# Higher Type Recursion for Transfinite Machine Theory

Philip Welch(✉)

School of Mathematics, University of Bristol, Bristol BS8 1TW, England
p.welch@bristol.ac.uk
http://people.maths.bris.ac.uk/∼mapdw/

**Abstract.** We look at some preliminary work in the theory of transfinite Turing machines generalised in the manner of Kleene to higher type recursion theory. The underlying philosophy is that ordinary Turing computability and inductive definability is replaced by the example here of Infinite Time Turing Machine computability and quasi-inductive definability.

## 1 Introduction

The purpose of this paper is to give a purely descriptive account of how notions of 'recursion' obtained from transfinite computational machines could be harnessed to yield a theory of *higher type of recursion* using those machines. (To make it clear from the outset: type 0 objects are of the form: $n \in \omega$; type 1 are of the form $x : \omega \to \omega$, and type 2 are of the form $F : (\mathbb{N}^{\mathbb{N}}) \to \omega$ *etc.* We shall not deal with objects here of type higher than 2.)

We restrict ourself here to ideas and definitions. We summarise some results that characterise the semi-decidable sets for such notions, but all proofs must be omitted. The point is to indicate how analogies with Kleene's theory of Higher Type recursion from the late '50's and early '60's can be used to develop these ideas in the transfinite context.

Our transfinite machine will be the $\omega$ length tape Infinite Time Turing Machine ("ittm") model of Hamkins and Kidder [8] with which we shall assume the reader is familiar. Much of what we say generalises to machines with longer tapes.

We shall give analogies to Kleene's type-2 recursion and the objects that naturally arise there, but formulated for type-2 recursion using ittm's. We don't claim to give the final form of this: there are a number of decisions and choices along the way, that could have been made differently. Kleene's theory can be cast in that of *monotone inductive definitions* which we first recall. The concept corresponding to this for ittm-theory is that of a *quasi-inductive definition*. In Sect. 2 we give first a sketch of Kleene's theory applied to wellfounded trees of Turing machines ("tm" will always denote a regular Turing machine) and the type-2 objects that naturally occur here. The theory of hyperarithmetic sets and the fact that 'semi-decidable' in this context corresponds to $\Pi^1_1$ are of great


F. Manea et al. (Eds.): CiE 2019, LNCS 11558, pp. 72–83, 2019.
https://doi.org/10.1007/978-3-030-22996-2_7

weight in what follows. In Sect. 3 we give a description of the ittm version of this, according to a choice of type-2 oracles. As much for motivation, or additional justification for our structure, as for anything else, in Sect. 4 we state some applications results in low levels of determinacy.

## 2 Inductive Operators

Let $\Phi : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ be any arithmetic $\Gamma$ operator (that is '$n \in \Gamma(X)$' is an arithmetic relation of $X$ and $n$.)

**Definition 1.** *(i) $\Phi$ is* monotone *iff* $\forall X \subseteq Y \subseteq \mathbb{N} \quad \longrightarrow \quad \Phi(X) \subseteq \Phi(Y)$*;*
*(ii) $\Phi$ is* progressive *iff* $\forall X \subseteq \mathbb{N} \quad (X \subseteq \Phi(X))$.

In either case we set: $\Phi_0(X) = X$ and then: $\Phi_\alpha(X) = \Phi(\bigcup_{\beta<\alpha}(\Phi_\beta(X)))$. We call $\Phi$ *inductive* if it is monotone or progressive. Clearly $\Phi$ inductive implies there will be *fixed points* the least of which will be: $\Phi_\infty(X) =_{df} \Phi_\alpha(X)$ where $\alpha$ is least with $\Phi_\alpha(X) = \Phi_{\alpha+1}(X)$; clearly $\alpha$ will be countable.

The theory of inductive operators was heavily investigated in the 1960's and early 70's by Spector, Gandy, Hinman, Richter, Aczel, Moschovakis, Aanderaa, Cenzer and others. From this work developed Moschovakis's theory of *generalised definability* and *inductive definitions* over *abstract structures* [15]. This tied in with previous work in *admissibility theory* "The next admissible set" (Barwise-Gandy-Moschovakis Theorem, [1]), and the Spector-Gandy Theorem that: "$\Pi^1_1 = \Sigma_1(L_{\omega_1^{ck}})$" - $L_{\omega_1^{ck}}$ being the least admissible set over $\mathbb{N}$.

**Definition 2 (Quasi-inductive operators).** *Let $\Phi$ be any operator. Define iterates $\Phi$ as before except for limits $\lambda \leq On$:*

$$\Phi_\lambda(X) = \liminf_{\alpha\to\lambda} \Phi_\alpha(X) = \bigcup_{\alpha<\lambda} \bigcap_{\lambda>\beta>\alpha} \Phi_\beta(X).$$

For arithmetic operators this is, in effect, due to Burgess [3], but which has its roots in the notion of *revision theoretic definability* of Gupta and Belnap [6].

**Lemma 1.** *Any such operator has a least countable $\zeta = \zeta(\Phi, X)$ with $\Phi_\zeta(X) = \Phi_{On}(X)$. Moreover there is a cub class of ordinals, closed and unbounded beneath any uncountable cardinal, of ordinals $\xi$, with $\Phi_\zeta(X) = \Phi_{On}(X)$.*

There are not a huge number of examples of quasi-inductive operators in the literature, but an important one is that of an *infinite time turing machine* (ittm) where we regard the $\omega$-length tape(s) as a sequence of cells whose contents are revised according to the transition table of the program. This results in a *recursive* operator $\Phi$ which moreover only updates at most one cell, so one integer of $X$, at each stage. All the active new work takes place at the limit stages with the lim inf rule.

Kleene in [10] developed an equational calculus, itself evolving out of his analysis of the Gödel-Herbrand General Recursive Functions (on integers) from

the 1930's, but now enlarged for dealing with recursion in objects of finite type. A particular type-2 functional was that derived from the *ordinary jump* $\mathsf{oJ}$, where

$$\mathsf{oJ}(e, \boldsymbol{m}, x) = \begin{cases} 1 & \text{if } [e](\boldsymbol{m}, x)\downarrow \text{ (meaning has a } \textit{defined value} \text{ or } \textit{converges}) \\ 0 & \text{otherwise.} \end{cases}$$

(We shall also use the notation "$[e]^{\mathsf{I}}(p)$" rather than "$\{e\}^{\mathsf{I}}(p)$" to indicate that we are using Kleene recursion using tm's, and reserve the latter more usual notation for ittm recursion.) Here $\boldsymbol{m}$ is a string of integers, and $x$ a function $x : \omega \to \omega$ (thus an object of type 1) and $e$ the index number of an ordinarily Turing recursive functional of type-1 objects. (A vector of such functions will be denoted in bold.) The reader should note the use of the downarrow in $[e](\boldsymbol{m}, \boldsymbol{x})\downarrow$ to mean just what it says: the expression is *defined*, and for which we use *convergence* as a synonym. Similarly $[e](\boldsymbol{m}, \boldsymbol{x})\uparrow$ will mean the expression is *undefined* with synonym of *divergence*. Functions of type greater than 1 are conventionally called 'functionals', but we may occasionally let this slip.

The functional $\mathsf{oJ}$ can be considered as a functional just on type-2 objects (absorbing objects of lower type by their type-2 counterparts). Using coding of vectors of functions we ultimately think of this as $\mathsf{oJ}$ having domain $\omega \times {}^k\omega \times {}^l({}^\omega\omega)$ for any $k, l \in \omega$.

Kleene then developed (see Hinman [9] Ch. VI) a theory of generalised recursion in type-2 (and higher) functionals; in this theory a designation such as '$[e]^{\mathsf{I}}$' refers to the $e$'th function *recursive in the type-2 functional* $\mathsf{I}$. (Warning: this is not just the simple use of the oracle $\mathsf{I}$ in a linear computation as the notation might suggest, but refers to a tree of computation with calls to the oracle.) During a computation of, say, $[e]^{\mathsf{I}}(\boldsymbol{n}, \boldsymbol{y})$ oracle steps are allowed whereby the result of a query $(f, \boldsymbol{m}, \boldsymbol{x})$ is directly asked of $\mathsf{I}$, and an integer result, $\mathsf{I}(f, \boldsymbol{m}, \boldsymbol{x})$, is returned. (Of course even to make the query the values of each of the infinitely many values of the functions $\boldsymbol{x}$ have to already have been calculated; calculating each of these values can in turn require asking the oracle $\mathsf{I}$ for further values *etc.*; thus such a recursion can be represented by a tree, which if convergent is well founded, but is potentially infinitely branching at any node, with each branch calculating some $x(k)$ say.) In this formalism the index set $H_{\mathsf{oJ}}$ defined by:

$$H_{\mathsf{oJ}}(e) \leftrightarrow [e]^{\mathsf{oJ}}(e)\downarrow$$

is a complete semi-recursive (in $\mathsf{oJ}$) set of integers, and Kleene showed that this is in turn a complete $\Pi^1_1$ set of integers. Further he showed that the $\mathsf{oJ}$-*recursive sets of integers*, *i.e.* those sets $R$ for which

$$R(n) \leftrightarrow [e]^{\mathsf{oJ}}(n)\downarrow 1 \ \wedge\ \neg R(n) \leftrightarrow [e]^{\mathsf{oJ}}(n)\downarrow 0$$

for some index $e$, are precisely the hyperarithmetic ones. (See Hinman [9] Ch. VII.1 for a discussion of this.)

Kleene gave his account of recursion in objects of finite type which we have alluded to above in [10,13]. In order to give further weight to his definition he then showed it was extensionally equivalent to an alternative given by a Turing

machine model enhanced with oracle calls to a higher type functional, see [11,12]; this was just as for the case of ordinary Turing computation. Many different concepts of computation on numbers turned out to be equivalent. By showing that the equational model had the same functions as a Turing machine model he was emulating the same conceptual move Turing had made. In the first paper [11] he showed how any Turing machine computation of finite type could be achieved on the generalized recursive equational approach. The second paper [12] showed the reverse. In both directions a convergent, (so defined) computation could be represented, not as a finite tree of computations as for ordinary recursion, but now as a well-founded but in general infinitely branching tree of computations of function values - which in general required calculating infinite objects (as we indicated above), such as all values $x(n)$ for a function $x : \mathbb{N} \longrightarrow \mathbb{N}$, at some level in the tree before submitting that completely calculated function itself as an argument to a function of higher type at the level above. The wellfounded tree of either functional calculations, or of Turing machine computational calls, depending on the representation, witnessed a successfully defined or convergent computation. The tree occurs dynamically as part of the computational process.

Our account here is motivated in spirit by that latter approach. Instead of using an equational calculus we shall couch our model not just in terms of the Kleenean Turing machine, but in terms of ittm's and their computations, viewed as quasi-inductive operators now recursive in a certain operator $\mathsf{iJ}$ in place of Kleene's $\mathsf{oJ}$.

Viewed as a class of quasi-inductive operators, the output tape (or every third cell say of a single tape model) of an ittm represents an element of Cantor space at any stage; that output tape may or may not converge to a fixed value. If it does then the the real there is to be regarded as the output of the computation. Notice this is a more generalised notion than that of the machine *halting* and hence with a fixed output tape for that reason. Halting is really just a special case of the basic phenomenon of 'fixed output'. The idea that a tape is *eventually settled* is broader in the infinite time context: a calculation can continue indefinitely, without any changes to the output tape section. It must have seemed natural to consider only the halting computations when first thinking about ittm behaviour, but as [17] showed, even to characterise those halting calculations required stepping back and analysing the whole class of eventually settled computations: the latter we regard as more fundamental, and as characteristic of the ittm process. To analyse ittm behaviour is to analyse the eventually settled outputs (which we shall call 'fixed outputs' below), and to find out what they are capable of computing requires analysing those fixed outputs, not just the more specialised halting outputs. The Spector class naturally associated to this form of definability by itttm's is precisely that using this fixed output rather than the proper subclass using nominally halting output. And of course this is in accord with the quasi-inductive scheme above.

Given a set $A \subseteq \omega \cup {}^{\omega}2$, this can be used as an oracle during a computation on an ittm in a familiar way: ? *Is the integer on (or is the whole of) the current output tape contents an element of* $A$? and receive a 1/0 answer for "Yes"/"No".

We identify elements of $\omega$ as coded up in ${}^{\omega}2$ in some fixed way, and so may consider such $A$ as always subsets of ${}^{\omega}2$. But further: since having $A$ respond with one 0/1 at a time can be repeated, by using an $\omega$-sequence of queries/responses, we could equally well allow $A$ to return an element $f \in {}^{\omega}2$ as a response (we have no shortage of time). We could then allow as functionals also $\mathsf{I} : {}^{\omega}2 \longrightarrow {}^{\omega}2$. However for this paper we shall only consider functionals into $\omega$. Some examples follow. As is usual we let $\{e\}$ represent the partial function computed by the $e$'th ittm programmed machine $P_e$.

**Definition 3.** *(The infinite time jump* $\mathsf{iJ}$*)*
*(i) We write* $\{e\}(\boldsymbol{m}, \boldsymbol{x})\downarrow$ *if the e'th ittm-computable function with input* $\boldsymbol{m}, \boldsymbol{x}$ *has a fixed output* $c \in 2^{\mathbb{N}}$*, in which case we write* $\{e\}(\boldsymbol{m}, \boldsymbol{x}) = c$*.*
*(ii) We then define* $\mathsf{iJ}$ *by:*

$$\mathsf{iJ}(e, \boldsymbol{m}, \boldsymbol{x}) \quad = \begin{cases} 1 & \text{if } \{e\}(\boldsymbol{m}, \boldsymbol{x})\downarrow; \\ 0 & \text{otherwise (for which we write } \{e\}(\boldsymbol{m}, \boldsymbol{x})\uparrow). \end{cases}$$

The functional $\mathsf{iJ}$ then is the counterpart of the standard tm operator $\mathsf{oJ}$.

**Definition 4.** *For $x$ a real, the complete (ordinary) ittm-semirecursive-in-$x$ set, denoted by $\tilde{x}$ is the set of integers $\{e \mid \{e\}(e, x)\downarrow\}$.*

One consequence of the (relativized to a real $x$) $\lambda$-$\zeta$-$\Sigma$-Theorem (*cf.* [16] Thm 2.6) is that $\tilde{x}$ is recursively isomorphic to the complete $\Sigma_2$-Theory of $L_{\zeta^x}[x]$.

## 3 Higher Type Recursion

In the Kleenean recursion in type-2 functionals, in [11,12] a successful computation (meaning one with output) could be effected by imagining tm's placed at nodes on a wellfounded tree, with computations proceeding at nodes that make computation calls to a lower node, seeking the value of some $x(k)$ say. The computation time at each node, regarding each call to a lower node as being just one step in the computation of the calling node, is then finite. (For otherwise the computation at the node is never completed and the whole overall computation will fail.) An overall computation may fail by instituting a series of calls to subcomputations that form an infinite descending path in the tree. In such cases the machines on the path all hang after finitely many steps, all waiting for data to be passed up from the immediate subcomputation it has called.

In the ittm case we may again conceive of an overall or master ittm computation taking place at the top level; such a computation may take infinitely many steps in time, and will be considered as successful if the output tape is fixed from some point in time onwards. The master computation may make queries of a type-2 functional $\mathsf{I}$ in which the computation is considered recursive. It may call subcomputations of exactly the same type: ittm's with the capability to make oracle queries of $\mathsf{I}$.

We give a more detailed description of this as a representation in terms of underlying ittm's. $\{e\}^{\mathsf{I}}(\boldsymbol{m}, \boldsymbol{x})$ will represent the $e$'th program in the usual format,

say transition tables, but designed with appeal to oracle calls possible. We are thus considering computations of a partial function $\{e\}^{\mathsf{I}} : {}^{k}\omega \times {}^{l}({}^{\omega}2) \to {}^{\omega}2$. Such a computation has potentially computation time, or stages, unbounded in the ordinals.

The computation of $P_e^{\mathsf{I}}(\boldsymbol{m}, \boldsymbol{x})$ proceeds in the usual ittm-fashion, working as a tm at successor ordinals and taking lim inf's of cell values *etc.* at limit ordinals. (We take lim inf's rather than lim sup's as this accords more with the notion characteristic functions of quasi-inductive operators. This makes no difference to the computational possibilities of ittms's or here at higher types.) At a time $\alpha$ an oracle query may be initiated. We may conventionally fix that the real number subject to query is that infinite string on the even numbered cells of the scratch type. If this string is $(f, m, y_0, y_1 \ldots,)$ then setting $y = y_0, y_1 \ldots$, the *query* or *oracle call* which we shall denote $Q(\mathsf{I}, f, m, y)$ is the question: ? *What is* $\mathsf{I}(z)$ *where* $P_f^{\mathsf{I}}(m, y)\downarrow z$ ? and at stage $\alpha+1$ receives the value $\mathsf{I}(z)$. If it is not the case that $P_f^{\mathsf{I}}(m, y)\downarrow z$ for any $z$, *i.e.*, it fails to have a fixed output, then there is no $z$ to which $\mathsf{I}$ can be applied, and the overall computation fails. (We could try to stay closer to the Kleenean setting, where a tree branches infinitely often downwards, to compute for some $z \in {}^{\omega}\omega$, $z(0), z(1), \ldots$ in turn, and then can ask for $\mathsf{I}(z)$. There, if any of the computations $z(k)$ failed, then the query to $\mathsf{I}$ did not take place, and the overall computation failed. But one thing we have with ittm computation is plenty of time, so we can amalgamate the individual computations $z(k)$ as simply one computation of all of $z$.)

Space prohibits a formal definition of the representation above, but we can determine its effect as follows *via* an inductive operator $I$. Just as the Kleene equational calculus can be seen to build up in an inductive fashion a set of indices $\Omega[\mathsf{I}]$ for successful computations recursive in $\mathsf{I}$ (see Hinman [9], pp. 259–261), so we can define the fixed point of a monotone operator $I = I^{\mathsf{I}}$ on $(\omega \times \omega^{<\omega} \times (\omega^{\omega})^{<\omega}) \times \omega^{\omega}$ which will give us the successful ittm-computations recursive in $\mathsf{I}$.

**Definition 5.** *We set* $I(X) =:$

$$\{\langle\langle e, \boldsymbol{m}, \boldsymbol{x}\rangle, z\rangle |\ P_e^X(\boldsymbol{m}, \boldsymbol{x})\downarrow z \text{ is an ittm-computation making only oracle calls } Q(X, e', \boldsymbol{m'}, \boldsymbol{x'}) \text{ and receiving back } \mathsf{I}(z') \text{ where } X(\langle e', \boldsymbol{m'}, \boldsymbol{x'}\rangle) = z'\}.$$

*As this is monotone, we may let*
$I^0 = \varnothing$; $I^{<\alpha} = \bigcup_{\beta<\alpha} I^{\beta}$ *&* $I^{\alpha} = I(I^{<\alpha})$ *in the usual way, and reach a least fixed point* $I^{\infty}$.

Then:

**Theorem 1 (The $\{e\}$'th function generalised recursive in $\mathsf{I}$).** *Using* $I^{\infty}$: $\{e\}^{\mathsf{I}}(\boldsymbol{m}, \boldsymbol{x})$ *is defined, or convergent, with output* $z$ *iff* $I^{\infty}(\langle e, \boldsymbol{m}, \boldsymbol{x}\rangle) = z$.
*In which case we set* $\{e\}^{\mathsf{I}}(\boldsymbol{m}, \boldsymbol{x}) = z$. *Otherwise it is undefined or divergent.*

Overall we have a *computation tree* - also called a *tree of subcomputations*, with subcomputation calls performed at branching nodes below the top level. However, although the computation is most easily represented by a tree, we may

think of the computation as a linear sequential process as we visit each node of the tree in turn.

We therefore make the following conventions. During the calculation of $\{e\}^{\mathsf{I}}(\boldsymbol{m}, \boldsymbol{x})$ the initial calculation takes place at the topmost node $\nu_0$ which we declare to be *at Level* 0 in our computation tree $\mathfrak{T} = \mathfrak{T}^{\mathsf{I}}(\boldsymbol{e}, \boldsymbol{m}, \boldsymbol{x})$. Let us suppose the first oracle query concerning $\{f_0\}^{\mathsf{I}}(n_0, y_0)$ is made at some stage. The tree $\mathfrak{T}$ will then have a node $\nu_1$ below $\nu_0$, labelled with $\langle f_0, n_0, y_0\rangle$ and we declare the computation $\{f_0\}^{\mathsf{I}}(n_0,y_0)$ to be performed at this Level 1. Thus 'control' of the overall process is at the level of the node. Further, we may define the *overall length function* $H = H(\mathsf{I}, e, \boldsymbol{m}, \boldsymbol{x})$ as the length of the computation that occurs at the nodes of the wellfounded part of $\mathfrak{T}$. Sequentially $H$ totals up the ordinal number of stages of operation at each of the nodes where control currently resides.

**Definition 6.** *(i) The* level *of the computation* $\{e\}^{\mathsf{I}}(\boldsymbol{m}, \boldsymbol{x})$ *at time* $\alpha$ *(as given by* $H$*), denoted* $\Lambda(e, \mathsf{I}, (\boldsymbol{m}, \boldsymbol{x}), \alpha)$*, is the level of the node* $\nu_\iota$ *at which the overall computation is being performed at time* $\alpha$*, where:*
*(ii) the* level *of a node* $\nu_\iota$ *is the length of the path in the tree from* $\nu_0$ *to* $\nu_\iota$*.*
*(iii) By* Level $n$ *we accordingly mean the set of nodes in the tree with level* $n$*.*

Thus for a convergent computation, at any time the level is a finite number ('depth' would have been an equally good choice of word). A *divergent computation* is one in which either (i) an oracle call resulting in a calculation at some node fails to produce an output $z$ (and so no value $\mathsf{I}(z)$ can be returned to the level above) or (ii) $\mathfrak{T}$ is illfounded (with a rightmost path of order type then $\omega$).

Recall that a 'snapshot' at time $\gamma$ in a computation by an ittm is the $\omega$-sequence of bits of information consisting of the current read/write head position, transition state number, and the sequence of cell values. The snapshots up to the stage in a calculation $P_e^{\mathsf{I}}(\boldsymbol{m}, \boldsymbol{x})$ where it ends its first loop (if this occurs) will have all the relevant information then in the calculation: everything thereafter is mere repetition. (This would be undefined if the computation tree is illfounded). We say that a computation *'exhibits final looping behaviour'* ('at stage $\sigma$', or 'by stage $\tau$'), if there are stages or times $\xi < \sigma\, (\leq \tau)$ with at the top level (a) identical snapshots at $\xi$ and $\sigma$, and moreover (b) no cell that had a stable value at time $\xi$ changes that value in the interval $(\xi, \sigma)$.

**ITTM Recursion in $^2\mathsf{E}$.** We shall draw to a close the discussion of generalised recursion in functions $\mathsf{I}$ as this will take us too far from our goal, and shall leave this for future work. For us, as for Kleene, recursion in $^2\mathsf{E}$ is fundamental. Recall, for $y \in \mathbb{N}^{\mathbb{N}}$, $^2\mathsf{E}(y) = 0$ if $\exists n\, y(n) = 0$ and $^2\mathsf{E}(y) = 1$ otherwise. Many of the theorems of type-2 recursion about functionals $\mathsf{F}$ have to be prefixed with the requirement that $^2\mathsf{E}$ is recursive in $\mathsf{F}$. (Such $\mathsf{F}$ are called *normal*.)

**Definition 7.** *We say* $\mathsf{F}$ *is* (generalised) ittm-partial recursive in $\mathsf{G}$ *if there is an index* $e$ *so that* $\mathsf{F} = \{e\}^{\mathsf{G}}$. $\mathsf{F}$ *is* ittm-recursive in $\mathsf{G}$ *if it is partial recursive in* $\mathsf{G}$ *and total. A relation* $R$ *is* ittm-recursive in $\mathsf{I}$ *if its characteristic function is.* $R$ *is* ittm semi-recursive in $\mathsf{I}$ *if it is the domain of a functional ittm-partial recursive in* $\mathsf{I}$.

Kleene showed that the functionals ${}^2\mathsf{E}$ and $\mathsf{oJ}$ are mutually (Kleene) recursive in each other (*cf.* [9] VI.1.4.). We shall have this too:

**Theorem 2.** *The functionals* $\mathsf{iJ}$ *and* ${}^2\mathsf{E}$ *are mutually ittm-recursive.*

We wish to apply this theory to the particular case of $\mathsf{iJ}$ - the *infinite jump.*

A number of elementary facts concerning computation trees $\mathfrak{T}$ living in transitive admissible sets $M$ may be proven.

**Lemma 2.** *Suppose* $\{e\}^{\mathsf{iJ}}(\boldsymbol{m}, \boldsymbol{x})$ *has a computation tree* $\mathfrak{T} \in M$*, and with* $\boldsymbol{x} \in M$*, where* $M$ *is a transitive admissible set, closed under the function* $y \rightarrowtail \tilde{y}$*. Then* $(\{e\}^{\mathsf{iJ}}(\boldsymbol{m}, \boldsymbol{x})$ *is convergent*$)^M \longleftrightarrow \{e\}^{\mathsf{iJ}}(\boldsymbol{m}, \boldsymbol{x})$ *is convergent.*

It was an essential feature of ordinary ittm-theory that if a computation $P_e(m)$ produced an output it would always have done this by stage $\zeta$ where $\zeta$ is least so that for some $\Sigma > \zeta$ we had $L_\zeta \prec_{\Sigma_2} L_\Sigma$; this was shown in the "$\lambda$-$\zeta$-$\Sigma$ Theorem" (see [16] 2.1 and 2.3). The $\Sigma_2$ liminf nature of the limit rule underlay this, and the same is true here.

**Definition 8.** *A pair of ordinals* $(\mu, \nu)$ *is a* $\Sigma_2$-extendible pair *if* $L_\mu \prec_{\Sigma_2} L_\nu$ *and moreover* $\nu$ *is the least such with this property with respect to* $\mu$*. We say* $\mu$ *is* $\Sigma_2$-extendible *if there exists* $\nu$ *with* $(\mu, \nu)$ *a* $\Sigma_2$-extendible pair*. By relativisation, a pair of ordinals* $(\mu, \nu)$ *is an* $(\boldsymbol{x}, \mathsf{I})$-$\Sigma_2$-extendible pair*, and* $\mu$ *is* $(\boldsymbol{x}, \mathsf{I})$-$\Sigma_2$-extendible*, if* $L_\mu[\boldsymbol{x}, \mathsf{I}] \prec_{\Sigma_2} L_\nu[\boldsymbol{x}, \mathsf{I}]$.

Then of importance for our purposes are:

**Lemma 3.** *The computation* $\{e\}^{\mathsf{I}}(\boldsymbol{m}, \boldsymbol{x})$ *exhibits final looping behaviour if and only if there exists some* $(\boldsymbol{x}, \mathsf{I})$*-*$\Sigma_2$*-extendible pair* $(\zeta, \Sigma)$ *so that* $\Lambda(e, \mathsf{I}, \boldsymbol{x}, \zeta) = 0$*.*

The dependence on $\mathsf{I}$ in the above is natural. With $\mathsf{iJ}$ it can be dropped:

**Lemma 4.** *The computation* $\{e\}^{\mathsf{iJ}}(\boldsymbol{m}, \boldsymbol{x})$ *exhibits final looping behaviour if and only if there exists some* $\boldsymbol{x}$*-*$\Sigma_2$*-extendible pair* $(\zeta, \Sigma)$ *so that* $\Lambda(e, \mathsf{iJ}, \boldsymbol{x}, \zeta) = 0$*.*

Usual methods prove an $S^n_m$-theorem and:

**Theorem 3 (The $\mathsf{I}$-Recursion theorem).** *If* $\mathsf{F}(e, \boldsymbol{m}, \boldsymbol{x})$ *is ittm-recursive in* $\mathsf{I}$*, there is* $e_0 \in \omega$ *so that*

$$\{e_0\}^{\mathsf{I}}(\boldsymbol{m}, \boldsymbol{x}) = \mathsf{F}(e_0, \boldsymbol{m}, \boldsymbol{x}).$$

*Another Example: Lubarsky's Feedback-ittm Recursions*
We are indebted to Lubarsky's work in [14] and grateful for discussions with him on his earlier FITTM's (= Feedback ITTM's). His notion of 'feedback' uses the concept of properly halting where the basic outcome occurs when an ITTM *halts* rather than having, as here, a fixed output. (We have indicated above why we consider 'fixed output' to mean a fixed output tape.) He describes wellfounded

computation trees, not as arising from a Kleene style recursion, but as ITTM's (with extra tapes) that have the additional state of "an oracle query *does the Feedback ITTM with program the content of the first additional tape on input the content of the second* [*tape*] *converge?*" which will receive a Yes/No answer. (As intimated, convergence is halting.) He then describes the semantics of such computations as wellfounded trees, where subcalls are again queries of the same type ("*Does* $\{e\}_{FITTM}(x)$ *halt*?"). An FITTM computation *freezes* if the tree becomes illfounded. He asks a number of questions, such as to the ranks for the wellfounded trees occurring, what are the reals output or appearing on tapes of such machines. We briefly state answers to these below.

We may describe an induction building up directly the class of successful FITTM-computations as a fixed point of a monotone operator, in this spirit, just as in Definition 5 above. However we construe this fixed point as that arising from a type-2 operator, let us call it here $\mathsf{hJ}$, from an ittm-recursion defined as in this paper. Recursion in $\mathsf{hJ}$ then also becomes an example of ittm-recursion in ${}^2\mathsf{E}$ with Theorem 2 applying again: $\mathsf{hJ}$ and ${}^2\mathsf{E}$ are mutually ittm-recursive in each other. Thus for us Feedback-ITTM computations become a particular example of this higher type ittm recursion, and the class of $x \in \mathbb{N}^{\mathbb{N}}$ FITTM-computable coincide with those ittm-recursive in ${}^2\mathsf{E}$.

**Computation Times.** The Lemmata 3 and 4 above give sufficient conditions for a computation $\{e_0\}^{\mathsf{iJ}}(\boldsymbol{m}, \boldsymbol{x})$ to converge. We need to find out exactly how long computations in $\mathsf{iJ}$ take in order to characterise the $\mathsf{iJ}$-recursive and semi-recursive sets. The clue is that ordinary ittm-computations can compute the theories and codes for the levels of the $L$-hierarchy up to the end of the first $\Sigma_2$-extendible pair interval $(\zeta, \Sigma)$. (This is shown in [17], and in [4] a programme is explicitly given that shows how codes and theories can be simultaneously produced by an ordinary ittm recursion on $\alpha$ for $\alpha < \Sigma$; after stage $\Sigma$ it drops into a repeating loop of reproducing the results on its output tape of the $\alpha \in (\zeta, \Sigma)$.) A machine that writes out codes for $L_\alpha$'s must in some sense be, at least akin to, a universal machine, since by absoluteness, any ittm computation on integer input can be run in $L$. Here we have, in effect, ittm's that can whilst within such a $\Sigma_2$-extendible interval, call other ittm's as part of a subroutine. It might not be inconceivable that such behaviour is overall fashioned, when they try to write codes for levels $L_\alpha$'s, by their reaching levels of the $L$-hierarchy, where the $\Sigma_2$-extendible pairs become *nested*.

**Definition 9.** *For* $m \geq 1$ *an* $m$-depth $\Sigma_2$-nesting *of an ordinal* $\alpha$ *is a sequence* $(\zeta_n, \sigma_n)_{n<m}$ *so that*
*(i) if* $m = 1$ *then* $\zeta_0 < \alpha < \sigma_0$*;*
*(ii) if* $0 < n + 1 < m$ *then* $\zeta_n \leq \zeta_{n+1} < \alpha < \sigma_{n+1} < \sigma_n$*;*
*(iii) if* $k < m$ *then* $L_{\zeta_k} \prec_{\Sigma_2} L_{\sigma_k}$*.*

We may show that there are processes generalised ittm-recursive in $iJmf$ that compute levels of $L$ up to the points where any finite depth of nesting occurs, where each additional depth of nesting corresponds to computing up to repeating snapshots at one depth lower in $\mathfrak{T}$. It then seems inevitable that illfounded trees

must ultimately occur by some ordinal corresponding to *infinite depth nesting*. But *prima facie* there is no such ordinal, since there can be no infinite descending chain $\sigma_{n+1} < \sigma_n$ in the above definition.

We thus shall want to consider *non-standard admissible models* $(M, E)$ of KP together with some other properties. We let $\mathrm{WFP}(M)$ be the wellfounded part of the model. By the so-called 'Truncation Lemma' it is well known (*v.* [2]) that this wellfounded part must also be an admissible set. Usually for us the model will also be a countable one of "$V = L$". Let $M$ be such and let $\alpha = \mathrm{On} \cap \mathrm{WFP}(M)$. By the above $\alpha$ is thus an admissible ordinal, *i.e.* $L_\alpha$ will also be a KP model. As remarked, an '$\omega$-depth' nesting cannot exist by the wellfoundedness of the ordinals. However an illfounded model $M$ when viewed from the outside may have infinite descending chains of $M$-ordinals in its illfounded part. These considerations motivate the following definition.

**Definition 10.** *An* infinite depth $\Sigma_2$-nesting of $\alpha$ based on $M$ *is a sequence* $(\zeta_n, s_n)_{n<\omega}$ *with:*
*(i)* $\zeta_n \leq \zeta_{n+1} < \alpha \subset s_{n+1} \subset s_n$*; (ii)* $s_n \in \mathrm{On}^M$*; (iii)* $(L_{\zeta_n} \prec_{\Sigma_2} L_{s_n})^M$.

Thus the $s_n$ form an infinite descending $E$-chain (where, as above, $E$ is the membership relation of the illfounded model) through the illfounded part of the model $M$.

Whilst any countable transitive admissible set can be extended to have an illfounded part, (again *v.* [2]) and, for example, there are illfounded end-extensions of $L_{\omega_1^{ck}}$, that does not mean that this latter model can be extended to an illfounded model $M$ which supports an infinite depth $\Sigma_2$-nesting: a relatively large countable admissible $\beta$ is needed for that:

**Definition 11.** *Let* $\beta_0$ *be the least ordinal* $\beta$ *so that* $L_\beta$ *forms the wellfounded part of an admissible end-extension* $(M, E)$ *based on which there exists an infinite depth* $\Sigma_2$*-nesting of* $\beta$.

It turns out that $L_{\beta_0}$ is a model of $\Sigma_1$-Separation. Hence it has a proper, and so least, $\Sigma_1$-elementary submodel: $L_{\alpha_0} \prec_{\Sigma_1} L_{\beta_0}$. These ordinals feature in what follows.

## 4 Conclusions

**Theorem 4.** *(i) If a recursion* $\{e\}^{\mathsf{iJ}}(\boldsymbol{m})$ *converges, then it does so by time* $\alpha_0$*, and the latter ordinal is the supremum (over* $e$ *and* $\boldsymbol{m}$*) of convergence times of such computations. (ii) There is a recursion* $\{h\}^{\mathsf{iJ}}(\boldsymbol{m})$ *that only diverges at* $\beta_0$*, and all such divergent computations diverge before or at this time.*

**Lemma 5.** *(i) The* iJ*-recursive sets of integers are precisely those of* $L_{\alpha_0}$*; (ii) the* iJ*-semi-recursive sets are those* $\Sigma_1(L_{\alpha_0})$. Q.E.D.

The following answers two questions of Lubarsky:

**Corollary 1.** *The reals appearing on the tapes of freezing fittm-computations of* [14] *are precisely those of $L_{\beta_0}$; similarly the supremum of the ranks of the wellfounded parts of freezing fittm-computation trees is also $\beta_0$.*

**Lemma 6.** *The complete ittm-semidecidable-in-*iJ *set of integers*

$$K = \{(e, m) \in \omega \times \omega \mid \{e\}^{\mathsf{iJ}}(e)(e, m) = 1\}$$

*as well as*

$$H^{\mathsf{iJ}}(e) \leftrightarrow \{e\}^{\mathsf{iJ}}(e)\downarrow$$

*are recursively isomorphic to the complete $\Sigma_1$-Theory of $\langle L_{\alpha_0,\in}\rangle$.*

These last two lemmata can be compared with a result of Kleene *et al.*:

**Theorem 5.** *The complete (Kleene)-semidecidable in* oJ *set of integers is recursively isomorphic to the complete $\Sigma_1$-Theory of $\langle L_{\omega_1^{ck}}, \in\rangle$. The* oJ*-recursive sets of integers are precisely those of $L_{\omega_1^{ck}}$, that is, the hyperarithmetic sets.*

**A Postlude.** In earlier work we had located in the $L$-hierarchy winning strategies for $\Sigma^0_3$ two person perfect information games. The games in [18] connected to nested $\Sigma_2$-extendability. The presumed connection with ittm's becomes a intriguing question, and most of this work was motivated by trying to understand this. The summary above indeed ties in with these results, which we mention here without explaining the connection. See [19].

**Theorem 6.** *Let $\eta$ be least so that for any $\Sigma^0_3$-game there is a winning strategy for one of the players definable over $L_\eta$. Then $\eta = \beta_0$.*

Subsequently S. Hachtman ([7]) found another remarkable characterisation of $\beta_0$:

- *Let $\gamma$ be least so that, as a model of a fragment of second order arithmetic, $\mathbb{R} \cap L_\gamma$ is a model of $\Pi^1_2$-monotone induction. Then $\gamma = \beta_0$.*

**Open Questions.** As can perhaps be seen from this sketch there are more open questions than known facts. A closer analysis of ittm recursions in general type 2 functionals needs to be done:

*Q1 Formulate a* Stage Comparison Theorem *for ittm-recursion. (See* [9].VI*)*
Much as there are several approaches to the hyperarithmetic sets uses Kleene recursion, there are notation systems associated with ittm-theory. One can use the theory of $\Im\Sigma^0_3$-monotone operators to obtain *norms*, thus prewellorderings, on ittm semi-decidable sets. Presumably many features of Kleene recursion have some analogue for ittm recursion.

*Q2 What is the correct definition and properties of the* superjump *(due to Gandy for Kleene recursion) for ittm higher type recursions? (See* [9].VI*)*
We have only considered type-2 recursions to date.

*Q3 Is there a suitable notion of ittm-recursion in this spirit at types-3 and above?* In another direction one can enlarge the notion of computation by taking on the *hypermachines* of [5]. Such machines may have loops at $\Sigma_n$-extendible ordinals by analogy with the ittm's.

*Q4 Develop a theory of higher type hypermachine recursion.*

## References


1. Barwise, J., Gandy, R., Moschovakis, Y.: The next admissible set. J. Symb. Log. **36**(1), 108–120 (1971)
2. Barwise, K.: Admissible Sets and Structures. Perspectives in Mathematical Logic, vol. 2. Springer, Heidelberg (1975)
3. Burgess, J.: The truth is never simple. J. Symb. Log. **51**(3), 663–681 (1986)
4. Friedman, S.D., Welch, P.: Two observations regarding infinite time turing machines. In: Dimitriou, I. (ed.) Bonn International Workshop on Ordinal Computability, pp. 44–48. Hausdorff Centre for Mathematics, University of Bonn, Bonn (2008)
5. Friedman, S.D., Welch, P.: Hypermachines. J. Symb. Log. **76**(2), 620–636 (2011)
6. Gupta, A., Belnap, N.: The Revision Theory of Truth. MIT Press, Cambridge (1993)
7. Hachtman, S.: Determinacy and monotone inductive definitions. Isr. J. Math. (to appear)
8. Hamkins, J., Lewis, A.: Infinite time Turing machines. J. Symb. Log. **65**(2), 567–604 (2000)
9. Hinman, P.: Recursion-Theoretic Hierarchies. $\Omega$ Series in Mathematical Logic. Springer, Berlin (1978)
10. Kleene, S.: Recursive quantifiers and functionals of finite type I. Trans. Am. Math. Soc. **91**, 1–52 (1959)
11. Kleene, S.: Turing-machine computable functionals of finite type I. In: Proceedings 1960 Conference on Logic, Methodology and Philosophy of Science, pp. 38–45. Stanford University Press, Stanford (1962)
12. Kleene, S.: Turing-machine computable functionals of finite type II. Proc. Lond. Math. Soc. **12**, 245–258 (1962)
13. Kleene, S.: Recursive quantifiers and functionals of finite type II. Trans. Am. Math. Soc. **108**, 106–142 (1963)
14. Lubarsky, R.: ITTM's with feedback. In: Schindler, R.D. (ed.) Ways of Proof Theory. Ontos (2010)
15. Moschovakis, Y.: Elementary Induction on Abstract structures. Studies in Logic series, vol. 77. North-Holland, Amsterdam (1974)
16. Welch, P.D.: Eventually infinite time Turing degrees: infinite time decidable reals. J. Symb. Log. **65**(3), 1193–1203 (2000)
17. Welch, P.D.: The length of infinite time Turing machine computations. Bull. Lond. Math. Soc. **32**, 129–136 (2000)
18. Welch, P.D.: Weak systems of determinacy and arithmetical quasi-inductive definitions. J. Symb. Log. **76**, 418–436 (2011)
19. Welch, P.D.: $G_{\delta\sigma}$-games. Isaac Newton Preprint Series. No. NI12050-SAS (2012)