eventually writable, and so $\nu_0 < \zeta$. We deduce that $\delta_i(\Sigma) \leq \nu_0 < \zeta$. However it is now obvious that $\delta_i(\zeta)$ is defined and must equal $\delta_i(\Sigma)$.

Now suppose that $\delta_i(\zeta)$ is defined. Let P_g be any program that eventually has a code for the eventually writable ordinal $\delta_i(\zeta)$ on its output tape. Suppose at time $\sigma(\nu)$ P_g has on its output tape a code for δ_ν. Let $P_{f'}$ be the following modification of the above program P_f: a further task (iv) is added: on calculating $\delta_i(\sigma(\nu))$ if it sees it must write this as a new ordinal code to R_1, it additionally writes to R_2, say, a code for the least $\xi \in (\delta_\nu, \delta_i(\sigma(\nu)))$ where C_i changed value (if such exists); otherwise it does nothing. Now at stage (iv), if we suppose $\sigma(\nu)$

sufficiently large so that a code for the eventual value $\delta_i(\zeta)$ is present on P_g for all times later than ν, and then an ordinal code is written on R_2, it would stay permanently there: it is thus eventually writable. However that contradicts the definition of $\delta_i(\zeta)$! Hence C_i cannot change value throughout the interval $(\delta_i(\zeta), \Sigma)$.

Finally note that if one (equivalently both) of $\delta_i(\zeta)$, $\delta_i(\Sigma)$ is undefined, then $C_i(\zeta) = C_i(\Sigma) = 0$ by the liminf rule on cell values. **Q.E.D.**

If we let P_e be the program of the universal machine $\mathcal{U}$ we see that $\mathcal{U}$ thus has repeating snapshot sequences $S^{\mathcal{U}}(\nu)$ for $\nu = \zeta$ and Σ. The above argument (for $\mathcal{U}$ now rather than P_e) shows that no cell stabilized at ζ changes value for the universal machine in (ζ, Σ). Further, the cells stabilized at Σ are then precisely those stabilized at ζ: all other cells then have value 0 at both these times by the liminf rule.

We thus have shown the first part of *(i)* of the next Lemma. The second part of *(i)* (which we have also just established) merely emphasises this point: we do have a permanently repeating loop: the "snapshot sequence" $\langle S^{\mathcal{U}}(\tau) | \zeta \leq \tau < \Sigma\rangle$ is destined to repeat exactly with $S^{\mathcal{U}}_{\Sigma.\mu+\tau} = S^{\mathcal{U}}_{\zeta+\tau}$ for any $\tau < \Sigma$. It thus has periodicity $\Sigma - \zeta = \Sigma$.

Lemma 2 *(i) $S^{\mathcal{U}}(\zeta) = S^{\mathcal{U}}(\Sigma)$. Additionally for every i, $C^{\mathcal{U}}_i$ stable at Σ implies $C^{\mathcal{U}}_i$ does not change value in (ζ, Σ).*

(ii) Moreover (ζ, Σ) is the lexicographically least pair of ordinals satisfying (i).

Proof: We are only left with observing (ii). If (z, s) were a lexicographically lesser pair, we could write a program that, simulating $\mathcal{U}$, searches for such a pair, and may then eventually write them (indeed may halt after finding them!) Thus both $z, s < \zeta$. But then we have that the universal machine $\mathcal{U}$ has started looping before time ζ, but this is absurd as then we should have that the eventually writable ordinals all have codes in the smaller L_z - and this cannot happen as L_ζ is a union of admissible sets, **Q.E.D.**

In the sequel we shall only refer to snapshot sequences for the universal machine $\mathcal{U}$ so for brevity we shall drop the superscript on $S^{\mathcal{U}}(\zeta)$ *etc.*

Definition 9 [32, Def.2.11] $\widetilde{0} =_{\text{df}} \{e \in \omega \mid \exists y P_e(0) \uparrow y\}$.

$\widetilde{0}$ is thus a jump operator, which codes indices of all programs of eventually stable output (we include all halted programs as also being of stable output). For the universal machine $\mathcal{U}$ by inspecting the 'tapes' of the programs $P_e(0)$ we can consider it calculating, we may 'read off' by inspecting certain cell values in

$S(\zeta)$ which indices are in $\widetilde{0}$ and we have that $\widetilde{0}$ is thus (1-1) reducible to $S(\zeta)$. The converse is also true: given any cell C_i whose value we are interested in during the computation of the universal machine $\mathcal{U}$, we may write two programs $P_{e(i)}, P_{e(i)'}$ to flash its 0 value to the output tape of $P_{e(i)}$ when it changes (or to $P_{e(i)'}$ when the value becomes 1). We can thus read that value eventual value (if it exists) off from $\widetilde{0}$. We can then conclude that $\widetilde{0}$ and $S(\nu)$ are recursively isomorphic. We thus shall have:

Lemma 3 $\widetilde{0} \equiv_1 S(\zeta)$.

Lemma 4 [33, Cor.3.1] *Suppose* $\exists y P_e(0) \uparrow y$. *Then the least* δ *so that* $\forall \delta' > \delta(y$ is on the output tape of $P_e(0))$ *is less than* ζ.

Proof: This is really a corollary to (the argument of) Lemma 1: one writes a code for the least $\delta = \delta(\sigma(\nu))$ so that the whole of a simulated run of $P_e(0)$ along $\sigma(\nu)$ in the interval $[\delta, \sigma(\nu))$ has constant output tape. The argument concludes that such a δ is eventually writable, thus is less than ζ. **Q.E.D.**

Lemma 5 *If* y *is eventually writable then* $y \leq_T S(\zeta)$.

Proof: The last lemma shows that y is on a recursive slice of the universal machine's output tape which is devoted to simulating the computation of $P_e(0)$, and it is there by stage ζ. Information as to the values $y(k)$ can be read off from $S(\zeta)$ in an (ordinary Turing) recursive fashion. **Q.E.D.**

Lemma 6 ([33, Cor. 3.4]) $H(\Sigma)$ *is not admissible: there is a* $\Sigma_1(H(\Sigma))$ *definable (in the parameter* $S(\zeta)$*) function* $g : \omega \longrightarrow \Sigma$ *which is cofinal.*

Proof: Note that we have seen that $H(\Sigma) = L_\Sigma$ (Fact 2.6). Let

$$E =_{\text{df}} \{i \in \omega \mid \exists\delta\forall\beta \in (\delta, \zeta)\ C_i^{\mathcal{U}}(\beta) = C_i^{\mathcal{U}}(\beta + 1)\}.$$

It is easy to see that E is recursive in $S(\zeta)$: we let the universal machine itself be simulated by some program P_k writing down cell values from that simulation on a recursive slice of *its* scratch tape. Those that settle down provide us with a (recursive copy of) E, and this we can read off, again recursively, from $S(\zeta)$.

For $i \notin E$ let $F_i =_{\text{df}} \{\beta \in (\zeta, \Sigma) \mid C_i^{\mathcal{U}}(\beta) \neq C_i^{\mathcal{U}}(\beta + 1)\}$; then each F_i is a $\Delta_1(L_\Sigma)$-definable class unbounded in Σ. Straightforwardly by its definition, $\alpha \longrightarrow S(\alpha)$ is a Σ_1 function defined over L_Σ. If L_Σ were admissible, a simple closure argument would provide a $\tau < \Sigma$ which is a common limit point of each of the F_i. This goes as follows. We may define by a Σ_1-recursion the function g defined by

$g(0) = \zeta$,
$g(k+1) =_{\text{df}}$ the least $\gamma > g(k)$ so that $\forall i < k(i \notin E \longrightarrow F_i$ is unbounded in $\gamma)$.

That g is Σ_1 definable over L_Σ from ζ is easily checked, and moreover is defined on all of $\mathbb{N}$. If L_Σ were admissible its range would be bounded in Σ. If $\tau = \sup(\text{ran}(g))$ this would mean $S(\tau) = S(\zeta)$ contradicting the minimality of Σ in Lemma 2(ii). **Q.E.D.**

Corollary 1 *Every $u \in H(\Sigma)$ is of the form $P_e(S(\zeta)) \uparrow u$ for some $e \in \omega$.*

Proof: Suppose a code for u appears on $P_f(0)$ at time β. Suppose $\beta < g(n)$ (where $g(n)$ is taken from the last lemma). Note that $g(n)$ is $\Sigma_1(H(\Sigma))$ from the snapshot $S(\zeta)$. Given $S(\zeta)$ then by computing grand sums below Σ as usual we may search along $\sigma(\nu) < \Sigma$ running a program $P_{e(n)}(S(\zeta))$ to search for a code for the $g(n)$. Suppose this code is $y \in$ WO. If now b is the integer in the field of y coding that part of the ordering of order type β, we may then find a code for u by looking at the universal course of computation for $P_f(0)$ at the $y \restriction b \cong \beta$'th ordinal place. The overall computation may then halt with this code. **Q.E.D.**

Remark Note that there will be many snapshots $S(\zeta')$ from which a given u can be computed. Indeed for any $u \in L_\zeta$ it can be shown that the least ζ' for which this can be done is less than ζ.

We now prove the principal theorem of this section. It relates the pair of ordinal suprema of two of the kinds of ordinal the ITTM universal machine can produce: the *eventually writable* which have supremum ζ and the *accidentally writable* which have supremum Σ. The theorem appeared first in [32], but we give a clean presentation here, which we can use for a *machine-theoretic* proof of the Friedman-Harrington theorem (Theorem 2) to follow. The import of Theorem 1 (and why we are taking the trouble to reprove it here) is that it is the clear that all the corollaries, and most of the answers to the questions raised earlier can now be seen to follow from it. Previously the questions were answered in a somewhat more piecemeal fashion over several papers. We now can see that, for example, that the "Clockables are writables" Theorem (that $\gamma = \lambda$) from [33], is actually a corollary to it. Moreover the Normal Form Theorem again becomes a corollary to it. We thus *unify* the results under one umbrella. The theorem is an equivalence between two notions defined in completely different ways: ordinals appearing on the "tapes" of a computational model involving transfinitely many stages in time, on the one hand, and on the other, the first pair of ordinals appearing at an appropriate level of definability of Gödel's hierarchy of constructible sets, with which he proved the consistency of the Axiom of Choice and the Continuum Hypothesis.

We could have put this and the next Theorem 2 together and have a three way equivalence, but choose not to cloud the main event. Theorem 2 is the first example whereby a result proper to *set theory* is proven (admittedly not for the first time) by appeal to an (admittedly infinitary) *model of computation*.

Theorem 1 ([33, 2.1]) *The pair (ζ, Σ) as defined above from ITTM's, is also the lexicographically least pair satisfying either of the following conditions:*

(i) (ζ, Σ) is a pair of 'final repeat ordinals' where (a) the snapshot $S(\zeta)$ at time ζ of the universal machine $\mathcal{U}$ equals $S(\Sigma)$, the snapshot at time Σ, and (b) $S(\zeta)$ then reappears for ever with periodicity Σ;

(ii) $L_\zeta \prec_{\Sigma_2} L_\Sigma$;

Proof: That (ζ, Σ) satisfy (i) has been shown at Lemma 2. We first show that (ζ, Σ) satisfy (ii). Suppose

(2) $L_\Sigma \models \varphi(\xi) \equiv \exists u \forall v \psi(u, v, \xi)$

where ψ is Σ_0, and as an eventually writable parameter sequence we take the single ordinal $\xi < \zeta$. Let $u_0 \in L_\Sigma$ be such that

(3) $L_\Sigma \models \forall v \psi(u_0, v, \xi)$.

For e a program index, write "$P_e(0)^{\leq \tau} u$" to abbreviate " *u appears on $P_e(0)$'s computation tape at some time before or at time τ*". Make also the obvious definition replacing "$\leq \tau$" with "$< \tau$". Let e_0 be a program index so that $P_{e_0}(0)^{<\Sigma} u_0$. Let e_1 be an index which "eventually writes" a code for ξ. The following Claim proves (ii).

Claim 1 There is $\overline{u} \in L_\zeta$ satisfying $L_\zeta \models \forall \psi(\overline{u}, v, \xi)$.

Proof of Claim 1 By Corollary 1 there exists e so that $P_e(S(\zeta)) \downarrow u_0$. Thus

(4) $L_\Sigma \models \exists u(P_e(S(\zeta)) \downarrow u \wedge \forall v \psi(u, v, \xi))$.

(5) If $\zeta' < \zeta \wedge P_g(S(\zeta')) \downarrow^{<\Sigma}$ then $P_g(S(\zeta')) \downarrow^{<\zeta}$.

[Proof of (5): let ζ', g be as in the antecedent. We may find a program P_h that (i): using grand sum ordinals $\sigma(\nu)$ eventually writes down a code for the snapshot $S(\zeta')$ and (ii): letting at time ν the approximation to $S(\zeta')$ currently available be $S(\zeta')_\nu$, it writes down, if possible, a halting time $\alpha_\nu \leq \sigma(\nu)$ for a simulation $P_g(S(\zeta')_\nu)$ along $\sigma(\nu)$. This will be possible, if the latter converges before $\sigma(\nu)$. In this manner, if $P_g(S(\zeta')_\nu) \downarrow^\alpha$, then we see that α_ν is eventually α, and α is thus less than ζ.]

Thus if $\zeta' < \zeta$ is assumed to satisfy (4) and $P_e(S(\zeta')) \downarrow u_1$ then we shall have $u_1 \in L_\zeta$ and $L_\Sigma \models \forall \psi(u_1, v, \xi))$ and then by Π_1 reflection, that $L_\zeta \models \forall \psi(u_1, v, \xi))$ and we'd have:

(6) $L_\zeta \models \exists u(P_e(S(\zeta')) \downarrow u \wedge \forall v \psi(u, v, \xi))$.

We should then be finished with *Claim 1*.

We thus only need to show that there is such a $\zeta' < \zeta$ satisfying (4). So we devise along by now familiar ways, a computation that eventually writes the least such ζ' satisfying (4), thus ensuring it is less than ζ.

Let $P_k(0)$ be the program that: (i) incorporates a program $P_f(0)$ which eventually computes on its output tape a code for ξ; we set ξ_ν for the value on this incorporated output tape at time ν; (ii) computes grand sum ordinals $\sigma(\nu)$ and searches $L_{\sigma(\nu)}$ for the least ζ' such that

(7) $L_{\sigma(\nu)} \models P_e(S(\zeta')) \downarrow u' \wedge \forall v \psi(u', v, \xi_\nu))$

and writes a code for such a ζ' (if it exists) to a reserved area R_1 of the output tape; again it only does this latter step after first checking that the new ζ' is greater than the current ordinal coded into R_1. We know (by Lemma 4) that once $\sigma(\nu)$ is sufficiently large below ζ then $\xi_\nu = \xi$ holds; we also know that once it is sufficiently large, above some ordinal τ say, which satisfies $P_e(S(\zeta)) \downarrow^{<\tau} u_0$, that it certainly can write down an ordinal $\zeta' \leq \zeta$, so the process is not vacuous. We claim the only ordinals written in this process to R_1 are less than ζ.

However once $\sigma(\nu)$ is above τ, then this ζ' would remain there - it would not be the case at any *later* stage $\mu > \nu$ with $\sigma(\mu) > \sigma(\nu)$, that for any lesser $\zeta'' < \zeta'$, we should then find u'' such that $P_e(S(\zeta'')) \downarrow u''$ with u'' fulfilling the requirement (7) (because by (5) $P_e(S(\zeta'')\downarrow \longrightarrow P_e(S(\zeta''))\downarrow^{<\zeta}$, and thus $L_{\sigma(\nu)} \models P_e(S(\zeta''))\downarrow^{<\zeta} u''$ already). Likewise no snapshot $S(\zeta'')$ for $\zeta'' > \zeta$ is necessary to find our u_0. However that makes ζ' eventually writable. Hence $\zeta' < \zeta$. **Q.E.D. Claim 1**

We have thus shown:

Claim 2 $L_\zeta \prec_{\Sigma_2} L_\Sigma$.

Claim 3 (ζ, Σ) *is the least pair satisfying* $L_\zeta \prec_{\Sigma_2} L_\Sigma$.

Proof: Suppose for a contradiction that $(z,s) <_{\text{lex}} (\zeta,\Sigma)$ also satisfied $L_z \prec_{\Sigma_2} L_s$. However note by consideration of the universal machine $\mathcal{U}$ running inside L_s, that $L_z \prec_{\Sigma_2} L_s$ alone implies that firstly that $S(z) = S(s)$, and secondly that $\mathcal{U}$ is actually entering a final loop at time z - which will repeat at time s, again as part of a permanent cycle and thus both constraints (a) and (b) of *(i)* would also be satisfied. We show this more formally, letting $C_i = C_i^{\mathcal{U}}$ *etc.*:

Subclaim: if $C_i(z) = 1$ *then for no* $\nu \in (z,s)$ *will we have* $C_i(\nu) \neq C_i(\nu+1)$.

Proof: Let $D = \{\nu < s \mid C_i(\nu) \neq C_i(\nu+1)\}$, and let $\delta_i(z) < z$ is the point where cell C_i stabilized below z ($\delta_i(z)$ exists since $C_i(z) = 1$). If $L_s \models$"$\exists \nu (C_i(\nu) \neq C_i(\nu+1) \wedge \nu > \delta_i(z)$" this would go down to L_z by Σ_1-reflection, contradicting the definition of $\delta(z)$. $\square$ Subclaim

The point of the subclaim is that although we have $S(z) = S(s)$ we have to again justify that the $\mathcal{U}$ has really finally entered a looping cycle and no cell which has value 1 at time z will at some later time have value 0, in other words that constraint (b) of *(i)* holds. Having established that, we have from Lemma 2(ii) that (z,s) must be (ζ,Σ). **Q.E.D. Claim 3 and Theorem**

We shall add another characterisation to the pair (ζ,Σ) when we give a machine-theoretic proof Theorem 6 below. We now can note:

Lemma 7 *(a)* $H(\zeta) = L_\zeta$ *is* Σ_2*-admissible (and is a limit of such), and (b)* $H(\Sigma) = L_\Sigma$ *is a union of admissibles (in fact* Σ_2*-admissibles).*

Proof: (a) is proven by standard methods: Σ_2-admissibility requires closure under Δ_2 Comprehension and Σ_2 replacement schemes, and the Σ_2 extendability of L_ζ gives us (more than) this. (b) is a simple application of Σ_2-extendability of L_ζ. **Q.E.D.**

Remark 1 It can be shown that (ζ,Σ) is the lexicographically least pair such that for the universal machine $\mathcal{U}$, $S(\zeta) = S(\Sigma)$, using the "Theory Machine" of

[9]. There a very explicit calculation is made as to the points of occurrence of codes of L_α levels and their Σ_2-truth sets, for $\alpha < \Sigma$.

We mention some further results that can be obtained from the above. By an addition to Theorem 1 we obtain:

Corollary 2 **(The λ-ζ-Σ-Theorem)** $L_\lambda \prec_{\Sigma_1} L_\zeta \prec_{\Sigma_2} L_\Sigma$.

Proof: The new part here is the Σ_1-elementarity of $H(\lambda) = L_\lambda$ in $H(\zeta)$. But if $L_\zeta \models \exists u \varphi(u, \xi)$ where $\xi < \lambda$, then we run an algorithm that halts after it finds at some point an α with $\xi < \alpha < \zeta$ for which there is such a $u \in L_\alpha$ and then halts with an output code for L_α. Hence $L_\alpha \in L_\lambda$ and thus $L_\lambda \models \exists u \varphi(u, \xi)$. **Q.E.D.**

Corollary 3 **("All clockables are writables")**
$\gamma =_{\mathrm{df}} \sup\{\alpha \mid \exists e \in \omega P_e(0)$ *halts in exactly* α *steps* $\} = \lambda$. *That is, the writable ordinals are unbounded in the "clockable" ordinals of [10] Sect. 3.*

Proof: If $P_e(0)$ halts in time α then this is a Σ_1-statement true in L_Σ. By the last corollary it is true in L_λ. Hence $\gamma \leq \lambda$. However γ is easily seen to be no less than λ, since if (a code for) $\lambda' < \lambda$ is the output of $P_f(0)$, then we may follow $P_f(0)$ by the algorithm that checks through the code for λ' for wellfoundedness; this takes at least λ' steps before halting. Hence there is a clockable ordinal $\geq \lambda'$. **Q.E.D.**

Corollary 4 *The ITTM decidable sets of integers* $x \subseteq \mathbb{N}$ *are precisely those sets* $x \in \mathcal{P}(\mathbb{N}) \cap L_\lambda$.

Proof: If $x \subseteq \mathbb{N}$ is ITTM decidable, using some procedure P_e then the queries $?P_e(n) \downarrow 1?$ will all be decided by a computation of the universal machine $\mathcal{U}$ by time ζ. The statement "$\exists y(y$ codes a halting course of computation witnessing $P_e(n) \downarrow 1)$" is a Σ_1 statement in the language of set theory, and by the λ-ζ-Σ-Theorem, if true in L_Σ is true in L_λ by Σ_1-elementarity. Hence the set $x = \{n \in \mathbb{N} \mid P_e(n) \downarrow 1\}$ is a set Σ_1 definable over L_λ. Conversely, if $x \in L_\lambda$ we may set an ITTM machine running to produce a code for the set $A = L_\alpha$ where α is least with $x \in L_\alpha$. We may then adjust this machine's program to output all of x and *a fortiori* x is seen to be decidable. **Q.E.D.**

The following is an analogue of Kleene's Normal Theorem involving the T-predicate which, in the (standard) Turing case, gives integer codes for halting computations sequences. This classical theorem due to Kleene gives integer codes y (using prime power coding or some such) for a course of computation $P_e(n)$ as "inputs" to a universal T predicate $T_1(e,n,y)$. (Here the output of $P_e(n)$ would be coded into the last element of the sequence that is coded by y.) For us we have similarly a universal predicate, indeed we have the universal machine alluded to above, and this machine may simulate all $P_e(n)$ for all e and n at once. We have to have a *real* now to code the whole course of computation that is naturally transfinite in general. This real y codes snapshots of each stage of the computation. Without knowing that every halting computation halts before λ one would not be in a position to know that we could find for each e an index of a program that yields as halting outputs a code of the whole computation sequence - essentially a sequence of snapshots $S(\tau)$ for τ less than the length of the computation $\varphi_e(n)$.

We thus need to have for each length of a halting computation a writable real of that length along which we encode these snapshots. That is what the last Corollary provides.

The effectivity in the Normal Form Theorem of the transition $e \longrightarrow e'$ is obtained merely by observing that if we are interested in program P_e then the following algorithm P'_e works. "Watch the universal machine's simulation of all programs P_f. Given n some program $P_f(n)$ will halt with output some real y' coding a wellorder of length longer than it takes for $P_e(n)$ to halt; now build a code for the course of computation $P_e(n)\downarrow$ utilising the ordering y' to form y. When this code y is complete with a final stage snapshot of the halting position of $P_e(n)$, then halt."

The algorithm sketched in quotation marks is then capable of being put into ITTM terms and is thus our $P_{e'}$. Although we have not given a complete numerical blow-by-blow description of the algorithm, it should be enough to see that we have something in close analogy to Kleene's T_1. If e is the e'th standard Turing computable function, Kleene shows us that $\{e\}(n)\downarrow \leftrightarrow \exists y \in \mathbb{N}\, T_1(e,n,y)$. In the ITTM setting we see this is also essentially obtained by considering "universal" processes or machines, just as we are doing here. The universal machine $\mathcal{U}$ provides a "universal ITTM T predicate", $\mathfrak{T}_1$ say, so that we have $P_e(n)\downarrow \leftrightarrow \exists y \in 2^{\mathbb{N}}\, \mathfrak{T}_1(e,n,y)$. The above does not show the fact we establish below in the next Section at Cor. 7 that we can provide tight bounds on the times needed to produce a code y. However the above reasoning establishes the following.

Corollary 5 (Normal form theorem I) *For any ITTM computable function φ_e we can effectively find another ITTM computable function $\varphi_{e'}$ so that on any input n from ω if $\varphi_e(n)\downarrow$ then $\varphi_{e'}(n)\downarrow y \in 2^{\mathbb{N}}$, where y codes a wellordered computation sequence for $\varphi_e(n)$.*

If the reader lets us informally define (without going into all the details) $Last(y) = z$ iff y codes a halting course of computation with the last output tape containing $z \in 2^{\mathbb{N}}$, we may summarise the above discussion.

Corollary 6 (Normal form theorem I contd.) *There is an ITTM decidable predicate $\mathfrak{T}_1$ so that $\forall e \forall n$:*

$$P_e(n) \downarrow z \quad \leftrightarrow \quad \exists y \in 2^{\mathbb{N}}[\mathfrak{T}_1(e,n,y) \wedge Last(y) = z].$$

There is a higher type version obtained by relativising all the results (now for λ^x, ζ^x etc.) above to real number inputs:

Corollary 7 (Normal form theorem II) *(a) For any ITTM computable function φ_e we can effectively find another ITTM computable function $\varphi_{e'}$ so that on any input x from $2^{\mathbb{N}}$ if $\varphi_e(x) \downarrow$ then $\varphi_{e'}(x) \downarrow y \in 2^{\mathbb{N}}$, where y codes a wellordered computation sequence for $\varphi_e(x)$. (b) The universal predicate $\mathfrak{T}_1$ satisfies $\forall e \forall x$:*

$$P_e(x) \downarrow z \quad \leftrightarrow \quad \exists y \in 2^{\mathbb{N}}[\mathfrak{T}_1(e,x,y) \wedge Last(y) = z].$$

The effectivity is again established in the same way, noting that the input (whether $n \in \mathbb{N}$ or $x \in 2^{\mathbb{N}}$) does not affect the above description of an algorithm in any dynamic way.

Although this is not necessary for what follows, we may relate some of the above to the Σ_2-mastercode of L_ζ this is a constructible theoretic notion (see [4]) and essentially is a set coding the whole of the structure L_ζ, however in this arena it essentially can be taken to *be* the set T^2_ζ, the Σ_2-truth set. We define these theories in generality as follows.

Definition 10 *For $n < \omega \leq \alpha$, let $T^n_\alpha =_{\text{df}} \{\ulcorner \varphi \urcorner \mid L_\alpha \models \varphi$ & φ is a Σ_n sentence of $\mathcal{L}\}$.*

Then T^2_ζ is recursively isomorphic to those ITTM indices that index halting or eventually stable computations on zero input - this is the set $\widetilde{0}$ defined above. Let $A = A^2_\zeta$ be that Σ_2-mastercode of L_ζ. (The reader to whom the notion of mastercode is unfamiliar, may simply drop the "A" from the following.)

Theorem 2 (cf. [32, Thm. 2.6]) $\widetilde{0} \equiv_1 A \equiv_1 T^2_\zeta$.

There are also intimate connections between the jump operator derived from the *halting problem* for ITTM's. Hamkins and Lewis define:

Definition 11 [10, Sect.5] *Let $x \in 2^{\mathbb{N}}$. Then $x^\nabla =_{\text{df}} \{e | P_e(x) \downarrow\}$.*

Theorem 3 *(i)*([32, Thm 1.7]) x^∇ *is recursively isomorphic to the* Σ_1*-theory of* $L_{\lambda^x}[x]$. *In particular* 0^∇ *is recursively isomorphic to the* Σ_1*-theory of* L_λ.

(ii)[32, Thm 1.5] *The assignment* $x \rightarrowtail \lambda^x$ *satisfies* Spector's Criterion*:*

$$x \leq_\infty y \longrightarrow (x^\nabla \leq_\infty y \longleftrightarrow \lambda^x < \lambda^y)$$

where $x \leq_\infty y \Longleftrightarrow \exists e \in \mathbb{N}(P_e(y) \downarrow x)$.

One could compare the above with the analogous result that Kleene's $\mathcal{O}$ is recursively isomorphic to the Σ_1-truth set of $L_{\omega_1^{\mathrm{ck}}}$ (the latter is the level of the Gödel constructible hierarchy indexed by the least non-recursive ordinal). Given that the relation $x \leq_\infty y$ defined in the last theorem is Δ^1_2 one sees that we have here a degree notion on sets of integers that is intermediate between hyperarithmetic, or Δ^1_1 and Δ^1_2. A completely precise characterization awaits.

The notion of $\leq_\infty$ reducibility is a very natural one, when thinking of a machine model. However the nature of the machines is determined by the limit rule specification: and that is a Σ_2-notion (cell C_i's values are 0 at a limit stage unless $\exists \nu \forall \nu' > \nu(C_i(\nu') = 1)$.) The ultimate behaviour of such machines is tied in, as we have seen, with the looping behaviour of the universal machine $\mathcal{U}$ and the ordinals ζ, Σ; and here the Σ_2 nature of the machinery is apparent. We may thus define a notion of relative computability not by when a machine *halts* with some t on its output tape, but by what it *eventually* has on its output tape (if anything). There is thus a central notion of x is *eventually computable in y:*

Definition 12 [32, 2.10]) *(i) We set* $x \leq^\infty y$ *iff for some index e we have* $P_e(y) \uparrow x$. *In other words:* $x \leq^\infty y \Longleftrightarrow_{\mathrm{df}} \exists e \in \mathbb{N}(P_e(y) \uparrow x)$*;*
(ii)([32, 2.11]) *we define the concomitant notion of jump by:*

$$\tilde{x} =_{\mathrm{df}} \{e \in \mathbb{N} \mid P_e(x) \uparrow\}$$

There is an argument to be made that this reducibility, and notion of relative computability should be the fundamental notion of ITTM computability. We have seen that we can obtain the results on clockable ordinals, and on what are the decidable sets of integers etc, only *after* settling all the Σ_2 questions concerning repeating snapshots *etc.* Further note that the universal machine $\mathcal{U}$ on 0 input is constructing the whole of the L hierarchy *permanently* up to ζ (and somewhat more ephemerally, up to L_Σ). There are natural *Spector classes* (see [24, 4C]) associated with this notion of computation and [10, 7.3] demonstrates the existence of that associated with the notion of semi-decidable; similar arguments show that the much larger Spector Class is intimately associated with "eventually semi-decidable" sets of reals, and is available from the ITTM architecture.

These concepts are discussed further in [35]. As the ordinary degree structure $\leq_\infty$ turns out not to be much like Turing degrees, but more like hyperdegrees, there is less psychological need to stick with *halting* computations, as with *stable* computations - and then we regard 'halting' as a particularly stable kind of stability!

If we make this decision then ITTM definable sets can then be related to the notion of *arithmetically quasi-inductive* ([2]) which we have mentioned in the introduction. Again the relation $x \leq^\infty y$ is Δ^1_2. We then have:

Theorem 4 *(i)([32, 2.13])* $\widetilde{x}$ *is recursively isomorphic to the* Σ_2*-theory of* $L_{\zeta^x}[x]$. *In particular* $\widetilde{0}$ *is recursively isomorphic to the* Σ_2*-theory of* L_ζ.
(ii)([32, 2.12]) The assignment $x \rightarrowtail \zeta^x$ *satisfies* Spector's Criterion*:*

$$x \leq^\infty y \longrightarrow (\widetilde{x} \leq^\infty y \longleftrightarrow \zeta^x < \zeta^y).$$

Along these lines, A. Klev has defined in [19] an extension of Kleene's $\mathcal{O}$ to an $\mathcal{O}^{++}$, that mirrors exactly Kleene's original definition as a tree (indeed the tree is literally an extension of Kleene's), and is to the complete $\Sigma_2(L_\zeta)$ set what $\mathcal{O}$ is to $\Sigma_1(L_{\omega_1^{\mathrm{ck}}})$.

The relevant definition and the L-hierarchy equivalent formulation at the higher type run as follows:

Definition 13 *Let* $A, B \subseteq 2^{\mathbb{N}}$. *Then* $A \leq^\infty B$ *iff there is some program index* $e \in \omega$, *and some* $y \in 2^{\mathbb{N}}$, *so that for any* $x \in 2^{\mathbb{N}}$ *we have:*

$$\forall x \in 2^{\mathbb{N}}[x \in A \leftrightarrow P_e^{y,B}(x) \uparrow 1 \wedge x \notin A \leftrightarrow P_e^{y,B}(x) \uparrow 0.$$

Theorem 5 *Let* $A, B \subseteq 2^{\mathbb{N}}$. *Then* $A \leq^\infty B$ *iff for some pair of* Σ_2 *formulae* $\varphi_0, \varphi_1 \in \mathcal{L}_{\dot\in,\dot x,\dot y \dot B}$, *for some* $y \in 2^{\mathbb{N}}$, *we have for any* $x \in 2^{\mathbb{N}}$*:*

$$x \in A \leftrightarrow L_{\zeta^{x,y,B}}[x,y,B] \models \varphi_0[x,y,B]$$

and

$$x \notin A \leftrightarrow L_{\zeta^{x,y,B}}[x,y,B] \models \varphi_1[x,y,B]$$

where we set $\zeta^{x,y,B}$ *to be least* ζ *so that there is some least* $\Sigma > \zeta$ *with* $L_\zeta[x,y,B] \prec_{\Sigma_2} L_\Sigma[x,y,B]$.

W. Richter asked (communication with Friedman and Harrington) whether, for any $2 \leq n$, the lexicographically least pair of ordinals (ζ^n, Σ^n) with $T^n_{\zeta^n} = T^n_{\Sigma^n}$ was the lexicographically least pair satisfying: $L_z \prec_{\Sigma_n} L_s$. This was positively answered, independently by L. Harrington and by S-D. Friedman.

Theorem 6 (Friedman [8], Harrington (unpublished)) *Let $n \geq 2$. If (z,s) is the lexicographically least pair satisfying $T^n_z = T^n_s$ then $L_z \prec_{\Sigma_n} L_s$.*

For $n = 2$ this question arises naturally in ITTM theory, and adds another characterisation to Theorem 1.

Lemma 8 *If (z,s) is the least pair satisfying $T^2_z = T^2_s$ then $(z,s) = (\zeta,\Sigma)$ and hence $L_z \prec_{\Sigma_2} L_s$.*

Proof: We use the notation of Theorem 1. For a contradiction suppose that $(z,s) <_{\mathrm{lex}} (\zeta,\Sigma)$ satisfied $T^2_z = T^2_s$. Again we see that $S(z) = S(s)$ as these snapshots are coded into the Σ_2-theories. Just as in Claim 3 (of Theorem 1) these cannot be snapshots of the final looping sequence of $\mathcal{U}$. So for some i and D defined as in the Subclaim of this Claim 3 there, $\delta =_{\mathrm{df}} \sup(D) < s$ and "$d = \delta$" is a Π_1 definable relation over L_s. Note that $L_{\delta+1} \models$"δ is countable" (otherwise, $L_{\delta+1} \models$"δ is a cardinal" but then we should have many pairs $u < v < d$ with $L_u \prec L_v$). Hence every ordinal $\delta' < \delta$ is $\Delta_1(L_{\delta+1})$ definable from δ (*via* the usual L-least map of ω onto the countable ordinal δ, which is again definable over L_δ). But then we have a parameter free Π_2 definition of T^2_s over L_s as follows:

$\ulcorner\varphi\urcorner \in T^2_s \iff$
$\forall d[d = \delta \longrightarrow L_{d+1} \models$"$\ulcorner\varphi\urcorner \in T^2_{f(k)}$ where $f : \omega \twoheadrightarrow L_d$ is the $<_L$-least onto map"]

and k is such that $f(k) = z$. But the Σ_2-theory of L_s does not have a Π_2, and a Σ_2, *parameter free* definition over L_s which the above provides. So this is a contradiction. As T^2_Σ clearly equals T^2_ζ, we obtain the desired result. **Q.E.D.**

Having thus set out the basic fundamentals of the theory we turn to two sets of problems which are left unanswered by the above.

3 Halting and Writing Times

In this section we address two issues: once a machine computing, say $P_e(0)$, has stopped, how *quickly* could we produce a code for that course of computation? If for example the machine halts after α steps, the Normal Form Theorem tells us that there is a y, a code for that course of computation, but gives no information as to where/when/how to find it. In order to produce such a y, one must first start out by producing a code for α itself. Once that is done, one can run another computation that takes the given code for α and simulates along the ordering coded the original course of computation of $P_e(0)\downarrow$. For a machine to build a course of computation it must have the materials consisting of an ordering along which to work (and the original instructions embodied in P_e of course).

The first theorem below says that in fact we can compute a code for α extremely fast: in fact there is another program that will do it in $\alpha+\omega$ steps at worst.

The second issue is whether the gaps in the clockable ordinals noted by Hamkins and Lewis ([10, Gap Existence Theorem 3.4]) are all initiated at stages in time which correspond to admissible ordinals. (They had noted that admissible ordinals can start gaps, since they had shown that no admissible ordinal can be the precise length of any halting computation.) We show here that gaps *must* be started by admissible ordinals.

3.1 Quick Writing

We prove here a result which we have announced, and has been used in the literature, (see [3]), but whose proof has not yet appeared. It concerns *quick writing*: if a program halts after α many steps, how soon can a machine produce a code for α? That it *can* produce a code is the '*'Clockables are Writables*" theorem (Cor. 3). But how quickly? We previously had announced a slightly weaker result that required $\alpha+\omega$ many steps to write α. We observe here that it can be done in $\leq\alpha$ steps.

Lemma 9 *Suppose e is such that $P_e(0)\downarrow$ in exactly $\alpha\geq\omega$ steps. Then there is f such that $P_f(0)\downarrow y$ in $\leq\alpha$ steps with $y\in \mathrm{WO}\wedge\|y\|=\alpha$.*

Proof: If $P_e(0)\downarrow$ in exactly α steps then α is inadmissible (by [10, Thm 8.8]). Set $\gamma=_{\mathrm{df}}$ the largest admissible, or limit of admissibles, $\leq\alpha$. We split into cases: although these could all be absorbed into *Case 3*, the other arguments are not unilluminating.
Case 1 $\gamma=\alpha$. Then α is a limit of admissibles. The program $P_e(0)$ is halting at the limit time α because it is viewing a particular configurations of $0,1$'s in the three cells $C_0(\alpha),C_1(\alpha),C_2(\alpha)$ (recall that the read/write head views three cells on each of the tapes at the same time; at time α it is in the limit state q_L and due to a line on its transition table, immediately goes next into a halting state because of the configuration it sees. Let the triplet $\langle C_0(\alpha),C_1(\alpha),C_2(\alpha)\rangle$be $\langle k_0,k_1,k_2\rangle$. Then not all of the k_j are 1: if so there would be some previous limit stage $\alpha'<\alpha$ at which the r/w head would be viewing precisely this configuration and it would have halted earlier. So at least one of the k_j is a 0. We could now divide

into subcases as to which particular combinations of these three cells contains 0's; but let us just pretend it is the cell $C_0(\alpha)$ that has value 0 at this limit stage, and it is this appearance of a 0 at this limit point in time alone that causes the machine to halt. It therefore cannot be that C_0 contained a 0 at any previous limit time: we conclude that there is only a 0 at time α because there has been a strictly increasing sequence of stages $\beta_n + 1$ for $n < \omega$ with $C_0(\beta_n + 1) = 0 \neq C_0(\beta_n + 2)$. Perforce $\sup\{\beta_n + 1 | n < \omega\} = \alpha$, (otherwise, again, there would have been an earlier halt). We now run the universal machine $\mathcal{U}$, and look for grand sums of ordinals $\sigma(\nu)$ for $\nu < \alpha$ along the way. By admissibility, $\nu < \alpha \longrightarrow \sigma(\nu) < \alpha$ (as we can think of running the machines inside L_α). We devise a modification to this program that (i) simulates the original $P_e(0)$, and (ii) that for the first time it sees an ordinal $\gamma_0 + 1$ with (a) $C_0(\gamma_0 + 1) = 0 \neq C_0(\gamma_0 + 2)$ and (b) $C_1(\gamma_0 + 1) = 1 = C_2(\gamma_0 + 1)$ it writes a code for γ_0 to a reserved area of tape R again. At a later time when a larger γ_1 is found satisfying (a) and (b), it first checks: (c) that the values $C_1(\gamma'), C_2(\gamma')$ have been constantly 1 in the interval $[C_1(\gamma_0 + 1, \gamma_1 + 1)$ (if this fails she must scrap the contents of R setting it to 0's), she then "adds" the codes for the wellordering γ_1 to the ordinal code in R (this takes at most ω steps) to obtain (if in this case (c) did not fail), $\gamma_0 + \gamma_1$. We can organise these additions in a continuous way (as we know that we shall have exactly order type ω many successful lengthenings of these wellorderings), so that continuing in this fashion at stage α it will have a code for the unbounded sum of the ordinals β_n in the area R; this will take as many steps as the calculation of $P_e(0)$, namely α. When the simulated $P_e(0)$ halts, so does this program. A code for α in this case is then computable in precisely α steps.

Case 2. $\gamma < \alpha < \gamma + \omega$.

Hamkins and Lewis (in the '*Speed-Up Lemma*')[10, Lemma 3.3] show in general that if $\delta + n$ is clockable then so is δ. Hence there is e' so that $P_{e'}(0)$ halts in exactly γ steps. It suffices to write a code for γ in the manner of Case 1, with k further points added on the end, where $\alpha = \gamma + k$ for some $k < \omega$. This can be done (providing we write the k many extra points first!), as we have seen, in γ many steps.

Case 3. $\gamma < \gamma + \omega \leq \alpha$.

We consider the following algorithm P_h for computing truth sets of structures $\langle L_\delta, \in\rangle$. P_h effects the following:

(i) Simulates $\mathcal{U}$ the universal ITTM;

(ii) Looks on $\mathcal{U}$'s "tape" for codes of wellfounded models of the form L_β, for increasing β:

(iii) When it finds an L_β it writes T^2_β, the Σ_2-truth set for L_β to a reserved part

R_1, of its scratch tape; it does this by writing a 0 in cell C_i if the gödel number of the i'th Σ_2 sentence is in T^2_β, and 1 otherwise.

(iv) when (iii) is done, we require it to flash a Flag cell "$1, 0, 1$"; then it goes back to (ii) to look for a larger $L_{\beta'}$ to repeat the process, and overwrite T^2_β with its new values $T^2_{\beta'}$.

Each time it cycles through (ii) - (iv) the flag is flashed. The result being that at some limit stage with the flag at '0', we know we have some T_{β^*} on the scratch tape, for $\beta^* = \sup_n \beta_n$ where all $T^2_{\beta_n}$ have previously appeared on the scratch tape. Note that this "eventual theory" $T_{\beta^*} \supseteq T^2_{\beta^*}$, but in general will not be equal. (We are obliged to Sy Friedman for pointing out that we had not taken care of this in our previous proof.) Some Σ_2 sentence may be true for arbitrarily large $\gamma' < \gamma$ without being true at γ.

Here we appeal instead to Lemma 1 of [9]: which shows that the Σ_2 theory of L_{β^*}, $T^2_{\beta^*}$, is then uniformly (in β^*) recursively enumerable in this 'eventual theory' T_{β^*}. From $T^2_{\beta^*}$ we can define recursively a real coding a wellorder of length β^*. This gives us a new, larger, ordinal to work on, and the algorithm at (iv), when it returns to (ii) can continue. In short we have a procedure which over any admissible L_δ is a Σ_1-definable: if on the η'th time we loop around this process the theory $T^2_{\eta^*}$ is obtained, then $\eta \longrightarrow \eta^*$ is an increasing function Σ_1 definable over any admissible L_δ where it will define a function $\delta \longrightarrow \delta$. Moreover at time δ it will have T_δ on R_1.

As γ is a limit of admissibles, then at stage γ this process has written T_γ to R_1. We now note that the next admissible ordinal $\gamma^+ > \gamma$ is the least ordinal not recursive in T_γ, *i.e.* $\gamma^+ = \omega_{1ck}^{T_\gamma} = \omega_{1ck}^{T^2_\gamma}$. As $\alpha \geq \gamma + \omega$ there is a subset, G of ω, recursive in T_γ by some (standard) Turing reduction $\{g\}^{T_\gamma}$ for some $g \in \omega$ with G coding a wellorder of type α. However now recall that if B is any set on the scratch tape at time δ, then at time $\delta + \omega$ we may arrange for B'' to be written to the output tape. As $G \leq_T T_\gamma$ we may similarly arrange instead for G to be written by time $\gamma + \omega \leq \alpha$. To complete the Lemma we need to observe that we can also get this algorithm to halt at exactly time α: this can be effected by running the original program $P_e(0)$ that halted in time α in parallel and simultaneously with the above. Since our "arrangement" described above will be writing many different "G" as $\{g\}^{T_\delta}$ for different $\delta \leq \gamma$ we need to know when to stop. We thus run $P_e(0)$ as a clock to tell us. **Q.E.D.**

From this kind of thinking we may obtain some information extra concerning possible normal form course of computations. The division into cases above can be overcome, if we allow ourselves some more steps to play with. We can then find a bound on the times needed to produce codes in the Normal Form Theorem Cor. 5. We omit the details of the proof.

Theorem 7 *If φ_e is an ITTM computable function then we may effectively find an $e' \in \omega$ satisfying the Normal Form Theorem so that for any $n \in \omega$ if $\varphi_e(n) \downarrow$ in α steps (where $\alpha \geq \omega$) then $\varphi_{e'}(n) \downarrow y$ in $< \omega^2.(\alpha + 2)$ steps*

Proof: We briefly sketch the argument which uses the "Theory Machine" of [9]. We use the J_α Jensen hierarchy for the constructible sets (see [15] or [4]). If $P_e(n) \downarrow^\alpha$ then by absoluteness of the machine construction $(P_e(n) \downarrow^\alpha)_{J_{\alpha+1}}$, and there is a course of computation code y witnessing this with $y \in J_{\alpha+1}$. With care one can wrote this out as a parameter free definition of y which is $\Sigma_2^{J_{\alpha+1}}$. However the Theory Machine continually produces codes for levels of the J-hierarchy together with their theories, on its output tape (without ever halting), and it turns out that this machine will have a real $c = c_{\alpha+1}$ on its output tape at time $\omega^2.(\alpha+1) + \omega$ which codes the structure $J_{\alpha+1}$. Recursively in c then (using the index e to define the reduction), we may define the code y. This takes another ω many steps, and we have a code for y written out in $\omega^2.(\alpha+1) + \omega + \omega$ many steps. **Q.E.D.**

There is a similar corollary for the Normal Form Theorem II, (Cor. 7), involving real number inputs x which is again uniform in x. We leave this to the reader to formulate.

3.2 Only admissibles start gaps

We now show that only admissible ordinals start gaps. Previously it was known that many gaps were started by admissible ordinals, and that no admissible was the length of any halting computation of the form $P_e(n) \downarrow$. The short argument that follow shows that admissibility is required for this to happen. This was announced (but unproven) as Theorem 17 in [35].

Theorem 8 *If γ starts a gap in the clockable ordinals, then γ is admissible.*

Proof: We suppose this fails, and that γ_0 starts a gap, but is inadmissible. By this supposition then:

$$L_{\gamma_0} \models \forall n \in \omega \exists \alpha \psi(n, \alpha, \xi)$$

for some Σ_1 ψ, some ordinal parameter ξ, where ψ is defining a function $F : \omega \to \gamma_0$ monotone cofinally. Note that γ_0 is a limit of clockable ordinals, (otherwise it could not be starting a gap in the clockables, but would be interior to one!) Hence $\zeta < \zeta' < \gamma_0$ for some clockable ζ' and by applying Lemma 9 as above, there is a program index f' producing in ζ' steps a code for ζ' and by a trivial modification that writes down a suitable initial segment, an index f can be effectively found from f' so that $P_f(0)^\tau \downarrow y_\zeta \in \mathrm{WO} \wedge ||y_\zeta|| = \zeta$ and $\tau \le \zeta'$.

We consider the following algorithm P: it simulates P_f until this has converged. We have, by using a universal machine, that there is a program index $e_0 \in \omega$ so that φ_{e_0} continually computes (without ever halting) a code on its scratch area for levels L_σ of the L-hierarchy, where σ is the (current) grand sum of all ordinals coded on all halted output computations as simulated by the universal machine.

Once P_f has converged, we run P_{e_0} at the same time checking within the L_σ's constructed whether $L_\sigma \models \exists\alpha\psi(n,\alpha,\zeta)$ for each n. It lists those n for which this holds to a piece of scratch tape, and flashes the lead cell C_0 1,0,1 each time a new n is added to the list (otherwise C_0 is untouched). P_e finally halts if at any limit stage C_0 contains a 0. But it is easy to check that this happens exactly at γ_0. Hence γ_0 does not start a gap. Contradiction! **Q.E.D.(Theorem 8)**

This finishes our results on the "standard models" of ITTM's.

4 3 Tape and 1 Tape limit halting times

We consider now the 1 tape model of [12]. Here, the 3 tape model is replaced with a single tape, and a read/write head viewing just one single cell at a time. In all other respects the machinery, software, behaviour at limit times *etc.*, is identical to the 3 tape machine. It is shown that in [12] that the 1 tape machine is not exactly the same in its behaviour as the 3 tape machine: the classes of computable functions on integers, or even function $f : 2^{\mathbb{N}} \longrightarrow \mathbb{N}$ are identical. However there is a difficulty of simulating 3 tape computations for functions $f : 2^{\mathbb{N}} \longrightarrow 2^{\mathbb{N}}$. One way around this difficulty is by allowing a 3-symbol alphabet. We outlined one method for this in [31], where we allowed a 'B' for a blank symbol as well as $0, 1$, and the B (for ambiguity) would be written at a limit stage λ in a cell that had changed value cofinally in λ. Such an arrangement is also fully powerful as regards the 3 tape model. In [35] we discussed further the relationship of the 1 tape model to the 3 tape model, principally in terms of classifying what is on the tapes at limit ordinal stages. One advantage of the 1 tape model is that one can get a definable pointclass of sets that is *adequate* (in the sense of Moschovakis [24] p.158). With a 3 tape model, and the R/W head reading 3 cells simultaneously at the start position, one obtains classes one step up in a *difference hierarchy* which are not adequate. (Adequacy is desirable as it allows for parametrization and some closure properties to be enjoyed). The paper [35] further classified descriptive set-theoretically pointclasses on 1 tape machines defined by halting times. A very nice suggestion coming from work on ordinal length tape Turing machines (from Dawson and Koepke) is that we do not place the R/W back on cell C_0 at a limit time λ, but on the liminf of the cell positions up to time λ, and moreover the machine state q_i at time λ is the liminf of the machine states up to this time also. This very elegantly puts the current 'instruction' to be at the outermost loop of any subroutines that were being enacted up to time λ. This would probably alter the nature of some of the results just mentioned.

The following Lemma holds for both the standard $0,1$ valued 3 tape and 1 tape models, for an enumeration of the cells $\langle C_i \mid i < \omega\rangle$. It also holds for the 1 tape model using blanks *mutatis mutandis*. Roughly speaking the tape's contents at a reasonably closed ordinal time γ is Σ_2 definable over L_γ; after a further ω many steps of calculation we expect to get a double (standard) Turing jump of that to be written to the tape. Hence Σ_4 definability of those cell values is in order here:

Lemma 10 *In the language $\mathcal{L}_{\{\dot{\in},\dot{x}\}}$, there are Σ_4 formulae $\varphi_0(v_0,v_1)$, $\varphi_1(v_0,v_1)$, so that during any computation of the form $P_e(x)$, for any primitively closed ordinal γ: $C_m(\gamma+\omega) = i$ $(i < 2)$ if and only if $L_\gamma[x] \models \varphi_i[x,m]$.*

Proof: First assume only that γ is a limit stage in the computation of $P_e(x)$. Let s_0 be the "snapshot" at stage γ; that is let s_0 code $\langle C_i(\gamma) | i < \omega\rangle$. We note just as above that there is an (ordinary) Turing recursive total function F, so that if $n < \omega$ "$F(s_0 \restriction n) = u$" iff u is a sequence of length n representing the cell sequence $\langle C_i(\gamma+n) | i < n\rangle$ after n stages of the operation of $P_e(x)$ beyond γ. Then

Claim 1) "$F(v) = u$" is $\Delta_1(L_\gamma)$.

Now assume that γ is reasonably closed, for example, p.r. closed. Then

Claim 2) "$C_i(\alpha) = j$" is $\Delta_1^{L_\gamma[x]}(\{\alpha\})$.

Claim 3) "$u = s_0 \restriction n$" is $(\Sigma_2 \vee \Pi_2)(L_\gamma[x])$.

Proof: We exhibit a formula demonstrating this:
$u = s_0 \restriction n \iff \mathrm{Fun}(u) \wedge \mathrm{dom}(u) = n \wedge \mathrm{ran} u \subseteq 2 \cup B \wedge \forall i \in \mathrm{dom}(u)$

$[(u(i) = j \wedge \exists \overline{\gamma} < \gamma \forall \gamma' > \overline{\gamma}(\gamma' < \gamma \longrightarrow C_i(\gamma') = j) \quad \vee$
$(u(i) = j = B \wedge \forall \overline{\gamma} < \gamma \exists \gamma' > \overline{\gamma}(\gamma' < \gamma \wedge C_i(\gamma') \neq C_i(\gamma' + 1))]$

Q.E.D.Claim 3)

Claim 4) $C_m(\gamma + \omega) = i < 2 \iff$
$\exists k < \omega \forall l > k(\forall u((u = s_0 \upharpoonright l) \longrightarrow (F(u))_m = 1))$.

The formula on the right hand side of Claim 4) yields our result. **Q.E.D.**

Theorem 9 *There is a limit ordinal $\gamma + \omega$ which is the halting time of a 3 tape machine computation of the form $P_e(0)$ for some e; but which is not the halting time of any 1 tape machine program.*

Proof: Let γ be least that is Π_3-reflecting. Then γ is in fact an admissible limit of admissibles. (Any Π_2 reflecting ordinal is admissible, and the admissibility axioms themselves allow a Π_3 axiomatisation.) We note for later that L_γ has no proper $L_\alpha \prec_{\Sigma_1} L_\gamma$ substructures (for that it would have to be much more than Π_n-reflecting for all n).
Claim 1) γ is 3 tape clockable.
Proof: By the minimality of γ, it is not Π_4-reflecting, *i.e.* there is a sentence σ and a Π_2 formula $\psi(v_0, v_1)$ so that $\sigma \iff \forall n \exists m \psi(n, m)$ is true at L_γ but at no earlier level. We may assume (as we have done) that without loss of generality, the quantifiers in σ are natural number quantifiers (this again follows from the leastness assumptions on γ that every $x \in L_\gamma$ has a parameter free Σ_1 definition.)

We run the algorithm P_e above, of Case 3 of Lemma 9 which writes Σ_2 theories to a scratch tape. By the comment there, the admissibility of γ ensures that T^2_γ is indeed the tape's contents at time γ.

We shall modify this procedure so that the resulting program halts at stage $\gamma + \omega$. By the remark following the description of P_e (concerning p.r. closed ordinals, and the fact that admissibles are limits of such), we shall have that the Flag of P_e at stage γ is a '0'. We'll make the additional assumption that this Flag of P_e was in fact the cell C_1. We assume also we have left C_0 free for our use now. We add the condition that whenever the Flag (*i.e.* C_1) is 0, at a limit stage, which it recognises by being in the limit state, q_l, we change it to 1, and also ensure that C_0 is 0 also; When additionally the Flag is '0' at a limit (as opposed to the flashing 1,0,1 at some successor stages in the above description of P_e) we then arrange to have inspected in the next ω many steps the current Σ_2-truth set which we can assume is present on the tape as described above. We set up C_0 so that it flashes $0, 1$ each time we find some $\langle 0, k_0 \rangle, \langle 1, k_1 \rangle, \ldots$ so that for some i $\ulcorner \psi(i, k_i) \urcorner$ appears in this truth set.

Subclaim At $\gamma+\omega$ we have $C_1 = 1, C_0 = 0$, and this is the first limit where this happens. Hence we can modify P_e to halt exactly at stage $\gamma+\omega$ when this happens.
Straightforward. **Q.E.D.** ***Claim 1)***

Claim 2) $\gamma+\omega$ is not 1 tape clockable.
Proof: Deny for a contradiction.
First suppose $\gamma+\omega$ was clockable *via* an algorithm that was programmed to halt when C_0 contained a 1 and thus it did so because $\exists n \forall m > n \; C_0(\gamma+m) = 1$. This is $\Sigma_4(L_\gamma)$, and also by Π_3-reflection must reflect down to some $\overline{\gamma} < \gamma$. That is, $C_0(\overline{\gamma}+\omega) = j$. Contradiction! Hence γ was clockable using an algorithm that halts on 0 and in which the contents of C_0 altered cofinally in $\gamma+\omega$. Thus we must have for some $0 < n < \omega$ "$C_0(\gamma+n) \neq C_0(\gamma+n+1)$". This is $\Delta_3(L_\gamma)$ (see (3) of the proof of Lemma 10). By Π_3-reflection we shall have that $C_0(\overline{\gamma}+n) \neq C_0(\overline{\gamma}+n+1)$ for unboundedly many $\overline{\gamma} < \gamma'$, for some $\gamma' < \gamma$ and the program would thus halt at such a γ'. **Q.E.D.** ***Claim 2)***

For completeness we just additionally note:
Claim 3) $\gamma+\omega+1$ is 1 tape clockable.
Proof: With the extra step we can run the argument of (1) and inspect the cell C_1 and halt at the right point. **Q.E.D.Theorem 9**

References

[1] K.J. Barwise. *Admissible Sets and Structures*. Perspectives in Mathematical Logic. Springer Verlag, 1975.

[2] J.P. Burgess. The truth is never simple. *Journal for Symbolic Logic*, 51(3):663–681, 1986.

[3] V. Deolalikar, J.D. Hamkins, and R-D. Schindler. $P \neq NP \cap co-NP$ for the infinite time Turing machines. *Journal of Logic and Computation*, 15:577–592, October 2005.

[4] K. Devlin. *Constructibility*. Perspectives in Mathematical Logic. Springer Verlag, Berlin, Heidelberg, 1984.

[5] F.R. Drake. *Set Theory: An introduction to large cardinals*, volume 76 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1974.

[6] H. Enderton. *Elements of Set Theory*. Academic Press, New York, 1977.

[7] G. Etesi and I. Németi. Non-Turing computations via Malament-Hogarth space-times. *International Journal of Theoretical Physics*, 41(2):341–370, 2002.

[8] S. D. Friedman. Parameter free uniformisation. *Proceedings of the American Mathematical Society*, to appear.

[9] S. D. Friedman and P. D. Welch. Two observations concerning infinite time Turing machines. In I. Dimitriou, editor, *BIWOC 2007 Report*, pages 44–47, Bonn, January 2007. Hausdorff Centre for Mathematics.

[10] J. D. Hamkins and A. Lewis. Infinite time Turing machines. *Journal of Symbolic Logic*, 65(2):567–604, 2000.

[11] J. D. Hamkins and A. Lewis. Post's problem for supertasks has both positive and negative solutions. *Archive for Mathematical Logic*, 41:507–523, 2002.

[12] J. D. Hamkins and D. Seabold. Infinite time Turing machines with only one tape. *Mathematical Logic Quarterly*, 47(2):271–287, 2001.

[13] M. Hogarth. Does general relativity allow an observer to view an eternity in a finite time? *Foundations of Physics Letters*, 5(2):173–181, 1992.

[14] M. Hogarth. Deciding arithmetic using *SAD* computers. *British Journal for the Philosophy of Science*, 55:681–691, 2004.

[15] R. B. Jensen. The fine structure of the constructible hierarchy. *Annals of Mathematical Logic*, 4:229–308, 1972.

[16] K.Hrbacek and S.Simpson. On Kleene degrees of analytic sets. In H.J.Keisler J.Barwise and K.Kunen, editors, *Proceedings of the Kleene Symposium*, Studies in Logic, pages 347–352. North-Holland, 1980.

[17] S.C. Kleene. Recursive quantifiers and functionals of finite type I. *Transactions of the American Mathematical Society*, 61:193–213, 1959.

[18] S.C. Kleene. *Introduction to Metamathematics*, volume I of *Bibliotheca Mathematica*. North-Holland, Amsterdam, 1964.

[19] A. Klev. *Magister thesis*. ILLC Amsterdam, 2007.

[20] S. Kreutzer. Partial fixed point logic on infinite structures. In *Annual Conference of the European Association for Computer Science Logic (CSL)*, volume 2471 of *Lecture Notes in Computer Science*. Springer, 2002.

[21] B. Löwe. Revision sequences and computers with an infinite amount of time. In *Proceedings of XVIII Deutscher Kongreß für Philosophie der AGPD.*, Okt. 1999.

[22] B. Löwe. Revision sequences and computers with an infinite amount of time. *Journal of Logic and Computation*, 11:25–40, 2001.

[23] B. Löwe and P. D. Welch. Set-theoretic absoluteness and the revision theory of truth. *Studia Logica*, 68(1):21 – 41, June 2001.

[24] Y. Moschovakis. *Descriptive Set theory*. Studies in Logic series. North-Holland, Amsterdam, 1980.

[25] P-G. Odifreddi. *Classical Recursion Theory: the theory of functions and sets of natural numbers*. Studies in Logic. North-Holland, Amsterdam, 1989.

[26] M. Rathjen. Recent advances in ordinal analysis: Π^1_2-CA and related systems. *Bulletin of Symbolic Logic*, 1(4):468–485, 1995.

[27] H. Rogers. *Recursive Function Theory*. Higher Mathematics. McGraw, 1967.

[28] G.E. Sacks. *Higher Recursion Theory*. Perspectives in Mathematical Logic. Springer Verlag, 1990.

[29] S. Simpson. *Subsystems of second order arithmetic*. Perspectives in Mathematical Logic. Springer, January 1999.

[30] P. D. Welch. Turing unbound: the extent of computation in Malament-Hogarth spacetimes. *British Journal for the Philosophy of Science*, to appear.

[31] P. D. Welch. Minimality arguments in the infinite time Turing degrees. In S.B.Cooper and J.K.Truss, editors, *Sets and Proofs: Proc. Logic Colloquium 1997, Leeds*, volume 258 of *London Mathematical Society Lecture Notes in Mathematics*. C.U.P., 1999.

[32] P. D. Welch. Eventually infinite time Turing degrees: infinite time decidable reals. *Journal for Symbolic Logic*, 65(3):1193–1203, 2000.

[33] P. D. Welch. The length of infinite time Turing machine computations. *Bulletin of the London Mathematical Society*, 32:129–136, 2000.

[34] P. D. Welch. Post's and other problems in higher type supertasks. In B. Löwe, B. Piwinger, and T. Räsch, editors, *Classical and New Paradigms of*

Computation and their Complexity hierarchies, Papers of the Conference Foundations of the Formal Sciences III, volume 23 of *Trends in logic*, pages 223–237. Kluwer, Oct 2004.

[35] P.D. Welch. On the transfinite action of 1 tape Turing machines. In S.B. Cooper, B. Löwe, and L. Torenvliet, editors, *New Computational Paradigms: Proceedings of CiE2005, Amsterdam*, volume 3526 of *Lecture Notes in Computer Science*, pages 532–539. Springer, 2005.