Chapter 1

Discrete transfinite computation models

P.D. Welch

School of Mathematics,
University of Bristol,
Bristol, BS8 1TW,
England

1.1. Introduction

1.1.1. *The Contents*

In the past few years there has been a resurgence of interest in discrete models of computation that are allowed to act *transfinitely*. Such conceptual machines act in simple steps or stages, and have as a paradigm the *standard Turing machine*. This, during its progress moves one cell at a time, to the left or the right along an unbounded tape that it is reading, and subsequently alters symbols, changes states and moves on. This paradigm has been with us for 70 years, and for much of this chapter we shall consider variants of such a device.

Our models will all be *discrete* acting computational digital models. We shall consider how Turing and other computing machines could possible behave when allowed to perform *supertasks* (by which we mean that they are allowed to complete an infinite sequence of tasks or operations). Such a machine is usually envisaged with an unbounded tape. If supertasks are allowed then naturally the whole of that tape comes into play, and we can imagine that tape already having some characteristic function written on it. The machine can then act on that tape and is then essentially a computer acting *at a higher type*, namely that of infinite sequences of $0, 1$'s, in other words of real numbers.

Surprisingly, even if one restricts one's model to, say, a register machine

model, where the registers are finite in number with natural number entries, then simply defined behaviour at transfinite limit stages of computation lead to quite powerful decision procdures. This chapter will look at these as well. Whereas at the finite level the power of Turing and register machines is the same, at the transfinite level they diverge markedly.

We shall not deal here with any machine that is, broadly speaking, an analog machine or computes in an analog fashion. Indeed apart from Section 2.1 we shall not be entering into any discussion of *physical* viabilities, feasibilities *etc.* We thus do not wish to discuss various machine proposals that could be seen to fall under the rubric we are setting ourselves, in that they seek to to compute functions whilst being constructed to conform to some ambient physical theory or constraint. We have in mind models such as Davies [Davies (2001)], and the models of Beggs and Tucker [Beggs and Tucker (2007)] that attempt to compute any set of natural numbers, by some kinematic-based device, and thus do so within a fragment of Newtonian mechanics; nor do we consider the interaction of standard machines with physically based 'advice' functions or oracles, such as is done in [Beggs *et al.* (2008)]. We also shall not particularly consider *classical quantum computation*, as functions computed by such models are also turing computable.

Section 2.1 is somewhat of an exception, in that we do consider a particular construction in detail due to Etesi and Németi [Etesi and Németi (2002)], and to Hogarth [Hogarth (1992)], that allows Turing machines to be placed or arranged within particular spacetimes to allow for the algorithmic decision of Π_1 (and beyond) predicates without supertask phenomena. We can calculate somewhat precisely, the bounds to what can be computed in such models. We also recount the observation from [Welch (2008)], that *separability* of the spacetime manifold puts a universal countable bound on formal systems of computation within that spacetime (under some mild assumptions).

A major lacuna is that we do not consider finite automata on infinite graphs, or in particular on infinite binary trees. This would have fitted well in any discussion of discrete models, and is admittedly a serious omission. Lastly this chapter is not about computation with an algebraic or structural flavour, and we include here, *inter alia*, the Blum, Shub, Smale model of computation on the reals ([Blum *et al.* (1989)]). Although we do want to consider computation on the reals, $\mathbb{R}$ is considered primarily as unordered Baire space, or Cantor space, and there is not any other algebraic or structural feature that distinguishes the models considered here.

1.1.2. *Argument*

In the 1960's and 70's much research was undertaken deriving ultimately from Kleene's theory ([Kleene (1959)], [Kleene (1963)]) of *recursion in higher types* (*Generalized Recursion Theory, Kleene Recursion*), and the pioneering work of Aczel, Gandy, Moschovakis amongst others, that would lead to *Spector classes*, the *theory of inductive definitions*, and the theory of *admissible sets*, Barwise, Kripke, Platek.)

Our motivation is that we wish to revisit some of these older theories and results and see how some recent activity in a class of models of computation fits in the older picture. Some of these later models are no more than familiar models with different style of inputs (register machines on ordinals say); others such as Infinite Time Turing Machine of Hamkins and Kidder ([Hamkins and Lewis (2000)]) are versions of the standard model adapted to enable larger computations to be performed by allowing transfinite sequences of operations or stages; yet a third class simply consists of 'standard computation' placed in an unconventional framework (Turing machines stacked up, or regarded as inhabiting particular spacetimes). We want to see how these models fit in with our conceptions of recursion and computation formed in the earlier period.

We emphasise the logico-mathematical part of this, in particular the descriptive set theoretical descriptions. No apology is needed for this: to understand the model is to understand what the model produces: if this transcends that produced by finitary operations (in whatever form) then we are obliged to consider the underlying set-theoretical fundamentals. We consider further in Section 5 the connections of the computation models to the theory of inductive definitions (either monotone or not), and to subsystems of second order arithmetic. We do not consider this an accurate account on historical principles and we do not even claim to do justice to the concepts and individuals involved, but are merely taking a snapshot, as we rush past a fast-evolving subject.

1.2. Computation on Integers

We start ahistorically in terms of the published literature, but with a fact that surely must have been known to early recursion theorists such as Post: that allowing a Turing machine to at least run out indefinitely allows for the printing of the characteristic function not just of the halting problem (as a complete Σ_1-set) but of a Δ_2-set. This has been called 'truth in the limit'.

A Turing machine may answer any Π_1-question, *if* it is allowed ω-many stages: given a recursive predicate $R(v_0, v_1)$ we may program a machine to investigate in turn $R(0,n), R(1,n), \ldots, R(k,n) \ldots$ in turn, and if for some k $\neg R(k,n)$, then we require it to halt with a "0" for "no" as output. If the machine does not halt, then were we able to "transcend time" we could look back and say that the machine verified $\forall v_0 R(v_0, n)$. If we assume the machine has an output tape as well as a scratch tape, and if we assume the output tape starts out with every cell having a "1" written to it, we could dovetail all the queries $?\forall v_0 R(v_0, n)?$ for each n, and have the machine change a 1 to a 0 as soon as it verified that $\forall v_0 R(v_0, n)$ failed. Then after ω many stages, if we still could look at the output tape, we'd have written the characteristic function of the Π_1 set $A =_{\mathrm{df}} \{n \mid \forall v_0 R(v_0, n)\}$. If we asked the question, *Is A non-empty?* this is a Σ_2-query: $\exists v_1 \forall v_0 R(v_0, v_1)$? We cannot, in general, answer this, without allowing ourselves some further infinitary operation.

Putnam, much later, made the following definition:

Definition 1.1. *(Putnam [Putnam (1965)])* P is a *trial and error predicate* if and only if there is a general recursive function f such that for every $x_1, \ldots, x_n$

$$P(x_1, \ldots, x_n) \equiv \lim_{y \to \infty} f(x_1, \ldots, x_n, y) = 1$$
$$\neg P(x_1, \ldots, x_n) \equiv \lim_{y \to \infty} f(x_1, \ldots, x_n, y) = 0.$$

Putnam asked, and answered, the question as to the complexity in the arithmetical hierarchy of such predicates: they are $\Delta_2 = \Sigma_2 \cap \Pi_2$ in the arithmetical hierarchy. We obtain the truth of $P(x_1, \ldots, x_n)$ "in the limit" as $y \longrightarrow \infty$.

If we imagine the recursive function f as being computed by a Turing machine M writing its 0/1 output to a particular cell C_0 on its tape, the clauses above amount to a prescription of M's behaviour on any input $x_1, \ldots, x_n$ that the contents of the cell C_0, after computing in turn $f(x_1, \ldots, x_n, 0)$, $f(x_1, \ldots, x_n, 1), \ldots$, $f(x_1, \ldots, x_n, y), \ldots$, "settles down" as y increases: it must be either eventually 1/0 depending. We may thus rephrase Putnam as:

$P \subset \mathbb{N}^n$ *is* a trial and error predicate *if there is a Turing machine* M_0 *so that*

$P(x_1, \ldots, x_n) \Longleftrightarrow$ *the eventual value of* M_0*'s output cell on input* n *is 1*
$\neg P(x_1, \ldots, x_n) \Longleftrightarrow$ *the eventual value of* M_0*'s output cell on input* n *is 0.*

Continuing with this model for a moment, one sees that if there is in advance a fixed bound on the number of alternations that M_0 makes on

the output cell C_0's value, then that knowledge allows us to compute the characteristic functions of predicates in particular levels of the *difference hierarchy* of Σ_1 sets. Briefly: we say that $P(\vec{x})$ is in the k'th level of the difference hierarchy, if there are Σ_1 sets $Q_0, \ldots, Q_{2k-1}$ with $P(\vec{x}) \leftrightarrow \bigvee_{i \leq k}(Q_{2i-2}(\vec{x}) \wedge \neg Q_{2i-1}(\vec{x}))$; if we specify that M_0' may only write to the cell C_0 at most $2k+1$ times, then M_0' may decide $P(\vec{x})$. Note that it is the fixity of the value k that determines the complexity within the class of Boolean combinations that is capable of being decided by such an arrangement. However even allowing k to be unbounded, is not quite sufficient. The following can be shown.

Fact *As long as the number of times that M_0' can change its mind about the value of C_0 is a recursive function, $f(\vec{x})$ say, of the input $\vec{x}$, then still such predicates do not exhaust Δ_2: to put it another way, there cannot be any recursive constraint on the number of alterations if the process is to decide all Δ_2 predicates.*

There is the possibility of using the output of one Turing Machine as input to another. For functions on integers, this could be regarded as just composition of recursive functions. If we allow the output of a machine after ω many stages, either as a truth-in-the-limit operation or otherwise then we can display this as an infinitary operation on an infinite sequence of 0's and 1's (or whatever the alphabet of the machine is). How one does this, or under what preconditions one allows such models to be considered depends on how fastidiously one takes exception to 'supertasks', the latter being roughly defined here as a process that at some stage has completed infinitely many subtasks.

We now consider various mechanisms for this.

1.2.1. *Transcending the finite through stacking Turing Machines*

One obvious objection to an infinitely running process is the 'Thomson Lamp' ([Thomson (1954-55)]) objection: if a switch has been thrown at times $\frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \ldots, \frac{n}{n+1}, \ldots$ at what position is it at, or what value does it have, at time "1"? Similarly if a cell on the turing machine tape has changed value infinitely often from 0 to 1 and back again at finite stages, what value should we allot it at 'time' ω? Placing models in certain spacetimes neatly sidesteps this puzzle.

1.2.1.1. *General Relativistic models: Malament Hogarth Space-times*

Pitowsky [Pitowsky (1990)] gave an account of an attempt to define space-times in which the effect of infinitely many computational tasks could be seen by an observer O_r in that spacetime. (By 'spacetime' we mean here a Hausdorff, paracompact, Riemannian manifold which is a solution to the Einsteinian GR equations - we refer the reader to [Hawking and Ellis (1973)].) He used the example of the, at the time, unresolved Fermat's Last Theorem but we can consider any other task that involves looking at a Π^0_1 question: for example, the consistency of the axioms of Peano Arithmetic, or Goldach's Conjecture, both of which involve a simple universal quantifier $\forall n$ over a recursive predicate. Let us take the latter: another observer O_q performs the tasks of checking that each even number in turn is the sum of two primes. This they do along their worldline γ_1, the proper time of which is infinite (as each calculation takes some finite unit of time). If they find a counterexample they send a signal to O_p travelling along her worldline γ_2. The point of the example is to arrange that the proper time of γ_2 is finite, and has the the whole of γ_1 in her chronological past. As Earman and Norton [Earman and Norton (1993)] mention, there are problems with this account not least that along γ_1 O_p must undergo unbounded acceleration. Since then more sophisticated spacetimes due to Hogarth [Hogarth (1992)] and Etesi & Németi [Etesi and Németi (2002)], have been devised. For the moment we follow formally [Earman and Norton (1993)] to define:

Definition 1.2. $\mathcal{M}{=}(M, g_{\mathrm{ab}})$ is a *Malament-Hogarth (MH) spacetime* just in case there is a time-like half-curve $\gamma_1 \subset M$ and a point $p \in M$ such that $\int_{\gamma_1} d\tau = \infty$ and $\gamma_1 \subset I^-(p)$ (where τ denotes proper time and $I^-(p)$ the causal past of p).

This seemingly makes no reference to the word-line of the observer $\mathcal{O}_p$ travelling along their path γ_2, but they point out that there will be in any case such a future-directed timelike curve γ_2 passing through a point $q \in I^-(p)$ to p such that $\int_{\gamma_2(q,p)} d\tau < \infty$, with q chosen to lie in the causal future of the past endpoint of γ_1. The important point is that the *whole* of γ_1 lies in the chronological past of $\mathcal{O}_p$. As Hogarth showed in [Hogarth (1992)] such spacetimes are not globally hyperbolic, thus ruling out many standard space-times (such as Minkowski space-time). (See [Earman and Norton (1993)] for a discussion on global hyperbolicity and a family of Penrosian Causal Censorship Hypotheses in this context - this is a interesting debate

on how one might add extra axioms to GR to limit the types of spacetimes permissible - but would take this chapter too far off course.)

To obtain a spacetime as above, they take Minkowski spacetime $\mathcal{N}_0 = (\mathbb{R}^4, \eta_{ab})$ and choose a scalar field Ω which is everywhere equal to 1 outside of a compact set C, and which rapidly goes to $+\infty$ as the point r is approached. The point r is removed and the MH spacetime is then $\mathcal{N} = (\mathbb{R}^4 \backslash \{r\}, g_{ab})$, where $g_{ab} = \Omega^2 \eta_{ab}$. Ω and γ_1 can be chosen so that γ_1 is a timelike geodesic. This 'toy' spacetime is pictured on the left.

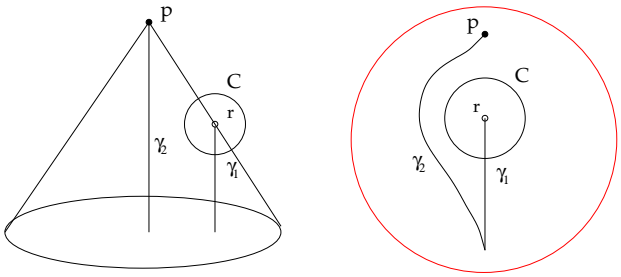

Fig. 1.1. A 'toy' MH Spacetime. On the right Hogarth's representation.

Earman and Norton discuss various possible spacetimes already in the literature that conform to being MH: Gödel spacetimes are MH but are causally 'vicious'; anti-de Sitter spacetime is MH, but fails a strong energy condition; Reissner-Nordstrom spacetime meets this, but as in all MH spacetimes there is divergent blue-shift of the signal to $\mathcal{O}_p$; further, of the unbounded amplification of signals that $\mathcal{O}_p$ may have to receive, *etc., etc.* Again this is beyond the terms of our discussion here, and apart from the rotating Kerr black hole solution of Etesi & Németi [Etesi and Németi (2002)] to which we shortly turn, our aim is only to analyse the logico-mathematical possibilities inherent in these models.

1.2.1.2. *Etesi & Németi's rotating black hole model*

The authors consider an observer sent axially into the region containing a rotating black hole of a certain size, and rotating at certain speeds - a Kerr solution. The first notable feature of such a black hole is that the primary singularity is ring-formed around the axis of rotation (see [Hawking and Ellis (1973)]). The observer, $\mathcal{O}_p$, is sent along the axis of rotation and receives signals sent from a turing machine that is orbiting forever around

the black hole. The machine is again looking for counterexamples, say, to a Π_1-predicate and will transmit one if it is found to $\mathcal{O}_p$. A clear desideratum for them is

Assumption 1 "No swamping" : it should not be the case that any part of the machinery or any observer should have to transmit or receive infinitely many signals.

Initially, the orbiting machine and $\mathcal{O}_p$, should send and receive respectively, a single signal: the witness to the failure of the Π_1 predicate under inspection. They further remark though (their Prop. 2) that in fact the computational arrangement allows deciding queries $?n \in R?$ for sets R slightly more complicated than Π_1 or Σ_1: R can be taken as a union of a Σ_1 and a Π_1 set for example. Indeed they indicate an argument (at their Proposition 3), that if the machine-observer $\mathcal{O}_m$ is allowed to send k different signals, (they take $k = 2$) then any k-fold Boolean combination of Σ_1 and Π_1 sets $R = \bigcap_{i<k-1}(S^i \cup P^i)$ (with $S^i \in \Sigma_1$ and $P^i \in \Pi_1$) can be decided. They ask how far in the arithmetical hierarchy this can kind of argument can be taken. The discussion above concerning Δ_2 predicates shows:

Theorem 1.1. [Welch (2008)] *The relations $R \subseteq \mathbb{N}$ computable in the Etesi-Németi model form a subclass of the Δ_2 predicates of $\mathbb{N}$; this is a proper subclass if and only if there is a fixed finite bound on the number of signals sent to the observer $\mathcal{O}_p$.*

We have seen (at the Fact above) that for a Δ_2 predicate R there is no recursive bound on the input n as to how many times the machine will have to change its mind concerning whether $n \in R$, and *a fortiori* no fixed in advance finite bound. Of course for checking whether any one n is in R, finitely many signals will suffice; hence only if the architecture of the experiment allows for *potentially unboundedly* many signals, then (and only then) can Δ_2 predicates be decided, still without breaking *Assumption 1*.

1.2.1.3. *Hogarth's Arithmetically Deciding Spacetime Regions*

Hogarth names a "unit" or "region" of spacetime that is capable of deciding a Π_1 question as above a "SAD_1 spacetime or region", and as a shorthand denotes it by the right hand diagram above at Fig 1.1. Hogarth in [Hogarth (1994)], (and in the later [Hogarth (2004)]) stacks up such regions to finite depths in order to answer Π_n queries.

If a spacetime contains a sequence $\vec{O} = \langle O_j | j \geq 0 \rangle$ of non-intersecting

open regions such that (1) for all $j \geq 0$ $O_j \subseteq I^-(O_{j+1})$ and (2) there is a point $p \in M$ such that $\forall j \geq 0$ $O_j \subseteq I^-(p)$ then $\vec{O}$ is said to form a *past temporal string* or just *string*. To decide membership in a Π_2-definable set of integers $P(n) \equiv \forall a \exists b Q(a, b, n)$ he then stacks up a string of regions taking each O_j as a SAD_1 region, each looking like the component of the right of Figure 1.1, with O_0 being used to decide $\exists b Q(0, b, n)$. If this fails a signal is sent out to $\mathcal{O}_p$; but if this is successful, a signal is sent to O_1 to start to decide $\exists b Q(1, b, n)$ *etc.* Ultimately, putting this all together, again $\mathcal{O}_p$ receives a signal if $\neg P(n)$, or else knows after a finite interval that $P(n)$. It should be noted that

Assumption 2 The open regions O_j are disjoint

and still that no observer or part of the machinery of the system has to send or receive infinitely many signals (thus the "no swamping" assumption is kept). This whole region is then dubbed a "SAD_2" spacetime.

A "SAD_{n+1}" spacetime is defined accordingly as composed from an infinite string of (again disjoint) SAD_n regions O_n, again all in the past of some point p. (Earman and Norton [Earman and Norton (1996)] show that a SAD_1 spacetime cannot decide Π_2 statements, and Hogarth [Hogarth (2004)] follows this up with the generalisation that SAD_j cannot decide Π_{j+1} statements.) In the Figure 2.2 below, on the right is the underlying tree structure of a SAD_3 region for computing queries of the form $?n \in A?$ for some Σ_3 set A: each large circle represents a SAD_2 region (which in turn contains a string of infinitely many of the small circles - pictured by the terminal nodes of the tree - representing the SAD_1 regions) which can be used for computing the answers to Σ_2 queries. Correspondingly, in Figure 1.2, if each O_n is a SAD_j region then the diagram on the left is that of an SAD_{j+1} region.

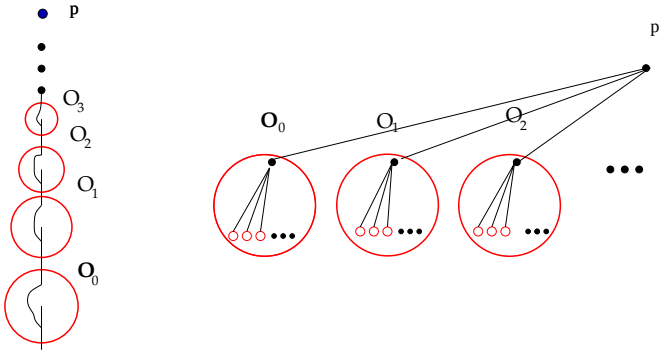

Fig. 1.2. A SAD_3 region as a past temporal string of SAD_2 regions; and its tree representation (right).

In Fig.1.2, each region O_n contains itself an ω sequence of SAD_1 regions which are shown in (the enlarged) circles of the tree.

We thus have that Π_{n+1} questions can be decided by allowing nested stacks of suitable SAD regions to depth n (if we define the depth of the simplest region in Fig. 1.1 as being 0). He then puts these altogether:

Definition 1.3. A spacetime (M, g_{ab}) is an *arithmetic deciding (AD) spacetime* just when it admits a past temporal string of disjoint open regions $\vec{O} = \langle O_j | j \geq 0 \rangle$ with each O_j a SAD_{j+1} region.

Of course there are many unresolved difficulties with this. There is a *recognition problem*: how, for example, could one ever recognise an AD spacetime region if such existed? Let alone one equipped with appropriate ranks of Turing machines coordinated and ready to go?

We now see how to go beyond Hogarth and observe that there is really no reason for us to stop at arithmetic. Hogarth has defined regions SAD_{n+1} containing stacked SAD_n subregions to a fixed finite depth n. He thus has used a subset of the class of *finite path trees* to label his regions. In the next definition $\mathbb{N}^{<\mathbb{N}}$ denotes the set of all finite sequences of natural numbers. $\mathbb{N}^{\mathbb{N}}$ denotes the set of all such infinite sequences from $\mathbb{N}$ to $\mathbb{N}$.

Definition 1.4. A *finite path tree* is any subtree $(T, <_T)$ of $\widetilde{T} = \langle \widetilde{T}, <_{\widetilde{T}} \rangle = \langle \mathbb{N}^{<\mathbb{N}}, \supseteq \rangle$where all branches under $<_T$ are of finite length.

We assign *ordinal ranks* to the nodes of a finite path tree (which are necessarily wellfounded) by recursion: the *rank of* T is then the rank of the empty sequence, (), the topmost node. A tree is in general infinitely splitting (a given node sequence node u in $\mathbb{N}^{<\mathbb{N}}$ may have in-

finitely many immediate one step extensions), even though all branches are of finite length; hence ranks of nodes can in general be infinite, but of countable ordinal height (for an account of this and the following context, see *e.g.* [Rogers (1967)] Sect.15.2). Finite path trees in general can be used to describe the construction of the *Borel Sets* on spaces such as $\mathbb{N}^k \times (\mathbb{N}^{\mathbb{N}})^l$ for any $k, l < \omega$. A space, taken for simplicity as, $\mathbb{N} \times \mathbb{N}^{\mathbb{N}}$ has a topology constructed from *basic open sets* typically of the form $U_{\langle s,p\rangle} =_{\rm df} \{\langle s, x\rangle \in \mathbb{N} \times \mathbb{N}^{\mathbb{N}} \mid s \in \mathbb{N} \wedge \exists k \in \mathbb{N}(x \upharpoonright k = p)\}$ where $p \in \mathbb{N}^{<\mathbb{N}}$.

Definition 1.5. (The Borel Hierarchy). (i) $X \subseteq \mathbb{N} \times \mathbb{N}^{\mathbb{N}}$ is in Σ_0 and in Π_0 if it is a basic open set in the above topology; (ii) $X \in \Pi_\xi$ iff $cX \in \Sigma_\xi$; (iii) $X \in \Sigma_\xi$ iff $X = \bigcup_n A_n$ where each $A_n \in \Pi_{\xi_n}$ for some $\xi_n < \xi$; a set X is *Borel* if for some countable ordinal ξ $X \in \Sigma_\xi$.

It is well known that this hierarchy is built up progressively through ω_1 many stages (where ω_1 is the first uncountable cardinal), and then no further sets are added (that is $\Sigma_{\omega_1} = \Pi_{\omega_1} = \Sigma_{\omega_1+1}$). Of particular interest is the *hyperarithmetical hierarchy* which is in one sense the constructive part of the Borel hierarchy. Here the construction of the Borel set is given by a *recursive finite path tree* (meaning the tree T and its extension relation $<_T$ are given by computable functions) with a recursive assignment of recursively open sets to the bottommost rank 0 nodes, that is to the leaves of the tree. Membership then in an hyperarithmetic set of integers (that is taking $l = 0$ in the above) is given by testing a recursive protocol of queries. Already one construal of Hogarth's AD spacetime region is that it is capable in the above notation of answering questions concerning *some* unions of arithmetic sets, $S \in \Sigma_\omega$. Why 'some'? Because the description of the union $\bigcup_n A_n$ must be given to us in an effective, *i.e.* recursive way. The upshot is that for any hyperarithmetic set $H \subseteq \mathbb{N}$ there could be constructed a spacetime region SAD_H for which queries $?n \in H?$ could be answered. Such a region satisfies Assumptions 1 and 2 and consists of SAD regions of smaller rank stacked according to the recursive finite path tree description for the construction of H.

A discussion and the details of the above can be found in [Welch (2008)]. Can a "hyperarithmetically deciding spacetime" by analogy with Hogarth's AD deciding spacetime be constructed? It can, if we can enumerate those turing programs that describe hyperarithmetic set building protocols.

Proposition 1.1. *(*[Welch (2008)]*) If $\langle e_i \mid i \in \mathbb{N}\rangle$ enumerates those indices of Turing programs that construct in the above sense hyperarithmetic sets*

S_{e_i}, via recursive trees, we may define a single MH 'hyperarithmetically deciding", HYPD, spacetime region in which any query of the form ?$n \in S_{e_i}$? can be answered in finite time.

Here we piece together regions that are "S_{e_i}-deciding" just as the AD-deciding region is built. Given input $\langle i, n\rangle$ to an initial control machine, it activates S_{e_i} and asks if ?$n \in S_{e_i}$? Of course this query will result in subqueries activating regions of lower rank down the tree coded by e_i which are themselves S_{e_j}-deciding *etc.*

At this point the reader will well, I think, object that the recognition problem has now got well out of hand: the collection of indices $\langle e_i \mid i \in \mathbb{N}\rangle$ enumerating hyperarithmetic set constructions is itself well beyond recursive or arithmetic, forming as it does a Π^1_1-complete set of integers. However it is worth emphasising that no machine in this tree array is itself performing "supertasks" (*i.e.* performing infinitely many actions in its own proper time), but if it issues a signal to another process, it does so only once after a finite amount of its own proper time. It is simply that the overall tree no longer has a recursive description, and its ordinal rank is no longer a recursively given ordinal. We have not violated our two core assumptions. However the point should be that anthropomorphic considerations are being put aside and we are calculating what is feasible given the kind of techniques Hogarth contemplates. We have here what might be called a *hyperarithmetic computer*.

1.2.1.4. *A universal constant upper bound for any computation*

Nevertheless if we take this discussion to its logical conclusion, one might ask how far could one possibly go building regions of higher and higher complexity without violating the core idea?

There is in any case a bound on the depth of any finite path tree to which we can assign MH regions without violating *Assumption 2*.

Definition 1.6. Let $\mathcal{M} = (M, g_{\mathrm{ab}})$ be a spacetime. We define $w(\mathcal{M})$ to be the least ordinal η so that $\mathcal{M}$ contains no SAD region whose underlying tree structure has ordinal rank η.

Note that $0 \leq w(\mathcal{M}) \leq \omega_1$. Here a zero value $w(\mathcal{M})$ implies that $\mathcal{M}$ contains no SAD regions whatsoever, that is, is not MH; the upper bound is for the trivial reason that every finite path tree is a countable object and so cannot have uncountable ordinal rank.

Proposition 1.2. *For any spacetime* $\mathcal{M}$, $w(\mathcal{M}) < \omega_1$.

Proof: Assumption 2 says that for different η the different SAD_η component must occupy disjoint open regions O_η of the manifold. However the manifold is separable (which follows from paracompactness and being Hausdorff). Let $X \subset M$ be a countable dense subset of M. Then each open region O_η of M contains members of X. As disjoint regions contain differing members of X there can only be countably many such regions $O_\eta \subset M$, and therefore a countable bound. QED

This is just the usual argument that separability of the real continuum $\mathbb{R}$ implies that any family of disjoint intervals of $\mathbb{R}$ must be countable. Consequently if $\mathcal{M}_{\mathrm{actual}}$ is *our* spacetime, (modelled using these basic assumptions) then $w(\mathcal{M}_{\mathrm{actual}})$ is a constant giving an upper bound to the complexity of MH regions, and so putative computations performable in $\mathcal{M}_{\mathrm{actual}}$. Dropping either of the Hausdorff or paracompactness properties from our list of properties of manifolds would seemingly result in unrecognizable (in current terms) 'spacetimes'. In short, although MH-spacetimes allow, at the most generous, for a reorganisation of any *countable* length computation (in some formalism, such as Turing machines) into one computation using trees of countable depth, this would be impossible for *uncountably* long (or many) computations whose stages occupy discrete spacetime regions. The same restriction would of course be true for any other system, or arrangement, of computations and is nothing to do with Hogarth style formalizations: this holds for any *separable* manifold and any generalised computation that requires a disjoint region of spacetime for each step or unit of computation. Somewhat more formally:

Proposition 1.3. *Let* $\mathcal{M} = (M, g_{\mathrm{ab}})$ *be a spacetime. Let* $\mathcal{F}$ *be some formal mechanism of computation, such that each computation step of the mechanism occupies a disjoint open neighbourhood of the manifold. Then there is a countable upper bound* $w(\mathcal{M}, \mathcal{F})$ *to the lengths of the* $\mathcal{F}$*-computations in* $\mathcal{M}$.

The Proposition is not a completely precise mathematical statement, since we have not defined 'formal mechanism', but the point we hope should be clear. We have not specified 'step' or 'unit' but again this can mean a Turing machine instruction step, or a cell unit. Anything that occupies a discrete interval in space-time, whether it be an MH-spacetime, (so as to avoid supertask like phenomena), or in other spacetimes more generally

with supertasks envisaged: one cannot in advance arrange the formalism to occupy uncountably many distinct open neighbourhoods. Hence the bound. If we are allowed to play God and are handed a separable manifold and (a set of integers coding) a countable ordinal α, *in advance*, then indeed we could cook up an MH manifold to accommodate computations of that length (of proper time), using stacked Turing machines, or any other form of computational model $\mathcal{F}$. What we cannot have is one manifold $\mathcal{M}$ that will work for our chosen $\mathcal{F}$ for all countable α.

1.2.2. *Allowing supertasks*

By means of using spacetime regions of a particular type the Etesi-Németi and Hogarth models avoid considering any supertasks, where any observer has performed infinitely many tasks in his or her chronological past. If we relax this constraint we may ask of our computing machines what they are capable of when given some well-defined behaviour in the transfinite. Amusingly even simple machines can perform a lot.

1.2.2.1. *Punch hole machines*

We first consider a simple kind of turing machine. We envisage such a machine as having a tape, infinite in one direction, thus with a leftmost starting cell, and a read/write head traversing the tape in the usual fashion. The alphabet of the machine is simple: it consists of a blank and a '0'. The latter we can think of as a hole that the machine punches. Thus the machine can only write once, or punch a hole, in a cell; otherwise it 'reads' and moves a cell left or right in the usual fashion. The program or transition table for such a machine is simply that of an ordinary machine of this architecture.

We have to specify what happens at stage ω and subsequent stages. There are several possibilities: but let us say that we simply allow the machine to run for ω many stages, then consider what is on the tape (a potentially infinite sequence of holes and blanks) as input to be fed back into the machine at its starting state again for the next ω many stages. We thus reset the R/W head to first leftmost cell, and let it run the program afresh.

So, ignoring difficulties with "hanging chads", such cells are usable once only. One easily sees that in ω steps again Δ_2, or trial-and-error predicates are decidable. One may simply arrange that any new calculation extends beyond the scratch area of tape already used up. If one has a Δ_2 predicate $P(v_0)$ it is not hard to arrange this so that a machine will punch holes

(using additional blank gaps) on an output 'sub-tape' so that the correct final 0/1 value is recorded. Now we have allowed the possibility to reset the head to its starting position, and let the machine continue running. We let ourselves do and re-do this process, at every limit stage in time, pulling the head back to the start position and letting it work on the accumulated tape-full of punch holes. Could we calculate more? For which predicates $P(n)$ of natural numbers n is there a machine of this kind that halts for a given n with the correct $P(n)\backslash\neg P(n)$ answer? These machines were first considered by Hamkins and Kidder but they discarded them as too weak, in favour of the *Infinite Time Turing Machine* to follow. Surprisingly perhaps, they can still calculate quite a lot: we have the following observation (due to S-D.Friedman and the author). Calling the above arrangement an "infinite punch tape machine", it is not hard to demonstrate:

Proposition 1.4. *(i) Precisely the* arithmetical predicates *are decidable by infinite punch tape machines; (ii) any computation either halts by, or is in an infinite loop, by time* ω^2.

1.2.2.2. *Infinite Time Register Machines (ITRM)*

Koepke and Miller [Koepke and Miller (2008)] consider the following register machine model. M_N is a Shepherdson-Sturgis register machine (see [Shepherdson and Sturgis], or as described in, *e.g.* Cutland [Cutland (1980)]). M_N has N registers $R_i (i < N)$ each of which may contain a natural number. Suppose that the program under consideration has instruction set $\vec{I} = I_0, \ldots, I_q$. Let us say that at time t R_i contains $R_i(t) \in \mathbb{N}$, and that instruction $I(t)$ is about to be performed. We adopt a slightly more subtle behaviour than that for the punch-hole machines. We consider the state list $q_0, q_1, \ldots, q_p$ of the machine and at time λ where λ is any limit ordinal, we say that the machine will next perform the instruction numbered $I(\lambda) =_{\mathrm{df}} \liminf_{\alpha \longrightarrow \lambda} I(\alpha)$ where $I(\alpha)$ is the instruction number about to be performed at time α. This formulation has the rather pleasant effect of placing the machine at time λ, at the start of the outermost nested loop that it entered (if any) unboundedly often before time λ.

We have to assign register values, and here of course a register may have changed value infinitely often.

For $i < N$ we set:

$\overline{R}_i(\lambda) =_{\mathrm{df}} \liminf_{\alpha \to \lambda} R_i(\alpha)$ *and if this is finite we set* $R_i(\lambda) = \bar{R}_i(\lambda)$. *If infinite we set* $R_i(\lambda) = 0$.

It is this 'resetting' of a register that gives the model its strength. We

may additionally consider such a machine to be able to consult an oracle: thus there is an instruction, so that if $Z \subseteq \mathbb{N}$, a register can be reset to 0 if $R_i(\alpha) \in Z$. Computations relative to an oracle Z can be regarded in this manner as using the set Z as an input; the infinite time available allows all of Z to be consulted. We discuss the strength of this model below, but again surprisingly complicated predicates can be calculated.

1.2.2.3. *Infinite Time Turing Machines (ITTM)*

This model, due to Hamkins and Kidder, awakened recent interest in transfinite computational models. It was designed in the 90's but an account of them only appeared in [Hamkins and Lewis (2000)] much later. We give a computationally equivalent version to that of their original model - and discuss the differences afterwards.

We go back to the punch-hole machine described above, but we now consider what to do if the machine is allowed to reuse cells. Clearly a cell may then be reused infinitely often and we must define a behaviour for it. We consider Turing machines with an alphabet of just three letters: 0,1, and B (for blank). We suppose that its standard program has states $q_0, \ldots, q_k$. The read/write head moves according to the description given by the usual transition table, but that also uses the *liminf of cell positions* that it has been reading at limit stages of time to calculate its position at a limit stage, (if this *liminf* of the r/w head positions is infinite at a limit stage, then the head is set back by fiat to the starting cell C_0). Further we set the state $q(\lambda)$ at limit times to be the liminf of previous states. (This has the same effect of course of positioning the r/w head at outermost loops as it did for ITRM's.) If the cells of the machine are enumerated $\langle C_i \mid i \in \mathbb{N}\rangle$ with values at time ν denoted by $\langle C_i(\nu) \mid i \in \mathbb{N}\rangle$ then we set at limit time λ:

$C_i(\lambda) = k \iff \exists \alpha < \lambda \forall \beta < \lambda(\alpha < \beta \longrightarrow C_i(\beta) = k)$*for* $k \in \{0, 1, \mathrm{B}\}$;
Otherwise C_i *is set to a* 'B'.

Thus if the machine has changed its mind unboundedly often below λ about the cell value then, this is set to a blank - for ambiguity if you will. Programs, are simply standard turing machine ones, and may be enumerated as $\langle P_e \mid e \in \mathbb{N}\rangle$. If a particular program P_e halts, then we can consider the contents of the tape the output of that machine. We may also prime the tape with an infinite string x from the alphabet, and consider $P_e(x)$ to be the computation of the e'th program on input x.

Such machines can decide Π^1_1-predicates. We illustrate by means of the complete Π^1_1-predicate on integers: those $e \in \mathbb{N}$ so that the e'th (standard) recursive function, $\{e\} = f$, computes the characteristic function of a *wellorder* of $\mathbb{N}$. Given input e the machine simulates the e'th standard turing program and writes the output characteristic function on, say the cells of the tape C_i where $i \equiv 0(\mathrm{mod}\, 3)$. The other cells are blank for scratch work. This takes ω many stages. When this is complete, the machine then checks the Π_2-condition of this characteristic function f coding a *discrete linear ordering*. (This is of the form $\forall n \exists m R(n, m, f)$ and R is recursive. This can be verified in ω steps, by starting with $n = 0$, recording '0' on the scratch tape cells C_i where $i \equiv 1(\mathrm{mod}\, 3)$, searching for an m, using the scratch tape cells C_i where $i \equiv 2(\mathrm{mod}\, 3)$, for auxiliary calculation, then proceeding to $n = 1$ if successful, *etc.*) If this test is passed, we have an order $<_e$; we then need to test for wellorderedness. We wipe clean the scratch tape area, and search for the $<_e$'th least element of the ordering. We may do this by simply guessing a least element on a scratch tape, and then continually revising our guess $<_e$-downwards each time we find a lesser one. If after ω many steps we did not find such then we did not have a wellorder, and we can output a 0; if after ω many steps we only changed our minds finitely often, then we indeed located the $<_e$-least element, say it was 23, and we have it written concretely on the scratch tape. We now proceed through the code function f and erase all mention of the element 23. This leaves us with a new code f' of a discrete linear order written on the cells $\langle C_{3i} \mid i \in \mathbb{N} \rangle$, and we simply now repeat this process. There are only two outcomes: either at some point we arrived at the situation where the 'current' linear order is illfounded, and we discover this fact, by descending through it infinitely often looking for its least element, or else we end up emptying out the C_{3i} cells for $i \in \mathbb{N}$ completely: after looking through this slice of the tape, and seeing it is empty (which takes a final ω many steps) we verify that it was truly wellfounded. If the order type of the ordering was α then this has been achieved in, rather roughly, $\omega + \omega.\alpha + \omega$ many steps.

Once the reader has convinced themselves of this, it is not hard to imagine programs that write out successfully on some slice of the scratch tape (which we might as well call 'output tape') those e_i with $\{e_i\} \in WO$. Moreover, one can also imagine a machine coding the *ordinal sum* of all the recursive ordinals $\alpha < \omega_1^{\mathrm{ck}}$ and outputting a code for that, *i.e.* ω_1^{ck}, on the output tape. However as Hamkins and Lewis showed, and we shall see later, this is only scratching the surface.

Suppose we denote by $P_e(n)$ the e'th computation on integer input of n, represented by an infinite string of n 1's followed by an infinite string of 0's. Several natural questions arise.

Q1 *What is* $0^{\nabla} = \{e | P_e(0)\downarrow\}$*? (The halting problem on integers).*

Q2 *What are the halting times that arise? That is, if* $P_e(0)\downarrow$ *halts in* α *steps how large is* α?

Q3 *What are the decidable predicates? Where we say* $R(n)$ *is* semi-decidable *if there is some* e *so that* $R(n) \leftrightarrow P_e(n)\downarrow 1$,*and is* decidable *if both it and its complement are strongly semi-decidable.*

[Hamkins and Lewis (2000)] first developed the theory of such machines, using the analogy of the standard turing machine as a source of the notion of recursion: they note that there are versions of the Recursion Theorems, and the Snm-Theorem for this notion of computation and there is a universal machine with a universal program. Much of the standard development proceeds very smoothly but of course there are considerable differences: for Turing machines the whole run of a halting computation, the *snapshots* of the states and of tape's contents *etc.* can be encoded by an single integer; it is thus of the same *type* as the objects on which it operates. However for ITTM's, computations $P_e(n)$ on integer input, are in general a transfinite sequence $\mathcal{S} = \langle S_\beta \mid \beta \leq \alpha\rangle$ of snapshots of the cell values $\langle C_i(\beta) \mid i < \omega\rangle$ at each stage $\beta \leq \alpha$. A computation is then an infinite object and must be coded in this context by a set of integers or a 'real' number. (One uses reals that code wellorderings of length $\alpha + 1$ and attaches by pairing functions the snapshots to the nodes of the wellorder, together with any auxiliary information such as machine state *etc.* along the way). Computations are thus of different types from the integer inputs. A central representation of standard turing machines comes *via* Kleene's T-predicate, yielding a canonical *Normal Form Theorem.* This theorem allows one to proceed effectively from e, and uniformly in n, from a halting computation of the form $P_e(n)\downarrow$ to a program e' so that $P_{e'}(n)\downarrow$ will halt, and moreover produce an integer output which codes the whole *course-of-computation* that demonstrates $P_e(n)\downarrow$. For such a notion to work in this new area we should need the program $P_{e'}$ to be capable of producing the reals needed to code the potentially transfinitely many steps in calculations such as that of $P_e(n)\downarrow$. But are they capable of this? In short, *Is every ordinal length of a halting computation on an integer input capable of being itself 'written' or being the output of some other computation?* This must be true if we are to have a hope to produce a Normal Form theorem. Hamkins and Lewis called the halting time ordinals the *clockable* ordinals, and the question

they asked is: *Is every clockable ordinal writable?*

The answer turns out, thankfully, to be affirmative, (it follows from the λ, ζ, Σ-Theorem below, [Welch (2000)]). We refer the reader not to the original papers, but to [Welch (2009)] for a later but somewhat tidier account of this theorem and the answers to the above three questions.

From this one gets the desired representation theorem (where $P_e(n)$ refers to ITTM computation).

Theorem 1.2. (Normal Form Theorem I [Welch (2004)],[Welch (2009)]) $\forall e \exists e' \forall n \in \mathbb{N}$

$P_e(n) \downarrow \longrightarrow (P_{e'}(n) \downarrow y$ *where* $y \in 2^{\mathbb{N}}$ *codes a wellordered course-of-computation sequence for* $P_e(x) \downarrow)$.

Moreover the map $e \longrightarrow e'$ *is effective (in the usual Turing sense).*

There is a higher type version obtained by relativising all the results (now for λ^x, ζ^x etc.) above to real number inputs. Part (b) below is simply a variant form stated to be reminiscent of the Kleene T-predicate. We let φ_e be the (partial) function computed by P_e.

Corollary 1.1. (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) There is a* universal predicate $\mathfrak{T}_1$ *which 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.

However the proof that all clockable ordinals are writable proceeds via an analysis of how each single cell C_i behaves during the course of a computation $P_e(n)$. In general cells may *stabilize* on some fixed value, or forever change value. The same is true for infinite sub-segments of the ITTM tape. Suppose we reserve cells C_i ($i \equiv 2 \bmod 3$) for "output" then we say that a real $y \in 2^{\mathbb{N}}$ is '*eventually computable*' if there is an ITTM computation $P_e(n)$ - which is not required to halt - but which has y as the characteristic function of the output tape from some point in time onwards. The notion is then that a computation need not formally halt in order to 'produce' an output: it is sufficient that the output tape segment be stable from some point onwards. Hamkins and Lewis called an ordinal α *eventually writable*,

if there was an eventually computable $y_\alpha \in 2^{\mathbb{N}}$ coding α. Clearly we can consider any halting computation a special case of an eventually stable one, and thus if λ is the supremum of all writable ordinals, and ζ the supremum of the eventually writable ordinals then $\lambda < \zeta$. Evaluating λ, ζ turned out to be tied up with calculating stabilisation points of cells C_i in the universal machine calculations, and the following characterisation is possible.

Theorem 1.3. (The λ, ζ, Σ-Theorem) (*cf.*[Welch (2009)]) *(i) Any ITTM computation $P_e(n)$ on integers which halts, does so by time λ, the latter defined as the supremum of the writable ordinals;*
(ii) any computation $P_e(n)$ with eventually stable output tape, will stabilize by time ζ the supremum of the eventually writable ordinals;
(iii) moreover ζ is the least ordinal so that there exists $\Sigma > \zeta$ with the property that

$$L_\zeta \prec_{\Sigma_2} L_\Sigma;$$

(iv) then λ is the least ordinal satisfying:

$$L_\lambda \prec_{\Sigma_1} L_\zeta.$$

We thus have a clear picture of the action of ITTM computations on integers. The machines run using very constructive rules, even for the limit stages, so their action is of course absolute to Gödel's constructible universe L. As [Hamkins and Lewis (2000)] had noted, if an ITTM machine has its hands on a real y coding an ordinal α then there is a standard turing machine program for using that code to run a construction of the L-hierarchy 'along' that ordering y, thereby producing a real code for the α'th level L_α. Hence the tie up with L is natural. A further observation on the λ, ζ, Σ-Theorem is in order. The machine limit rules of *liminf* can be expressed in a Σ_2 way. If one has two levels of the L hierarchy satisfying $L_{\zeta'} \prec_{\Sigma_2} L_{\Sigma'}$ then running the universal machine inside L it is pretty much immediate that the machine's snapshots at time ζ' and Σ' will be identical: this is what the elementarity entails. The machine will then either have halted, or, as one can show, has entered an eternally repeating loop (although the elementarity assumed is suggestive of this, in fact the latter still has to be shown). It turns out the the pair (ζ, Σ) is the lexicographically least pair of ordinals where the universal machine has identical snapshots, and first enters an infinite loop.

What further seems to emerge from the proofs above, is that the primary notion here is not that of a 'halted computation' but of a 'stable

computation': there are computations of the form $P_e(n)$ which do not formally halt, but eventually have a settled output tape, and thereafter just footle around for ever on their scratch tape areas. Halting is just a special case of *stabilizing*, and this is borne out by the fact that we cannot fully analyse halting computations without analyzing stabilizing ones. Halting can be expressed by a Σ_1 statement in set theory ("There exists a real y that successfully codes the course of computation of $P_e(n)$ with a last halting state"); this is at the basis of the Σ_1 characterisation of λ in the λ, ζ, Σ theorem as $L_\lambda \prec_{\Sigma_1} L_\zeta$, *once* we have discovered ζ. We may further establish theorems corresponding to those for halting computations.

The reader may have noticed that we seem to be avoiding discussion of the obvious fact that ITTM's can work on infinite input as well have infinite output: such computation is thus on one type up, on that of sets of integers, or reals themselves, rather than merely on integers. Before we turn to this we emphasise that Hamkins and Kidder's original formulation of an ITTM immediately visualised such capabilities: their machine was devised as coming equipped with three infinite tapes, for input, scratch and output. A single read/write head surveyed a single cell from each of the three tapes simultaneously and according to its state and program, would write from an alphabet set of $\{0, 1\}$. At limit stages a cell C_i's value was determined by taking the *limsup* of the previous cell values (there was no Blank character); the R/W head at limit stages would be brought back to the very first triplet of cells on the tapes, and the machine would enter a special 'limit state' q_L. The differences between this arrangement and that sketched above play no role in determining the classes of functions or sets computed (either on integers or on reals which we are coming to): they are the same for either model. There are minor differences in calculating halting times, and in precisely which classes of ordinals are clockable - often by an obvious factor of ω or so, but apart from these finer details there are functionally no differences between the models proposed. (This discussion does conceal one remark, that in fact, a one tape machine with two symbols cannot produce the same class of computable functions $f : \mathbb{R} \longrightarrow \mathbb{R}$. However for functions of type $f : \mathbb{R} \longrightarrow \mathbb{N}$ or $f : \mathbb{N} \longrightarrow \mathbb{N}$ a one tape 2-alphabet machine turns out to be sufficient. For the wider class curiously a third character - which we have introduced here by the way of the Blank above - turns out to be necessary. See [Hamkins and Seabold (2001)] for a discussion of this somewhat technical point and proofs of these results mentioned.)

1.3. Computation on Reals

Kleene developed an equational calculus for developing the notion of *recursion on a higher type object* (see [Kleene (1978)], [Kleene (1959)], [Kleene (1963)]). The relevant type here is *Type 2*, the objects under consideration are functionals $\mathcal{I} : \mathbb{N}^{\mathbb{N}} \longrightarrow \mathbb{N}$. This generalised his earlier equational calculus for (standard) recursive functions that used (characteristic functions for Type 1) oracles, $I : \mathbb{N} \longrightarrow \mathbb{N}$. The intuitive notion of a functional $\mathcal{F}$ being computable relative to $\mathcal{I}$ is that we have some kind of machine that can take inputs in the form of (finite sequences) of integers and reals, $\vec{n}$, $\vec{x}$, and which is connected to some oracle/memory device that has access to the graph of $\mathcal{I}$ - itself a set of size the continuum. As the domain of $\mathcal{I}$ is $\mathbb{N}^{\mathbb{N}}$, the machine must compute a real x to present to the oracle, which will return $\mathcal{I}(x)$. Thus an infinite amount of computation must be performed in some scratch/storage area before this oracle query can be launched. A *computation* will thus in general be of infinite length, but is perhaps better thought of as given by an infinite *tree* where, for example, there may be *infinite branching* nodes: below the call for $\mathcal{I}(x)$ will be the prior individual computations for $x(0), x(1), \ldots$. An illfounded tree, that is one with an infinite descending path, represents an undefined computation. There is some discussion of this in Rogers ([Rogers (1967)], p. 406, where there is no oracle $\mathcal{I}$ discussed) but where the allusion is to an "$\aleph_0$-mind' capable of forming such *generalised machine* computations. A crucial point is that a generalised computation step only be allowed to take previously, inductively, defined generalised steps. The resulting notion is '*hyperarithmetic computability*'. (See also here [Kleene (1962b)],[Kleene (1962a)].)

With the addition of the oracle $\mathcal{I}$ it can thus be loosely characterised ([Hrbacek and Simpson (1980)]) as a model of computation in which the computational device has a

(i) *countably infinite memory,* and

(ii) *an ability to manipulate (search through, write to) that memory in finite time;* and optionally

(iii) *an ability to quiz an oracle $\mathcal{I}$ about that memory contents (in a single step).*

If the above is all done within the e'th program we call the above computation $\{e\}(\vec{n}, \vec{x}, \mathcal{I})$ which again, may or may not halt. The following functional is essential for developing much of the regular theory of relative recursiveness.

$$\mathcal{E}(x) = \{ \begin{array}{l} 0 \text{ if } \exists n x(n) = 0 \\ 1 \text{ otherwise} \end{array}$$

The immediate import of this is that computation relative to the object $\mathcal{E}$ is closed under existential number quantification (for any $\mathcal{I}$ the class of relations semi-recursive in $\mathcal{I}$ is closed under universal number quantification). A second effect is that:

• If A is an arithmetical set of reals then A is recursive in $\mathcal{E}$.

More important consequences follow: if $\mathcal{I}$ is any functional such that $\mathcal{E}$ is recursive in $\mathcal{I}$, then we have the full *Ordinal Comparison Theorem* for stages of computation (see [Moschovakis (1974)]) which is crucial for developing the theory of relations semi-recursive in a type-2 functional. By 'relation' in the next theorem, we mean any predicate $R(\vec{n}, \vec{x}) \subseteq \mathbb{N}^k \times (\mathbb{N}^{\mathbb{N}})^l$ for $k, l \in \mathbb{N}$.

Theorem 1.4. *(Kleene) The hyperarithmetic relations are precisely those recursive in $\mathcal{E}$.*

The Π^1_1 relations are precisely those semi-recursive in $\mathcal{E}$.

If we are considering relative recursion of a set of reals $A \subseteq \mathbb{R}$ in a set of reals B (which we may identify with its characteristic function oracle $\mathcal{I}_B$) we may denote such:

$x \in A \simeq \{e\}(x, B, \mathcal{E}) \downarrow 1$

and say that 'A is recursive in B' if $\{e\}$ gives a function total on inputs x, and then one has appropriate versions of the above theorem relativised to B. There is an appropriate notion of *Kleene Degree*:

Definition 1.7. Kleene degrees: Let $A, B \subseteq \mathbb{R}$; we say that

$A \leq_K B$ iff there is an index e and a real y so that

for any $x \in \mathbb{R}$ $(x \overset{\in}{\underset{\notin}{}} A \longleftrightarrow \{e\}(x, y, B, \mathcal{E}) \downarrow \overset{1}{0})$;

A is *Kleene-semi-recursive in* B iff there is an index e and a real y so that

for any $x \in \mathbb{R}$ $(x \in A \longleftrightarrow \{e\}(x, y, B, \mathcal{E}) \downarrow 1)$.

The presence of the fixed real y ensures that the degree class of B contains continuum many sets of reals A; moreover the degree of B, being thus closed under continuous pre-images, forms a so-called *Wadge degree*. In general a computation evolves its own tree structure as it grows, according to its instruction set. But one can think of y as also contributing to some part of the computational tree structure. In this case, as y is allowed to

vary, we see that 0_K contains $\varnothing, \mathbb{R}$, and in fact consists of the *Borel sets*. $0'_K$ (the K-degree of a complete Kleene semi-recursive set of reals) contains WO, the set of reals coding wellorders, and so a complete Π^1_1 set of reals. In fact it consists of all the co-analytic, so precisely the $\mathbf{\Pi^1_1}$, sets.

It is possible to give a set theoretical description of Kleene recursion in a relation B and $\mathcal{E}$. In what follows, $\omega_{1\,\mathrm{ck}}^{B,y,x}$ denotes the least ordinal which does not have a real code recursive in (B, x, y); it turns out that the wellfounded computation tree of a converging Kleene recusion will have rank less than $\omega_{1\,\mathrm{ck}}^{B,y,x}$. This is the basis of the following characterisation: we only need to look inside a model with enough ordinals - namely $\omega_{1\,\mathrm{ck}}^{B,y,x}$ - to see whether the computation tree is wellfounded. Moreover, in admissibility theory wellfoundedness of any relation inside a transitive admissible set is actually a Σ_1-notion. Here $\mathcal{L}_{\in,\dot{X}}$ is the language of set theory augmented by a predicate symbol $\dot{X}$ - to be interpreted by B.

Lemma 1.1. *$A \leq_K B$ iff there are Σ_1-formulae in $\mathcal{L}_{\in,\dot{X}}$ $\varphi_1(\dot{X}, v_0, v_1)$, $\varphi_2(\dot{X}, v_0, v_1)$, and there is $y \in \mathbb{R}$, so that for any $x \in \mathbb{R}$*

$$x \in A \Longleftrightarrow L_{\omega_1^{B,y,x}}[B, y, x] \models \varphi_1[B, y, x] \Longleftrightarrow L_{\omega_1^{B,y,x}}[B, y, x] \models \neg\varphi_2[B, y, x].$$

Thus to determine whether $x \in A / x \notin A$ we perform Σ_1-searches through the least admissible set $L_{\omega_1^{B,y,x}}[B, y, x]$ relative to B containing y, x. As intimated equivalence with the former definition comes about through the original (relativised) theorem of Kleene 1.4 and the theory of admissible sets (*cf.* [Barwise (1975)]).

The generalised theory of recursion in higher types was much investigated and developed in the late 60's and 70's, with a history too rich to go into here, with names such as Gandy, Moschovakis, Sacks, Grilliot, Fenstad, Normann, Moldestad, Harrington prominent. The recursion relative to the single operator $\mathcal{E}$ is in one respect merely illustrative, being the historical example from the earlier days and papers ([Kleene (1959)], [Kleene (1963)], [Kleene (1978)]) of the Kleene Equational Calculus. The reader may consult Hinman's [Hinman (1978)] for an overall development of the theory, Fenstad [Fenstad (1980)] for an attempt to present an axiomatic approach to general computation theories, and the latter Part D of Sacks [Sacks (1990)] for the further development in relation to set recursion.

Mention must now be made of the connections to the theory of *inductive definitions* and here more particularly to the theory of *Spector classes*. The latter is a general unifying theory of definability developed by Moschovakis in [Moschovakis (1974)]. We consider here just pointclasses $\Gamma \subseteq \mathbb{N}^k \times (\mathbb{N}^{\mathbb{N}})^l$

(for any $k, l < \omega$) and use the notation that $\breve{\Gamma} = \{\neg R : R \in \Gamma\}$. $\exists^{\mathbb{N}}$, $\forall^{\mathbb{N}}$ represent natural number quantifiers as opposed to $\exists^{\mathbb{N}^{\mathbb{N}}}$, $\forall^{\mathbb{N}^{\mathbb{N}}}$ over elements of $\mathbb{N}^{\mathbb{N}}$.

Definition 1.8. A *Spector class* of pointsets $\Gamma \subseteq \mathbb{N}^k \times (\mathbb{N}^{\mathbb{N}})^l$ for any k, l, is a collection that is (i) closed under $\cap, \cup$, number quantification: $\exists^{\mathbb{N}}$, $\forall^{\mathbb{N}}$; closed under (standard) recursive substitutions, has a universal set U indexing by $\mathbb{N}$ all members of Γ, and lastly has the Prewellordering property:

PW: For any $P \in \Gamma$ there is $\sigma : P \longrightarrow \lambda$ for some ordinal λ with the property that there are relations: $x \leq_0^\sigma \in \Gamma$, $x \leq_1^\sigma y \in \breve{\Gamma}$ so that:

$$P(y) \Rightarrow (\forall x[P(x) \wedge \sigma(x) \leq \sigma(y)] \Longleftrightarrow x \leq_0^\sigma y \Longleftrightarrow x \leq_1^\sigma y).$$

It would be impossible to give a full exposition of the import of Spector pointclasses here, but suffice it to say that the definition above encapsulates a fundamental unifying approach to the theory of inductive definability. Familiar Spector pointclasses are Π^1_1 and Σ^1_2 but there are many others. For Σ^1_n or Π^1_n the existence of the prewellordering property depends on the surrounding set theory in which one works. We shall only be discussing Spector pointclasses within $\Delta^1_2 = \Pi^1_2 \cap \Sigma^1_2$. The Kleene recursion theory then throws up a canonical example of a Spector pointclass: the Kleene semi-recursive (in $\mathcal{E}$) sets are precisely the Π^1_1 sets.

The type of formalism on the right hand side equivalences of Lemma 1.1 in fact is also one way of defining Spector classes within the Δ^1_2 pointclass.

1.3.1. *ITTM computations on reals*

If we now return to the ITTM model we shall see that it fits very nicely into this overall general theory. We have a choice to make here. At **Q3** above we called the *decidable* predicates, those where a characteristic function of the predicate could always be computed by a *halting* computation. It is natural particularly given the machine nature of the origins of the notion to think of *halting* as somehow fundamental, and therefore it is this that should be used to characterise 'decidability'. However here we are adopting the position that the fundamental feature of the ITTM's is the Σ_2 nature of the limit rule for the cell values, and the concomitant phenomenon of their having stabilized output without halting; it was indeed from this stabilizing and looping times, from ζ and Σ, that we could characterize the halting times. The halting computations are for this purpose to be regarded as the

special sub-class of 'fully stabilized' computations: halting is just a special kind of stabilization (sic). This position is further strengthened when we consider below its relation to previous notions of higher type recursion.

We stated the Normal Form Theorems in the stronger, halting, version, as these would be more familiar to the reader, but there are equally well Normal Form Theorems which are verbatim as above but with $\downarrow$ replaced by $\uparrow$ throughout. The viewpoint here is that the strongly stabilizing, *i.e.* halting, computations should probably be thought to give rise to a notion of *strong decidability* (and *strong semi-decidability*) whilst the stabilizing computations correspond to the notion of *decidability* and *semi-decidability*. However most papers distinguish the 'stabilizing' form, with the adverb 'eventually', used in the form: 'eventually (semi)-decidable predicates' or adjectively as in 'eventual ITTM degrees' etc. This is established enough that it would egregious to go against it here.

However for notions from higher type recursion theory, one says in general that a class of 'semi-decidable sets are those semi-recursive in a $\mathcal{F}$' where the latter $\mathcal{F}$ is some higher type functional. *Then*, for the appropriate $\mathcal{F}$ for ITTM's, actually '*semi-recursive in* $\mathcal{F}$' would correspond to the stabilizing behaviour rather than the halting one. This would also accord with the usage inherited from Kleene Recursion. We shall call here then "ITTM-semi-recursive" those predicates where membership facts can be represented as the stable output of some program, and thus corresponding to 'eventually semi-decidable' in the literature. (For different classes of machines such as the Σ_n-machines mentioned below, the notion of 'output' becomes somewhat more rarified, but these too one we would like to think of as providing mathematical classes of sets that are generalised (semi-)recursive in some way.)

It is a conceptually simple adjustment to have within an ITTM program an oracle call that requests of some oracle $B \subseteq \mathbb{R}$ (here $2^{\mathbb{N}}$) whether the current contents of the scratch tape, $y \in 2^{\mathbb{N}}$, is an element of B, and receive a 0/1 reply. Thus computation relative to an oracle for sets of reals is unproblematic. We again adopt the same notation that $P_e^B(x) \downarrow y$ if the e'th machine with oracle B, on input $x \in 2^{\mathbb{N}}$ halts with output $y \in 2^{\mathbb{N}}$. Changing the arrow to $P_e^B(x) \uparrow y$ indicates that *eventually* y is written to the output tape, and remains there unchanging from some point on. (We have to have some other notation such as "$P_e^B(x)$ |" for the computation diverges or is undefined.)

We first give the integer version.

Definition 1.9. A set of integers x is *ITTM-semi-recursive* in a set y if and only if:

$$\exists e \forall n \in x \, [P_e^y(n) \uparrow 1 \longleftrightarrow n \in x \ ]$$

(ii) A set of integers x is *ITTM-recursive* in a set y if and only if:

$$\exists e \forall n \in x \, [P_e^y(n) \uparrow 1 \leftrightarrow n \in \ x \wedge P_e^y(n) \uparrow 0 \leftrightarrow n \notin \ x] \, .$$

We may write $x \preceq^\infty y$ for the reducibility ordering.

Equivalently: x is ITTM-recursive in y if both x and $\neg x$ are ITTM-semi-recursive in y (since if the latter holds it is easy to amalgamate the two programs into a single program P_e with the effect of the Definition. The relation $\preceq^\infty$ is in the class Δ^1_2. There is a natural prewellordering that arises on computations P_e establishing membership in some set x: put $n \prec m$ if the computation $P_e(n) \uparrow 1$ stabilizes to an output of 1 before that of $P_e(m) \uparrow 1$ does. The relation $\prec$ is itself ITTM-semi-recursive (think of the universal machine that observes the simulated copies of computation sequences of P_e for various n - eventually it itself will stabilize into seeing that $P_e(n)$ stabilizes before $P_e(m)$) and thus we can establish the prewellordering property very easily.

There is a natural notion of *complete ITTM-semi-recursive set of integers*:

Definition 1.10. $\tilde{x} =_{\mathrm{df}} \{e \mid P_e(0) \uparrow\}$ - the complete set of *stable indices.*

The following tells us what this set is by way of a set theoretic characterisation. We regard $x \rightarrowtail \tilde{x}$ as an analogy to the hyperjump operation.

Theorem 1.5. ([Welch (2000)]) *$\tilde{x}$ is (Turing-)recursively isomorphic to the Σ_2-theory of $\langle L_{\zeta^x}[x], \in, x\rangle$. In particular $\tilde{0}$ is recursively isomorphic to the Σ_2-theory of $\langle L_\zeta, \in\rangle$.*

This should be compared with Kleene's result that his notation system set $\mathcal{O}$ - a complete Π^1_1 set of integers coding indices of wellfounded finite path trees - is in fact (Turing-) recursively isomorphic to the Σ_1-truth set of $\langle L_{\omega_1^{\mathrm{ck}}}, \in\rangle$. Indeed A. Klev has defined in [Klev (2007)] 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). By the above, it is thus to the complete $\Sigma_2(L_\zeta)$ set what $\mathcal{O}$ is to $\Sigma_1(L_{\omega_1^{\mathrm{ck}}})$.

The following is the natural version for real computation.

Definition 1.11. A set of reals A is *ITTM-semi-recursive* in a set of reals B if and only if:

$$\exists e \forall x \in 2^{\mathbb{N}} \left[P_e^B(x) \uparrow 1 \leftrightarrow x \in A \right]$$

(ii) A set of reals A is *ITTM-recursive* in a set of reals B if and only if:

$$\exists e \forall x \in 2^{\mathbb{N}} \left[P_e^B(x) \uparrow 1 \leftrightarrow x \in A \wedge P_e^B(x) \uparrow 0 \leftrightarrow x \notin A \right]$$

Definition 1.12. $A \leq^\infty B$ iff for some $e \in \omega$, for some $y \in \mathbb{R}$: A is ITTM-recursive in (y, B).

Notice in the above that we have included a parameter real y to ensure the closure under continuous preimages as before. This will ensure we have *Wadge pointclasses* and that the ensuing notion of $\leq^\infty$- *degree* with the degree ordering induced, will be wellfounded. The structure of this degree ordering is dependent on the ambient set theory - we shall not go into this now, but under the assumption of "sufficient Determinacy" (that of two person perfect information games of sufficient complexity in their payoff sets) we shall have that the degrees are wellordered; under the assumption of $V = L$ the ordering of $\leq^\infty$ degrees is very different, and below the complete $\leq^\infty$-semi-recursive set of reals there are plenty of $\leq^\infty$-incomparable sets (and hence Post's problem has a rich positive solution); whilst under "sufficient determinacy" assumptions, there are no intermediate degrees at all. This was to be expected, and serves only to confirm the position of the pointclass of ITTM semi-recursive sets as one within the totality of the Wadge ordering of all reasonable pointclasses of sets of reals. See [Welch (2004)] for a further discussion and results.

By analogy with Kleene recursion we have:

Lemma 1.2. *$A \leq^\infty B$ iff there are Σ_2-formulae in $\mathcal{L}_{\in,\dot{X}}$ $\varphi_1(\dot{X}, v_0, v_1)$, $\varphi_2(\dot{X}, v_0, v_1)$, and $y \in \mathbb{R}$, so that for all $x \in \mathbb{R}$*

$$x \in A \Longleftrightarrow L_{\zeta^{B,y,x}}[B, y, x] \models \varphi_1[B, y, x] \Longleftrightarrow L_{\zeta^{B,y,x}}[B, y, x] \models \neg\varphi_2[B, y, x].$$

The Lemma then identifies structures in which we can look to ascertain the outcomes of our ITTM computations relative to a set of reals B say. By way of analogy with ζ, the ordinal $\zeta^{B,y,x}$ is the least that is not ITTM-(B, x, y)-recursive. It is thus also least such that $L_{\zeta^{B,y,x}}[B, y, x]$ has a proper Σ_2-elementary end-extension.

Just as one has a Uniformisation Theorem for Kleene-semi-recursive sets (namely the Novikoff-Kondo-Addison Π^1_1-Uniformisation Theorem) that

such sets in the plane can be uniformised by Kleene semi-recursive functions, so we have:

Lemma 1.3. (Uniformisation Theorem) *Suppose $A \subseteq \mathbb{R} \times \mathbb{R}$ is ITTM-semi-recursive. Then there is a (partial) function $F : \mathbb{R} \to \mathbb{R}$ with ITTM-semi-recursive graph with the property that:*

$$\forall x \in \mathbb{R}[\exists y \in \mathbb{R}(\langle x, y\rangle \in A) \Rightarrow (x \in dom(F) \wedge \langle x, F(x)\rangle \in A)].$$

1.4. Computation on ordinals, and ordinal length machines

In the 70's the theory of *α-recursion* coming out of the *meta-recursion* of the 60's reached its highest stage of development. The observation that an enumeration of a Π^1_1-complete set of integers was very naturally effected, not in ω, but in ω_1^{ck} (the least non-recursive ordinal) steps led to a discussion on the role of hyperarithmetic *vis à vis* finite. In meta-recursion the motivation was to have a generalization of recursion theory where infinitely long computations converged. Initially the emphasis had been on using an analogy between finite/recursive/recursively enumerable to yield a notion of meta-finite/metarecursive/meta-r.e. In the latter the integers would be replaced by recursive ordinals, and a meta-r.e. set was a set of recursive ordinals whose *indices* formed a Π^1_1 set. Meta-recursive sets would be those that were both meta-r.e. and co-meta-r.e. The notion that replaced finiteness, was that of meta-finiteness which was to be identified with a set of ordinals together with a hyperarithmetic index set. In particular the domain of computation had now changed: instead of ω it would become ω_1^{ck}. (See, *e.g.* , the discussion in [Sacks (1990)] Part V for an account of this development.) This was not the first generalisation of recursion theory to ordinals: Takeuti [Takeuti (1960)] had replaced 'recursive enumerability' by a scheme equivalent to Σ_1-definability and was the first to generalise recursion theory from natural numbers to ordinals. There were a number of developments from Kleene's equational calculus to include ordinal valued functions in equations: Machover [Machover (1961)], Levy [Levy (1963)], Tugué [Tugué (1964)], Kripke [Kripke (1964)], Platek [Platek (1966)] all had such calculi. The latter two involved what emerged as a primary notion, that of an *admissible ordinal* with the concomitant axiomatisation of *admissible set theory* as a fragment of full ZFC set theory. Platek had the notion of an *admissible set*. From one perspective, it seems pointless to split the distinction between an 'equational calculus' and an abstract 'machine' (if there is one to split). Platek though seems to have had in mind,

or at least the picture of, an ordinal register machine of some sorts, which we shall turn to these later.

1.4.1. *Ordinal length tapes*

Since the Hamkins-Kidder machines can construct levels of the Gödel L-hierarchy up to a certain stage (below the level Σ alluded to above) it is a natural generalisation to think up behaviours for machines with tape not an ω sequence of cells, but longer. Indeed why not consider a sequence of cells C_α for α any ordinal? Both Koepke and Dawson independently, and at roughly around the same time, came up with the idea of ordinal length tape machines, equipped with *liminf* rules to locate read/write heads and instruction numbers within a program list. One allows the head to move left, but again must specify if the the head is over a limit cell C_λ what default action to do if the machine is asked to move left one cell. As sets can be coded by sets of ordinals (assuming the Axiom of Choice) we have some means of dealing with, or representing sets on tapes. If a machine runs and produces a sequence of 0's and 1's on a tape, then again, if of the right form, we can say that the machine is *producing* (codes for) sets. Dawson [Dawson (2009)] formulated an *Axiom of Computability* that states that every set is computable, in that there is a program that produces (not necessarily halting) at some point a code for that set. He then proves that the computable sets form a transitive class satisfying the ZF axioms together with AC. A condensation lemma on the elements appearing in a table of a long computation then produces the Generalised Continuum Hypothesis. As the construction of the machine and its action is completely absolute in character, we can imagine the machine running inside the constructible universe L, performing the same actions with the same outcomes. Since L is the minimal transitive class model of ZF, then of course the machine is producing precisely the constructible sets.

Koepke gave a detailed description in [Koepke (2005)], [Koepke and Koerwien (2006)] of the organisation of such results, and whereas Dawson was considering codes for sets running on an everlasting machine, Koepke considers *halting computations* starting from an input tape with marks for finitely many ordinals. Koepke then shows in detail that a *Bounded Truth function* for L is computable. He then has:

Theorem 1.6. (Koepke [Koepke (2005)]) *A set x is computable from a finite set of ordinal parameters if and only if it is a member of the constructible hierarchy.*

He then proceeds to derive GCH again using this analysis. During the 1970's Silver produced a description of the constructible hierarchy using, what came to be called 'Silver Machines'. Silver's motivation was to avoid R. Jensen's 'fine-structural' description of L, which Jensen had used to great effect in establishing fundamental properties both of L, and of the universe of all sets. An account of Silver's method is in [Devlin (1984)], Part IX. The 'machine' nature of the description is essentially that of an extremely slowed production of constructible sets, and owes more to a desire to have as simple as possible method of set construction, rather than a perspective with a mechanical flavour. Silver convincingly made use of his theory by producing a fine-structure free proof of an important combinatorial principle of L called $\square$, due to Jensen. The ordinal length tape Turing machine model held out hope that another different proof of $\square$ might be possible using the machine's description. That hope has not been realised, and it seems that despite the smoothness of set construction at successor steps, the infinitary nature of the limit rule mitigates against certain construction principles that seem common to most proofs of $\square$ to date, so maybe this appoach would seem difficult.

Nevertheless, the description of the constructible sets, now adds a further method of describing L besides the two originally due to Gödel, and to those of Jensen and Silver.

1.4.1.1. *α-length tapes*

Rather than take ON length tapes, it would be possible to consider computation using the above machines but with the length of tape, and perhaps time, restricted to say suitable ordinals α, such as initial ordinals or cardinal numbers. There would indeed be nothing against this: one could produce, say just the hereditarily countable members of L by allowing only computations that took countable lengths of time. For restricting to computations not of cardinal length, some closure considerations come into effect. In order to have effective methods of combining even very elementary processes on sets, one should require that ordinals be sufficiently closed to enable this, and something such as closure under the primitive recursive set functions (*cf.* [Devlin (1984)], p.100) would be suitable.

The notion of *admissible ordinal* stands out, not least because of the development of *α-recursion theory* in the 1960's and 70's. We have briefly mentioned the origins of this theory at the beginning of this section. The motivation for its development was indeed a theoretical one: to lift from

$\mathbb{N}$ the theory of recursion to other domains. The closure of an admissible ordinal was soon seen to result in a powerful theory of sets that when axiomatised gives essentially a reduced form of ZF, with the scheme of Replacement restricted to Σ_1 instances, and that of Comprehension to Δ_1. An *admissible set* was then a model of this theory, and $\langle L_{\omega_1^{\mathrm{ck}}}, \in\rangle$ is the least transitive model of this theory (if one includes the axiom of infinity). An account of this development is contained in [Sacks (1990)].

One could therefore simply restrict an ordinal length tape machine to an admissible ordinal length α, and consider calculations of length at most α in time.

Does one get back precisely the theory of α-recursion theory? Does "computably enumerable" correspond to α-r.e., and if so does the machine approach give any new slant on the old results from the 70's such as the Sacks-Simpson theorem [Sacks and Simpson (1972)] that there are incomparable α-r.e. sets neither (weakly) α-recursive in the other; or the Shore Splitting and Density theorems [Shore (1975)], [Shore (1976)]? These are matters still under investigation. Dawson ([Dawson (2009)]) has established for a notion of what he calls *uniform α-computation* that indeed one has the Sacks-Simpson and Shore Density results.

1.4.2. *Ordinal Register Machines*

We now turn to full blooded finite register machines with the capability of ordinal entries. Again such machines are allowed to run transfinitely using an ordinary register arrangement, and finite instruction set, with a suitable *liminf* rules for *register values* and *instruction numbers* at limit ordinal λ lengths of time. We have mentioned that one (unpublished by Platek) approach yielded an equational calculus for ordinal recursion up to ω_1^{ck}, Siders and Koepke [Koepke and Siders (2006)], consider register machines with a stack, and remarkably even a machine with finitely many registers allows one to calculate a bounded truth predicate for L. One thus can represent L both using Ordinal Register Machines (ORM) and Ordinal Time Turing Machines (OTM).

As for the ITTM's one has notions of *clockable ordinal* (one for which an ORM or OTM halts on say 0 input) and *writable ordinal* (one for which a code can be written: this is easier to formulate for an OTM: a code can be written literally on the tape; for an ORM one simply has the machine halt with the ordinal in, say the first register). For both these notions it is easier than for ITTM's to conclude that γ the supremum of the clockable

ordinals is that of the writable ordinals. In [Hamkins and Miller (2009)] it is explicitly shown how to convert calculations from an ORM to an OTM and vice versa.

Using ordinal register machines with values up to the admissible ordinal γ Hamkins and Miller have used priority arguments to produce a Friedberg-Muchnik like solution to Post's problem [Hamkins and Miller (2009)] for ORM's: they produce ORM-enumerable but incomparable sets $A, B \subset \gamma$ that are below the appropriate notion of jump.

Definition 1.13. Let P_e be the e'th ORM program, the (weak) jump is the set

$$0^{\diamondsuit} = \{e \in \mathbb{N} \mid P_e(0) \downarrow\}.$$

Although neither [Hamkins and Miller (2009)] nor [Koepke and Siders (2006)] make the following characterisation, it appears reasonable to argue that the ordinal γ in fact is recognisable by set theorists as the first Σ_1-*stable ordinal* σ. This is defined to be the least ordinal σ so that $\langle L_\sigma, \in\rangle \prec_{\Sigma_1} \langle V, \in \rangle$, that is, L_σ is an elementary substructure of the universe of all sets, but only for Σ_1 sentences expressible using parameters from L_σ. (See [Hinman (1978)] p. 412 for an equivalent definition in terms of ∞-partial recursive functions.) Consequently if any ORM (or OTM) halts on integer input (or indeed any input less than γ) then the length of that computation must be also less than σ, as this halting assertion is itself a simple Σ_1-statement in the language of set theory. (Moreover anything output by such a machine must also clearly be an ordinal less than σ by the same reasoning.) Hence $\gamma \leq \sigma$. To see that $\sigma \leq \gamma$ observe that in the L hierarchy, new Σ_1 sentences become true in L_δ for arbitrarily large ordinals $\delta < \gamma$. Now given a true Σ_1 sentence in the language of set theory, run an ORM (or OTM) program to search for that ordinal δ, and then halt. This task must take more than δ (but also less than σ) steps. Hence $\sigma = \gamma$. One then obtains:

Proposition 1.5. $0^{\diamondsuit}$ *is recursively isomorphic to the* Σ_1*-truth set of* $\langle L_\sigma, \in \rangle$.

One can compare this with the statement that the standard Turing halting set is recursively isomorphic to the Σ_1-truth set of $\langle L_\omega, \in\rangle$ where $L_\omega = \mathrm{HF}$ the class of hereditarily finite sets. A similar result holds (with the appropriate formulations) for OTM's for the same reasons.

1.5. Theoretical machine strength

We consider finally the *theoretical strengths* of the various types of mechanisms considered here. We have answered in one fashion at least, the capabilities of the machines in the Malament-Hogarth spacetimes, and the Etesi-Németi model in particular. It is also clear that the ON-length tape machines are full ZFC-machines that are capable of producing Gödel's constructible universe.

For the intermediate machine models we have mentioned, one could simply be satisfied by seeing at which level of complexity the machines can answer queries concerning predicates. One can however somewhat more formally, formulate a theory in which the behaviour of the machine can be represented, and one may then calibrate this theory, not necessarily proof theoretically, but at least as a theory within other theories, for example as a subsystem of second order analysis, much as is done in the *Reverse Mathematics Program* (see [Simpson (1999)]). The discussion becomes somewhat technical, but for the logician, interesting.

Towards analysing the ITTM's we first look at connections to certain kinds of *quasi-inductive definition* that were defined earlier, at least in one form, by Burgess in [Burgess (1986)].

Let $\Gamma : \mathcal{P}(\omega) \to \mathcal{P}(\omega)$ be any arithmetic operator (that is "$n \in \Gamma(X)$" is arithmetic; we emphasise that Γ need be neither monotone nor progressive). We define the following iterates of Γ : $\Gamma_0(X) = X; \Gamma_{\alpha+1}(X) = \Gamma(\Gamma_\alpha(X)); \Gamma_\lambda(X) = \liminf_{\alpha\to\lambda} \Gamma_\alpha(X) = \cup_{\alpha<\lambda} \cap_{\lambda>\beta>\alpha} \Gamma_\beta(X)$. Following [Burgess (1986)], we say that $Y \subseteq \omega$ is *arithmetically quasi-inductive* if for some such Γ, Y is (1-1) reducible to$\Gamma_{\mathrm{On}}(\varnothing)$. Any such definition has a least countable $\xi = \xi(\Gamma)$ with $\Gamma_\xi(\varnothing) = \Gamma_{\mathrm{On}}(\varnothing)$. If we let ζ denote the supremum of all such $\xi(\Gamma)$, then we have that the ζ defined here is none other than the ζ defined above relating to ITTM's. In fact the ITTM's give an example of a *recursive quasi-inductive operator* that is *complete* for all *arithmetic quasi-inductive operators*. (Think: a universal ITTM can be programmed to mimick any *arithmetic* quasi-inductive operator.) Hence the same class of sets arises, it turns out, if one restricts to simply recursive Γ.

For any such arithmetic quasi-inductive operator Γ let us now define the *repeat pair* of Γ on a starting set X, as the lexicographically least pair $(\zeta(\Gamma, X), \Sigma(\Gamma, X))$ with $\Gamma_\zeta(X) = \Gamma_\Sigma(X)$.

Definition 1.14. AQI is the sentence: *"For every arithmetic operator* Γ, *for every* $X \subseteq \mathbb{N}$, *there is a wellordering* W *with a repeat pair*

$(\zeta(\Gamma, X), \Sigma(\Gamma, X))$ *in* Field(W)*"*. If an arithmetic operator Γ acting on X has a repeat pair, we say that Γ *converges* (with input X).

Then AQI can be formulated in second order number theory, and essentially is asserting that there are sufficient wellorderings for every operator on every input set X to converge. One may ask

Q: *What is the strength of* $\mathsf{ACA}_0 + \mathsf{AQI}$?

(The choice of ACA_0, *arithmetical comprehension*, as a base theory is somewhat arbitrary. We refer the reader to [Simpson (1999)] in what follows for all notions concerning these axiom systems, and determinacy hypotheses etc.) We could have equivalently reformulated a version of AQI which mentioned instead looping points of ITTM's, but this would turn out to be equivalent, as we have intimated. $\mathsf{\Pi^1_3CA}_0$ is sufficient to prove there are β-models of $\mathsf{ACA}_0 + \mathsf{AQI}$.

Theorem 1.7. ([Welch (2005)])

(i) $\mathsf{\Pi^1_3CA}_0$, $\mathsf{ACA}_0 + \mathsf{AQI}$ *and* $\mathsf{\Pi^1_2CA}_0$ *are in descending order of strength in that each theory proves the the existence of* β*-models of the next.*

More precisely, and in the same sense:

(ii) $\mathsf{\Delta^1_3CA}_0$+$\mathbf{\Sigma^0_3}$-Determinacy, $\mathsf{ACA}_0 + \mathsf{AQI}$, and $\mathsf{\Delta^1_3CA}_0$ *are similarly in strictly descending order of strength.*

Determinacy makes an appearance here, since this theorem is the outcome of an attempt to generalise the theorem of Solovay (see [Kechris (1978)]) that strategies for Σ^0_2-games appear at the level σ^1_1 of the L hierarchy, the closure ordinal for Σ^1_1-monotone inductive definitions. (In turn, as is well known, stategies for Σ^0_1-games appear at $\omega_{1\,\mathrm{ck}}$ the closure ordinal for Π^1_1-monotone inductive definitions. We are thus trying to link AQI's, or ITTM's to strategies for certain infinite games.) Thus AQI's are close to, but not equivalent to, Σ^0_3-Determinacy. That they are stronger than Σ^0_2-Determinacy, is because $\sigma^1_1 < \zeta$. Moreover, letting "Bool(Σ^0_2)" denote Boolean combinations of Σ^0_2 sets, the constructible rank of the height of the least β-model of $\mathsf{\Pi^1_2CA}_0$ (as shown by Möllerfeld and Heinatsch [Heinatsch and Möllerfeld (2007)]) where strategies for Bool(Σ^0_2) games are to be found, is less than ζ, and in fact is again a "writable" ordinal less than λ, in the sense of ITTM's. This shows that the assertion that any ITTM halts or loops, is stronger than Bool(Σ^0_2)-Determinacy.

That $\mathsf{\Delta^1_3CA}_0 + \mathsf{AQI}$ is stronger than $\mathsf{\Delta^1_3CA}_0$ in the above sense should be plausible in that the universal ITTM on input $\varnothing$ will go into a loop at the "repeat pair" ordinals ζ and Σ where $L_\zeta \prec_{\Sigma_2} L_\Sigma$. However it is easy to

see from this Σ_2-extendability property that any such L_ζ is a Σ_2-admissible set (where we now require the admissible set to additionally be a model of Σ_2-Replacement) and is also a union of such. The reals of such a model then form a β-model of $\mathsf{\Delta}^1_3\mathsf{CA}_0$.

Connections to ordinal analysis

The notion of "Σ_2-extendibility" of a model, that is of having a proper Σ_2-elementary end extension, would seem *prima facie*, to be connected to any attempt to prove a generalisation of Rathjen's ordinal analysis ([Rathjen (2005)]) of $\Pi^1_2\mathsf{CA}_0$ that could be lifted to $\mathsf{\Pi}^1_3\mathsf{CA}_0$. In the former proof, chains of arbitrary but finite length of Σ_1-end entensions in the constructible hierarchy of the form $L_{\xi_1} \prec_{\Sigma_1} L_{\xi_1} \prec_{\Sigma_1} \cdots \prec_{\Sigma_1} L_{\xi_n}$ are analysed. (Note that the least β-model of $\Pi^1_2\mathsf{CA}_0$ consists of $\mathcal{P}(\mathbb{N}) \cap L_{\xi_\omega}$ where ξ_ω is the least ordinal with L_{ξ_ω} a union of an infinite tower of of Σ_1 substructures.) To analyse $\mathsf{\Pi}^1_3\mathsf{CA}_0$ in a similar way would require lifting the '1' above to '2' and looking at arbitrarily long chains of Σ_2-extensions. It would seem then that any ordinal analysis of $\mathsf{\Pi}^1_3\mathsf{CA}_0$ would first have to go through an analysis of AQI, the latter being but the very first step in this linkage.

Σ_n-machines.

The notion of *liminf* is essentially a two quantifier alternation: *"there exists a time such that for all later times...."*. It is possible to enquire whether there are other types of limit rule that bring out about different notions of computation, or different classes of computable function. One attempt to consider this question is a result of [Welch (2000)] which shows that, amongst all possible Σ_2-rules the *liminf* (or equivalently the *limsup*) rule is complete, or the most general. This is entirely unsurprising: if the universal ITTM machine can produce the constructible hierarchy up to L_Σ, there is little else for it to do. Further any other Σ_2-rule would itself produce looping behaviour between L_ζ and L_Σ.

One may thus broaden the enquiry and look for more complex rules. It is possible to develop a Σ_3-machine which incorporates a Σ_3-limit rule cf.[Friedman and Welch (2009)]; the essential idea is that instead of taking a liminf along all ordinals one takes a liminf using only those ordinals that already bound the reapparances of earlier (shifted) snapshots. One thus has in some sense a *dynamic limit rule* in that the behaviour at a limit rule depends more formally on the tapes' prior contents. One then has the analogous result that a universal machine program would then have identical snapshots at the least pair $(\zeta(3), \Sigma(3))$ where $L_{\zeta(3)} \prec_{\Sigma_3} L_{\Sigma(3)}$ to mirror the earlier λ-ζ-Σ theorem at Σ_2. After $\Sigma(3)$ it then returns to the previous snapshot at $\zeta(3)$ and thereafter repeats for ever. It is possible to gener-

alise this to higher quantificational levels Σ_n with the snapshot/looping behaviour at the appropriate pair $(\zeta(n), \Sigma(n))$ lexicographically least with $L_{\zeta(n)} \prec_{\Sigma_n} L_{\Sigma(n)}$. However showing these facts is more technical, and is reliant more on the underlying set theory; it thus perhaps has decreasingly less of an appeal to intuitions concerning machine computation. This is explained in [Friedman and Welch (2009)].

ITRM's on integers

The ITRM's of Miller and Koepke (2.2.2) with entries restricted to natural numbers turn out to be pleasantly strong. It is possible show that such machines are Π^1_1-complete, in that for any Π^1_1 set A there is a program on an ITRM, that correctly accepts or rejects n depending on whether n is or is not in A. It can thus for example decide for which indices e the e'th (standard) Turing function $\{e\}$ is the characteristic function of a wellorder or not. Moreover it can be shown that the strength of the machine strictly increases with the number N of registers. It is possible with $2N$ registers to simulate an N register machine, whilst giving as output integer codes of those programs on N registers that halt. A corollary is that there can be no such *universal machine.* Here we let $P_{e,N}$ denote the e'th ITRM program for an N register machine.

Definition 1.15. Let ITRM be the axiom scheme that states for each $N \in \mathbb{N}$ that halting sets for N-register machines exists:

"For any $N \in \mathbb{N}, K_N =_{\mathrm{df}} \{e | P_{e,N}(\vec{0}) \downarrow\}$ exists".

One then obtains (with RCA_0 as the *recursive comprehension scheme*):

Theorem 1.8. $\mathsf{RCA}_0 \vdash \mathsf{ITRM} \longleftrightarrow \Pi^1_1\mathsf{CA}_0$.

Reverse Mathematics has shown that a wealth of theorems can be proven in the system $\Pi^1_1\mathsf{CA}_0$. As a sample we have the following, in which we assume the ITRM is equipped with an *oracle set* $Z \subseteq \mathbb{N}$ (and a register operation to query it):

Theorem 1.9. *Let* $T \subseteq \{\sigma \mid \sigma \in^{<\mathbb{N}} \mathbb{N}\}$ *be a set of sequences which we consider as forming a* tree. *Then if* $Z \subseteq \mathbb{N}$ *codes* T *(via some recursive coding), the* perfect kernel *of* T *is ITRM-computable in the oracle* Z.

(By the perfect kernal we mean the maximal subtree whose branches form a perfect set, that is without isolated points.) Thus, as with much of this kind of study, a seemingly simple model in fact turns out to be rather powerful.

References

Barwise, K. (1975). *Admissible Sets and Structures*, Perspectives in Mathematical Logic (Springer Verlag).

Beggs, E., Costa, J., Loff, B. and Tucker, J. (2008). Oracles and advice as measurement, in C. C. et al. (ed.), *Unconventional Computing*, *Lecture Notes in Computer Science*, Vol. 5204 (Springer), pp. 33–50.

Beggs, E. and Tucker, J. (2007). Can newtonian systems, bounded in space, time, mass and energy compute all functions? *Theoretical Computer Science* **371**, pp. 4–19.

Blum, L., Shub, M. and Smale, S. (1989). On a theory of computation and complexity over the real numbers, *Notices of the American Mathematics Society (N.S.)* **21**, 1, pp. 1–46.

Burgess, J. (1986). The truth is never simple, *Journal of Symbolic Logic* **51**, 3, pp. 663–681.

Cutland, N. (1980). *Computability: an Introduction to Recursive Function Theory* (CUP).

Davies, E. (2001). Building infinite machines, *British J. for Philosophy of Science* **52**, 4, pp. 671–682.

Dawson, B. (2009). *Ordinal time Turing computation*, Ph.D. thesis, Bristol.

Devlin, K. (1984). *Constructibility*, Perspectives in Mathematical Logic (Springer Verlag, Berlin, Heidelberg).

Earman, J. and Norton, J. (1993). Forever is a day: Supertasks in Pitowsky and Malament-Hogarth spacetimes, *Philosophy of Science* **60**, pp. 22–42.

Earman, J. and Norton, J. (1996). Infinite pains: the trouble with supertasks, in A. Morton and S. Stich (eds.), *Benacerraf and his critics*, *Philosophers and their critics*, Vol. xi (Blackwell, Oxford), p. 271.

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

Fenstad, J. (1980). *General recursion Theory: an axiomatic approach*, Perspectives in Mathematical Logic (Springer, Berlin, Heidelberg, New York).

Friedman, S. D. and Welch, P. D. (2009). Hypermachines, *submitted to the* Journal of Symbolic Logic .

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

Hamkins, J. and Miller, R. (2009). Post's problem for ordinal register machines: an explicit approach, *Annals of Pure and Applied Logic* **160**, 3, pp. 302–309.

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

Hawking, S. and Ellis, G. (1973). *The large scale structure of space-time* (Cambridge University Press).

Heinatsch, C. and Möllerfeld, M. (2007). Determinacy in second order arithmetic, in S. Bold, B.Löwe, T. Räsch and J. van Bentham (eds.), *Foundations of the Formal Sciences V*, Studies in Logic (College Publications, London), pp. 143–155.

Hinman, P. (1978). *Recursion-Theoretic Hierarchies*, Ω Series in Mathematical Logic (Springer, Berlin).

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

Hogarth, M. (1994). Non-Turing computers and non-Turing computability, *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association Vol. 1* **1994**, pp. 126–138.

Hogarth, M. (2004). Deciding arithmetic using *SAD* computers, *British Journal for the Philosophy of Science* **55**, pp. 681–691.

Hrbacek, K. and Simpson, S. (1980). On Kleene degrees of analytic sets, in H. J.Barwise and K.Kunen (eds.), *Proceedings of the Kleene Symposium*, Studies in Logic (North-Holland), pp. 347–352.

Kechris, A. S. (1978). On Spector classes, in A. S. Kechris and Y. N. Moschovakis (eds.), *Cabal Seminar 76-77*, *Lecture Notes in Mathematics Series*, Vol. 689 (Springer), pp. 245–278.

Kleene, S. (1978). Recursive functionals and quantifiers of finite types revisited, in *Generalized Recursion Theory II, Proceedings 2nd Scandinavian Logic Symposium, Oslo, 1977*, *Studies in Logic and Foundations of Mathematics*, Vol. 94 (North-Holland, Amsterdam, New York), pp. 185–222.

Kleene, S. C. (1959). Recursive quantifiers and functionals of finite type I, *Transactions of the American Mathematical Society* **91**, pp. 1–52.

Kleene, S. C. (1962a). Turing-machine computable functionals of finite type I, in *Proceedings 1960 Conference on Logic, Methodology and Philosopy of Science* (Stanford University Press), pp. 38–45.

Kleene, S. C. (1962b). Turing-machine computable functionals of finite type II, *Proceedings of the London Mathematical Society* **12**, pp. 245–258.

Kleene, S. C. (1963). Recursive quantifiers and functionals of finite type II, *Transactions of the American Mathematical Society* **108**, pp. 106–142.

Klev, A. (2007). *Magister thesis* (ILLC Amsterdam).

Koepke, P. (2005). Turing computation on ordinals, *Bulletin of Symbolic Logic* **11**, pp. 377–397.

Koepke, P. and Koerwien, M. (2006). Ordinal computations, *Mathematical Structures in Computer Science* **16.5**, pp. 867–884.

Koepke, P. and Miller, R. (2008). An enhanced theory of infinite time register machines, in A. B. *et al.* (ed.), *Logic and the Theory of Algorithms*, *Springer Lecture Notes Computer Science*, Vol. 5028, Swansea (Springer), pp. 306–315.

Koepke, P. and Siders, R. (2006). Computing the recursive truth predicate on ordinal register machines, in A. B. *et al.* (ed.), *Logical Approaches to Computational Barriers*, Computer Science Report Series (Swansea), p. 21.

Kripke, S. (1964). Transfinite recursion on admissible ordinals I,II, *Journal of Symbolic Logic* **29**, pp. 161–162.

Levy, A. (1963). Transfinite computability (abstract), *Notices of the American Mathematical Society* **10**, p. 286.

Machover, M. (1961). The theory of transfinite recursion, *Bulletin of the American Mathematical Society* **67**, pp. 575–578.

Moschovakis, Y. N. (1974). *Elementary Induction on Abstract structures*, *Studies in Logic series*, Vol. 77 (North-Holland, Amsterdam).

Pitowsky, I. (1990). The physical Church-Turing thesis and physical computational complexity, *Iyyun* **39**, pp. 81–99.

Platek, R. (1966). *Foundations of Recursion Theory*, Ph.D. thesis, Stanford.

Putnam, H. (1965). Trial and error predicates and the solution to a problem of Mostowski, *Journal of Symbolic Logic* **30**, pp. 49–57.

Rathjen, M. (2005). An ordinal analysis of parameter-free Π^1_2 comprehension, *Archive for Mathematical Logic* **44**, 3, pp. 263–362.

Rogers, H. (1967). *Recursive Function Theory*, Higher Mathematics (McGraw).

Sacks, G. (1990). *Higher Recursion Theory*, Perspectives in Mathematical Logic (Springer Verlag).

Sacks, G. E. and Simpson, S. G. (1972). The α-finite injury method, *Annals of Mathematical Logic* **4**, pp. 343–367.

Shepherdson, J. and Sturgis, H. (????). Computability of recursive functionals, .

Shore, R. A. (1975). Splitting an α recursively enumerable set, *Transactions of the American Mathematical Society* **204**, pp. 65–78.

Shore, R. A. (1976). The recursively enumerable α-degrees are dense, *Annals of Mathematical Logic* **9**, pp. 123–155.

Simpson, S. (1999). *Subsystems of second order arithmetic*, Perspectives in Mathematical Logic (Springer).

Takeuti, G. (1960). On the recursive functions of ordinal numbers, *Journal of the Mathematical Society of Japan* **12**, pp. 119–128.

Thomson, J. (1954-55). Tasks and supertasks, *Analysis* **15**, 1, pp. 1–13.

Tugué, T. (1964). On the partial recursive functions of ordinal numbers, *Journal of the Mathematical Society of Japan* **16**, pp. 1–31.

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

Welch, P. D. (2004). Post's and other problems in higher type supertasks, in B. Löwe, B. Piwinger and T. Räsch (eds.), *Classical and New Paradigms of Computation and their Complexity hierarchies, Papers of the Conference Foundations of the Formal Sciences III*, *Trends in logic*, Vol. 23 (Kluwer), pp. 223–237.

Welch, P. D. (2005). Weak systems of determinacy and arithmetical quasi-inductive definitions, *arXiv: 0905.4412, to appear in the* Journal of Symbolic Logic *2010?* .

Welch, P. D. (2008). Turing Unbound: The extent of computations in Malament-Hogarth spacetimes, *British J. for the Philosophy of Science* **15**, 4, pp. 659–674.

Welch, P. D. (2009). Characteristics of discrete transfinite Turing machine models: halting times, stabilization times, and normal form theorems, *Theoretical Computer Science* **410**, pp. 426–442.