Parallel Feedback Turing Computability

Robert S. Lubarsky
Dept. of Mathematical Sciences
Florida Atlantic University
Boca Raton, FL 33431
Robert.Lubarsky@alum.mit.edu

July 28, 2015

Abstract

In contrast to most kinds of computability studied in mathematical logic, feedback computability has a non-degenerate notion of parallelism. Here we study parallelism for the most basic kind of feedback, namely that of Turing computability. We investigate several different possible definitions of parallelism in this context, with an eye toward specifying what is so computable. For the deterministic notions of parallelism identified we are successful in this analysis; for the non-deterministic notion, not completely.

keywords: parallel computation, feedback, determinism, non-determinism, reflection, gap-reflection, admissibility
AMS 2010 MSC: 03D10, 03D60, 03D70, 03E10, 03E75

1 Introduction

Parallelism, as far as the author is aware, has not been studied much in the kind of computability theory done by mathematical logicians (Turing degrees, arithmetic sets, admissibility). This is for a good reason: it can be mimicked, via dovetailing. Using a universal machine, a parallel computation can be simulated by a sequential computation. This is in stark contrast with complexity theory. For instance, an NP problem can be understood as a polynomial problem with parallelism, so the addition of parallelism to polynomial computation results in a new and quite important notion.

This paper studies parallelism in an extension of Turing computability where it does make a difference, namely feedback. Feedback was first identified in [5], p. 406-407, even if not under that name, where some of the results of [1] were anticipated. Oddly enough, even though that was a very prominent text for decades, likely the best-known in (using the terminology of the day) recursion theory, no one ever picked up on those ideas. It was re-discovered independently for infinite time Turing machines in [2], where even parallelism was discussed, albeit briefly. Something was actually done with parallelism in [1], where it was shown that its addition to feedback Turing computability is non-trivial, in the sense that it gets you strictly more than you had before. (It was also shown there that parallelism added to feedback primitive recursion is essentially trivial in the same sense.) It was left open there just what is parallel feedback Turing computable (pfc in what follows).

In this paper, the issues around parallelism are clarified somewhat. For one, as already discussed in [1], there are several different ways that parallelism can be included in this framework. It is not yet clear which are fruitful. By analyzing some of them, we hope to bring this issue along.

In the end, we discuss three. In the next section, we show one to provide no new computational power, and hence (presumably) to be uninteresting. In section 3 we turn to the one from [1]; we expand upon its semantics, but despite that are still unable to characterize just what is so computable, although an upper bound is provided later. Finally, in the last section we define a semantics which is in a sense intermediate between these other two, and are successful in its characterization; we also provide here the upper bound for earlier.

We assume some familiarity with both feedback and parallelism, as presented in [1] and [2]. To summarize briefly, the oracle in a feedback computation contains the convergence and divergence facts about computations that call that very same oracle. If a computation queries an oracle about a computation for which the oracle does not have an answer, that computation freezes dead in its tracks. This allows for parallelism, since a computation could then ask the oracle which programs in a parameterized collection of programs do not freeze.

2 Absolutely Deterministic Parallelism

If a parallel oracle call about $\langle e\rangle(\cdot)$ is to return an n such that $\langle e\rangle(n)$ does not freeze (if any), there is a clear invitation to non-determinism: which n? Indeed, in [1], a semantics for deterministic parallelism was offered and then quickly passed over, as it turns out for a good reason: it gets you nothing new. Here we show this, if for no other reason than to demonstrate that this definition should no longer be considered.

The idea is that the oracle is supposed to return the "least" n leading to non-freezing, by some measure. The measure to be used is primarily that of ordinal height of a computation. That is, n minimizes the height of the tree of sub-computations. To help keep this paper self-contained, this tree will be presented, albeit in a way different from in [2] or [1], tailored to the purpose at hand.

The tree $D_\alpha^{(e,n)}$ (D for determinism) is defined inductively on α, simultaneously for all e, n, as is whether $\text{rank}(e,n) = \alpha$. Assume this is known for all $\beta < \alpha$. Start the run of $\langle e\rangle(n)$, which is considered as taking place at the root of $D_\alpha^{(e,n)}$. Suppose at some stage of that computation, an oracle call e' is made. Then a child of the root is established, to the right of any previous children, for the outcome of this oracle call. Suppose there is an n' such that

$\text{rank}(e', n') < \alpha$. Then let n' be chosen to minimize this rank; if there is more than one such, then among those pick the least in the natural ordering of ω. The tree $D^{(e',n')} = D^{(e',n')}_{\text{rank}(e',n')}$ is placed at the child, and the value $\langle e'\rangle(n')$ is returned to the main computation, which then continues. If there is no such n', then the computation pauses, and the construction of $D^{(e,n)}_\alpha$ is finished.

If no oracle calls pause, then by this stage α the computation $\langle e\rangle(n)$ is seen to be non-freezing; $D^{(e,n)}$ can be taken to be $D^{(e,n)}_\alpha$ and is the tree of sub-computations; $\text{rank}(e, n) \leq \alpha$; and the value of $\langle e\rangle(n)$ is the content on the output tape if the main computation ever entered into a halting state, else $\uparrow$ if it did not.

It is not hard to show that the rank of a computation is the ordinal height of its tree of sub-computations. For a freezing computation, i.e. one that remains paused however big α is taken to be, I do not (yet) have a good notion of a tree of sub-computations. For the eternally paused node, which trying to run, say, e' in parallel, it's paused because for each n' the trees $D^{(e',n')}_\beta$ remain paused, say at $e''_{n'}$. This could be viewed as countable branching from e', but of course this branching is different from that in $D^{(e,n)}$: in the latter tree, the branching shows the sequential computation, and the unsuccessful parallel runs are suppressed; from e', the branching represents all the parallel attempts. Of course, from $e''_{n'}$, the same story continues.

The problem with this notion is that it doesn't get us anything new. It is not hard to show that for a non-freezing computation the construction of the tree of sub-computations can be done in $L_{\omega_1^{CK}}$.

3 Non-deterministic Parallelism

Since choosing one canonical output to a parallel call didn't work out so well, let's go to other extreme and allow all possible answers. So when a computation makes an oracle call $\langle e\rangle(\cdot)$, an acceptable answer is *any* choice of n such that $\langle e\rangle(n)$ does not freeze. But wait a minute – since computations are non-deterministic, it could be that some runs of $\langle e\rangle(n)$ freeze and others do not, depending on how the oracle calls made while running $\langle e\rangle(n)$ turn out. So what does it mean to say "$\langle e\rangle(n)$ does not freeze?" We take that to be that *some* run of that computation does not freeze, if for no other reason than that is choice made in [1]. Since in the end we are not able to analyze this as we would like, perhaps it would have been better to say *all* runs do not freeze. Still, the question based on some run not freezing remains, and so we keep to that former notion, leaving the other for future work.

All of this can naturally be summarized in the *tree of runs*, defined below. This is not to be confused with the tree of sub-computations, so central in developing feedback. The tree of sub-computations summarized the sequential running of an algorithm, which can be viewed as traversing that tree, depth-first, from left to right. In contrast, the tree of runs captures the non-determinism. The splitting at a node is the many parallel runs of an oracle call. A single run of the algorithm is a path through the tree. There is no room in this tree for the sub-computations: if a node in the tree of runs represents $\langle e\rangle(n) = k$, the witness to that last computation is not contained in the tree, but rather must be found in the tree of runs for $\langle e\rangle(n)$.

3.1 The Tree of Runs

Definition 3.1 *The* **tree of runs** *is built from the root (thought of as being on the top) downwards, or, equivalently, as the computation proceeds, starting from the beginning, step 0. Each node has a start, meant to be the state of the computation when that node becomes active, and an end, meant as the state of the computation when the node becomes inactive. The start of the root is the program (e, n) being run. What the end of the root, or any other node for that matter, is, depends. If continuing the computation from the start of the node leads to an oracle call, say $\hat{e}$, then the end of the node is this $\hat{e}$; as need be, we may assume that the state of the computation at that point is also recorded in the node. If no such oracle call exists, then there are two possibilities. One is that after finitely many steps from the start of the node the computation has entered into a halting state. Then the end of the node is this halting state, and the content of the output tape is an output of the main computation. The other possibility is that the computation from the node's start never enters into a halting state, and so it diverges. Then the end of the node is this divergence, symbolically $\uparrow$, which is an output of the main computation.*

Nodes that end in a halting state or with divergence have no children. A node that ends with $\hat{e}$ may have children. For any natural number $\hat{n}$, and any output k of the computation $\langle\hat{e}\rangle(\hat{n})$, there is a child with start $(\hat{e}, \hat{n}, k)$, and which continues the computation of its parent with that start as the answer to the oracle call. Implicitly, and now explicitly, if there are no such $\hat{n}$ and k, then that node has no children, and the computation freezes there.

So a **run of a computation** is exactly a (maximal) path through its tree of runs. A finite output is given by a finite path, ending in a childless node in a halting state. A freezing computation is also given by a finite path, ending in a freezing node. A divergent computation can be given by a finite path, ending in $\uparrow$, and also by an infinite path.

The tree of sub-computations is absent from the tree of runs. It is hidden in the step from a node with end $\hat{e}$ to its children, or to its lack of children, which can be determined only by building $\hat{e}$'s own tree of runs. Of course, this latter tree might sub-contract out its own side-trees, and so on.

Because the semantics is given by a least fixed point, ordinal heights can be associated with these computations (when non-freezing). Ultimately, we will define the height of an output. But we must be careful here: because of the non-determinism, there could be wildly different ways to arrive at the same output. The simple solution to that would be to define the height of an output as the least ordinal among all the ordinals given by the different ways to get to that output. To do this right, one must define the height of a run of a computation, or, actually, the height of a *hereditary run.*

A **hereditary run** of a non-freezing computation is a run of that computation, along with an assignment, to each oracle call in the run (i.e. node in the run with end $\hat{e}$), with answer $(\hat{e}, \hat{n}, k)$ (i.e. the child in this run of that aforementioned node has start $(\hat{e}, \hat{n}, k)$), a hereditary run of $(\hat{e}, \hat{n})$ with output k.

The **height of a hereditary run** is defined inductively as the least ordinal greater than the heights of all of the sub-runs, meaning the hereditary runs assigned to oracle calls along the way.

The **height of a computation** $\langle e \rangle(n) = k$ is the smallest height of any hereditary run of such a computation. We will want to show that this is absolute among all transitive models.

Define $T_\alpha^{(e,n)}$, the sub-tree of the tree of runs of (e, n) which contains only those children of rank less than α, inductively on α.

For $\alpha = 0$, this tree contains only the root; if $\langle e \rangle(n)$ makes an oracle call then $T_0^{(e,n)}$ does not witness any output, else it witnesses either some finite k or $\uparrow$ as an output.

More generally, if $\beta < \alpha$, then $T_\beta^{(e,n)} \subseteq T_\alpha^{(e,n)}$. Furthermore, if a node in $T_\alpha^{(e,n)}$ ends with an oracle call $\hat{e}$, and there are $\beta < \alpha, \hat{n}$, and k (incl. $\uparrow$) such that $T_\beta^{(\hat{e},\hat{n})}$ witnesses that k is an output, then the child with start $(\hat{e}, \hat{n}, k)$ is in $T_\alpha^{(e,n)}$.

The outputs witnessed by $T_\alpha^{(e,n)}$ are the outputs of any terminal node (i.e. k if a node ends in a halting state with output k, or $\uparrow$ if a node ends with $\uparrow$), and also $\uparrow$ if $T_\alpha^{(e,n)}$ is ill-founded.

Notice that the height of $\langle e \rangle(n) = k$ is at most α iff $T_\alpha^{(e,n)}$ witnesses k as an output.

Proposition 3.2 *The height of $\langle e \rangle(n) = k$ is absolute among all transitive models.*

Proof: Inductively on α, the trees $T_\alpha^{(e,n)}$ and the outputs they witness are absolute. The outputs witnessed by terminal nodes are clearly absolute, individual nodes being finite, and for divergence, well-foundedness is absolute for well-founded models. ▌

3.2 Functions and Ordinal Notations

Ultimately we would like to characterize just what is parallel feedback computable. In the context of multi-valued functions, what this means should be clarified.

Definition 3.3 *A function f is* **parallel feedback computable (pfc)** *if there is an index e such that $\langle e\rangle(\cdot)$ is single valued and $\langle e\rangle(n) = f(n)$. A set is pfc if its characteristic function is.*

We would like to know what functions are pfc, and what relations are pfc.

While it should be no surprise that functions offer some benefits over relations, let's bring out a particular way that happens. Consider the index e which on any n returns both 0 and 1. (In more detail, let p be the parity function: $\{p\}(n)$ is 0 when n is even, 1 when odd. Let $\langle e\rangle(n)$ make a parallel call to p and return its output.) Notice that the characteristic function of any set at all is given by some run of e. So if you're non-deterministically searching for, say, the truth set of some L_α, there may well be a pfc function that gives you what you want, but you can't distinguish that from this e. And it does you no good to pick one non-deterministically, because if you pick e, when you go to use it again later, you might get different answers.

Since we expect that the analysis of this will involve computing initial segments of L, we might have need of notation for ordinals, which can be defined a la Kleene's $\mathcal{O}$. In honor of this history, and since the current subject is parallelism, we will call it $\mathcal{P}$. Because of the non-determinism present, there are several options for how this can be defined (in the limit case).

Definition 3.4 *Functional $\mathcal{P}$ ($f\mathcal{P}$) is defined inductively:*

- *$0 \in f\mathcal{P}$ and ord(0) = 0.*
- *If $a \in f\mathcal{P}$ then $2^a \in f\mathcal{P}$ and ord(2^a) = ord(a) + 1.*
- *If $\langle a\rangle(\cdot)$ is a function, and for all n we have $\langle a\rangle(n) \in f\mathcal{P}$, then $3 \cdot 5^a \in f\mathcal{P}$ and ord($3 \cdot 5^a$) $= \sup_n\{\mathrm{ord}\langle a\rangle(n)\}$.*

Definition 3.5 *Strict $\mathcal{P}$ ($s\mathcal{P}$) is defined inductively:*

- *$0 \in s\mathcal{P}$ and ord(0) = 0.*
- *If $a \in s\mathcal{P}$ then $2^a \in s\mathcal{P}$ and ord(2^a) = ord(a) + 1.*
- *If $\langle a\rangle(\cdot)$ is a total relation, and for all n and any possible output k_n of $\langle a\rangle(n)$ we have $k_n \in s\mathcal{P}$, and moreover ord(k_n) is independent of the choice of k_n (for a fixed n), then $3 \cdot 5^a \in s\mathcal{P}$ and ord($3 \cdot 5^a$) $= \sup_n\{\mathrm{ord}(k_n)\}$, where k_n is any output for $\langle a\rangle(n)$.*

Definition 3.6 *Loose $\mathcal{P}$ ($l\mathcal{P}$) is defined inductively:*

- *$0 \in l\mathcal{P}$ and ord(0) = 0.*
- *If $a \in l\mathcal{P}$ then $2^a \in l\mathcal{P}$ and ord(2^a) = ord(a) + 1.*
- *If $\langle a\rangle(\cdot)$ is a total relation, and for all n and any possible output k_n of $\langle a\rangle(n)$ we have $k_n \in l\mathcal{P}$, and $\sup_n\{\mathrm{ord}(k_n)\}$ is independent of the choice of k_n's, then $3 \cdot 5^a \in l\mathcal{P}$ and ord($3 \cdot 5^a$) $= \sup_n\{\mathrm{ord}(k_n)\}$, where k_n is any output for $\langle a\rangle(n)$.*

Clearly, $f\mathcal{P} \subseteq s\mathcal{P} \subseteq l\mathcal{P}$.

Proposition 3.7 *Every pfc well-ordering is isomorphic to one given by a functional ordinal notation.*

Proposition 3.8 *If X is pfc then $\mathcal{O}^X$ is pfc and ω_1^X has a functional ordinal notation.*

That $\mathcal{O}^X$ is pfc was proven in [1]. This is a slight extension of that argument.

Proposition 3.9 *If α has a loose ordinal notation then the Σ_1 truth set Tr_α of $L_{\omega_\alpha^{CK}}$ is pfc (where, as a function of α, ω_α^{CK} enumerates the closure of the set of admissible ordinals).*

Proof: Let $e \in \mathcal{P}$ be a fixed representation of α. By the recursion theorem, we can do this inductively on the ordinal height of $f <_{\mathcal{P}} e$.

If $f = 0$, then $Tr_f = \emptyset$.

If $f = 2^g$, then $Tr_f = \mathcal{O}^{Tr_g}$ from the previous proposition. (It is standard hyperarithmetic theory that $\mathcal{O}^X$ is Turing equivalent to the Σ_1 truth predicate of $L_{\omega_1^{CK}}$.)

If $f = 3 \cdot 5^g$, then the truth or falsity of any Σ_1 assertion ϕ in the limit structure can be determined as follows. Let n run through ω, and see whether ϕ is true according to each $Tr_{g(n)}$ in turn. If you ever find such an n making ϕ true, halt, else continue. Using feedback, ask whether that computation halts. If so, then ϕ is true in the limit structure; else ϕ is false there. ▌

Because of that last proposition, I bet the loose notations are ultimately the best, since they seem to capture the flavor of this kind of computation.

Proposition 3.10 *The characteristic function of $T_\alpha^{(e,n)}$ (along with the start and end of each node) is computable from a loose ordinal notation for α, as are the outputs witnessed by $T_\alpha^{(e,n)}$.*

With a bit of work, this could be presented as a corollary of the previous proposition, since $T_\alpha^{(e,n)}$ is definable over $L_{\omega_\alpha^{CK}}$.

Proof: By a simultaneous induction on ordinal notations.

The only notation for the ordinal 0 is 0. To compute $T_0^{(e,n)}$, one first asks the oracle whether computing $\langle e \rangle(n)$ will ever lead to an oracle call. If so, one runs $\langle e \rangle(n)$ until that call, which becomes the end of the root, and then stops. If not, one asks the oracle whether computing $\langle e \rangle(n)$ will ever halt. If so, one runs it until it halts; if not, then the output is $\uparrow$.

Consider the ordinal notation $a = 2^b$ for $\alpha = \beta + 1$. Of course, the root of $T_\alpha^{(e,n)}$ is computable, as above. For any node in $T_\alpha^{(e,n)}$, to see whether a child is in $T_\alpha^{(e,n)}$, we may assume the node ends with $\hat{e}$. A child starting with $(\hat{e}, \hat{n}, k)$ is in $T_\alpha^{(e,n)}$ iff $T_\beta^{(\hat{e},\hat{n})}$ witnesses that k is an output, which inductively is computable from b. The end of such a node is deterministic in the start. To compute whether k is witnessed to be an output, one can use the oracle to see whether the search through $T_\alpha^{(e,n)}$ for a terminal node with output k will halt. In addition, when $k = \uparrow$, check whether $T_\alpha^{(e,n)}$ is well-founded, which is computable in its hyperjump (cf. the penultimate proposition).

Now consider the ordinal notation $a = 3 \cdot 5^b$. We must decide membership in $T_\alpha^{(e,n)}$ of children of nodes ending in $\hat{e}$. For the child starting with $(\hat{e}, \hat{n}, k)$, use the oracle to see whether the search for an i such that, with $\beta_i = \mathrm{ord}(b(i))$, the tree $T_{\beta_i}^{(\hat{e},\hat{n})}$ witnesses that k is an output, halts. The determination of $b(i)$ is, of course, non-deterministic, as is the value β_i, but as β_i is guaranteed to be cofinal in α, this makes no difference. The computation of the outputs witnessed is as above. ▮

The hope is that the structure just identified will help in determining the pfc functions and relations, which we have not been able to do. Although the next section is dedicated to the study of a different kind of computation for its own sake, it also provides at least a coarse upper bound for those studied here.

4 Context-Dependent Determinism

4.1 Semantics

The problem of the first alternative offered is that it's too restrictive, and so gives you nothing new. The problem with the second is that it's too liberal, allowing for multi-valuedness, and so we couldn't analyze it. This time we're going for something in the middle. Any oracle call will return at most one value, but possibly a different value every time it's called.

The semantics begins just as in the non-deterministic case. Trees $C_\alpha^{(e,n)}$ (C for context) are defined inductively on α. The new intuition here is that these trees are built until an output is seen, and that first output is taken as the value of $\langle e \rangle(n)$. More precisely, $C_\alpha^{(e,n)}$ yields an output if it contains a halting node (with some integer output k) or a diverging node (with output $\uparrow$), or is ill-founded (with output $\uparrow$). Let α be the least ordinal such that $C_\alpha^{(e,n)}$ yields an output. If it yields more than one output, pick the left-most one. That is, starting at the root, traverse the tree downwards. Every non-terminal node ends with an oracle call $\hat{e}$. The child to be followed has start $(\hat{e}, \hat{n}, k)$, where $\hat{n}$ is the least natural number such that the tree beneath that node yields an outcome (and k is the value of $\langle \hat{e} \rangle(\hat{n})$).

As an example of this semantics in practice, the earlier proof that $\mathcal{O}^X$ is pfc from X [1] still works. The way that construction goes, given a non-well-founded order, and an n in the non-standard part, if k is in the standard part, it won't be chosen as a successor step after n, because that will definitely lead to a freezing state. Only a non-standard k (less than n in this ordering) will be chosen, and in fact that least such k in the natural ordering of ω will be.

4.2 Lemmas

Lemma 4.1 *There is a program which, on input e, diverges if e computes (the characteristic function of) the truth set of a model of some computable theory T, and freezes otherwise.*

We assume here some standard coding of syntax into arithmetic. The model can be taken to be a structure on, say, the odd integers, so that the even integers can be used for the symbols of the language, and formulas with parameters can be considered. Of course, this program can easily be converted into one that halts instead of diverges: ask the oracle about this program, and if the answer comes back "divergent," then halt. It will be easy to see that in some instances it can be recognized that e does not compute such a set, and our program could return that instead of diverging; but if, say, $\{e\}(0)$ freezes, then any such program as ours would have to freeze, and there seemed to be no benefit in a program that sometimes recognizes when e is not as desired and sometimes freezes.

Proof: It is feedback Turing computable to dovetail the generation of T, the computations of $\langle e\rangle(n)$ for all n, and the check that that latter theory is complete, consistent, and contains T. If e computes such a model, this procedure will never end; if e finds some violation, the procedure can be taken to freeze. If some $\langle e\rangle(n)$ freezes, the procedure will necessarily freeze. ∎

We will be using this to see if e codes a model of $V = L_\alpha$. We do not sharply distinguish between the Σ_1 truth set of some L_α and the full truth set, since this computational paradigm can easily shuttle between them.

Lemma 4.2 *There is a program such that, if e computes a partial order on a subset of ω, on input e it will return 0 if e's order is well-founded and 1 if ill-founded.*

Proof: This is a lot like the proof of the computability of $\mathcal{O}$.

For pre-processing, check whether the domain of e is finite. If so, you have your answer. Else, continue.

First we check for well-foundedness. Go through the natural numbers, and for each such n, if n has no e-predecessors (determined by an oracle call), halt, else run this same procedure, via the fixed-point or recursion theorem, on the same order restricted to those elements e-less than n. In the tree of sub-computations, the children of a node given by n are exactly the e-predecessors of n. So this tree is well-founded iff $<_e$ is well-founded. So this procedure diverges iff $<_e$ is well-founded, else it freezes.

To check for ill-foundedness, run in parallel the following procedure on each $n \in \omega$. If n has no predecessor, freeze. Else, by the fixed point theorem, run this same procedure on the same order restricted to those elements e-less than n. In the tree of runs, the children of a node given by n are exactly the e-predecessors of n. So this tree is well-founded iff $<_e$ is well-founded. Since the terminal nodes all freeze, the only possible non-freezing semantics is an infinite descending path, which exists exactly when $<_e$ is ill-founded.

Now run both of those checks in parallel. Whichever one does not freeze is what tells you whether $<_e$ is well- or ill-founded. ▮

Just as before, it is easy to see that what can be computed is exactly some initial segment of L. We will shortly see just what this initial segment is. Before that, we will prove some lemmas which handle some simpler cases, partly to get the reader (and author!) used to the kind of arguments employed, and partly so in the main theorem we can ignore some of the cases of weaker, messier ordinals, and focus on just the more strongly closed ones.

Lemma 4.3 *The supremum α of the computable ordinals is admissible.*

Proof: Suppose not. Let $f : \omega \to \alpha$ witness α's inadmissibility. For each n, using the previous lemmas, one can check whether $\langle e \rangle$ codes a model of "$V = L_\gamma$ is the least admissible set in which $f(n)$ is defined," and if so whether the model so coded is well-founded. On many inputs this will freeze, but since by hypothesis α is the least non-computable ordinal, there is at least one e_n on which this halts (possibly more, allowing for some flexibility in the coding). By making a parallel call of all natural numbers, one can produce such an e_n.

To see whether a Σ_1 formulas ϕ is in the Σ_1 truth set for L_α, consider the procedure which runs through each n, finds a truth set for $f(n)$ as above, and stops whenever ϕ shows up as true in one of those sets. Now ask the oracle whether that procedure halts. If so ϕ is true in L_α, else not. ▮

Lemma 4.4 *α is greater than the least recursively inaccessible.*

Proof:

The following procedure will generate the Σ_1 truth set of the first recursively inaccessible.

Start with (a code for) the truth set of $L_{\omega_1^{CK}}$. We will describe a procedure which pieces larger and larger initial segments of L together, which diverges (continues indefinitely) as long as it's still working on the first inaccessible, and which freezes whenever it finds a contradiction in what it has done so far.

At any stage along the way, there will be a well-founded model of $V = L_\gamma$, as well as a finite set of Π_1 sentences the procedure is committed to making true. As soon as the model at hand falsifies one of those sentences, then the procedure freezes, because it sees that the jig is up.

Dovetail consideration of all countably many Σ_1 formulas $\phi(x, y, \vec{z})$ and all countably many sets A and tuples $\vec{b}$ that show up in the models produced in this construction. At stage n we are considering a certain ϕ, A, and $\vec{b}$, and will decide whether we think $\forall a \in A \; \exists y \; \phi(a, y, \vec{b})$ is true or false in the first recursively inaccessible. In parallel, choose either true or false. Moreover, if you choose true, then you must provide a well-founded model of $V = L_\gamma$ extending the previously chosen model by at least one admissible, in which the chosen formula with parameters is true, and which also models there is no recursively inaccessible. If you choose false, then you must also choose a specific $a \in A$, and include in the set of sentences "$\forall y\ \neg\phi(a, y, \vec{b})$".

Since this construction has no halting condition, the only way it can not freeze is if it diverges. It cannot diverge by always making the chosen formula false, if for no other reason than there are infinitely many total Σ_1 functions in the starting model, and they cannot consistently be made partial. So infinitely often the model under consideration will be extended by at least one admissible. Hence the limit model will be an initial segment of L which is a limit of admissibles. Let ϕ be Σ_1 and $A, \vec{b}$ be in the limit model. Suppose it's true in this model that $\forall a \in A\ \exists y\ \phi(a, y, \vec{b})$. When that formula came under consideration, it could not have been deemed false, because then we would have committed ourselves to a specific counter-example, and that counter-example would have been seen to be invalid at some point, leading to a freezing computation. So the formula was deemed to be true. Hence a model was picked in which the induced relation was total, thereby providing a bound on the range. Hence the limit model is admissible. Since it's a limit of models of "there is no recursively inaccessible," it is itself the least recursively inaccessible.

We have just argued that any divergent run of this program produces the least recursively inaccessible. Furthermore, there are divergent runs, by always choosing whatever is in fact true of that ordinal. ▮

4.3 Main Theorems

Definition 4.5 *Let Γ be a collection of formulas, X a class of ordinals, and ν^{+X} the least member of X greater than ν. We say that α is* **Γ-reflecting on** *X if, for all $\phi \in \Gamma$, if $L_{\alpha^{+X}} \models \phi(\alpha)$, then for some $\beta < \alpha$, $L_{\beta^{+X}} \models \phi(\beta)$.*

We are interested in the case $\Gamma = \Pi_1$ and $X =$ the collection of admissible ordinals. For this choice of X, we abbreviate ν^{+X} by ν^+, which is standard notation for the next admissible anyway. This is called Π_1 **gap-reflection on admissibles**. Let γ be the least such ordinal.

It may seem like a strange notion. I will not dispute that. But this is not the first time it has come up. Extending work in [4], it was shown in [3] that such ordinals are exactly the Σ^1_1 reflecting ordinals. The reason this topic came up in the latter paper is that a particular case of its main theorem is that γ is the closure point of Σ_2-definable sets of integers in the μ-calculus. (The μ-calculus is first-order logic augmented with least and greatest fixed-point operators. In this context, Σ_2 refers to the complexity of the fixed points in the formula, namely, in normal form, a least fixed point in front, followed by a greatest fixed point, followed by a fixed-point-free matrix.) So this definition has been used (at least) twice before already. With a third use here, one might well think it has other applications too.

Theorem 4.6 *The ordinals so computable are exactly those less than γ.*

So there is an intimate connection between parallel feedback computability and Σ_2 definability in the μ-calculus. This was not expected. In the simpler case of feedback Turing computability [1], it was really no surprise that it turned out to be the same as hyperarithmeticity, as both are essentially joining well-foundedness to computation. But we have no intuition, even after-the-fact, in support of the current result.

Proof: We will argue that no computation $\langle e \rangle(n)$ can be witnessed to converge or diverge from stage γ onwards. Notice that for any $\gamma' > \gamma$, if $T^{(e,n)}_{\gamma'}$ is different from $T^{(e,n)}_{\gamma}$, that can only be because some other computation $\langle e' \rangle(n')$ was seen to converge or diverge at some stage at least γ and less than γ'. Tracing back the computation of $\langle e' \rangle(n')$, we are eventually led to a computation that was seen to converge or diverge at exactly stage γ. Since γ is a limit ordinal, there are no new terminal nodes on any tree of runs at stage γ. Hence there is some computation $\langle e \rangle(n)$ such that $T^{(e,n)}_{\gamma}$ is ill-founded, but $T^{(e,n)}_{\beta}$ is well-founded for any $\beta < \gamma$. How could the ill-foundedness of $T^{(e,n)}_{\gamma}$ be most economically expressed? Since γ is the γ^{th} admissible ordinal, $T^{(e,n)}_{\gamma}$ is definable over L_γ. It is a basic result of admissibility theory that a tree in an admissible set is well-founded iff there is a rank function from the tree to the ordinals in that very same admissible set. So the ill-foundedness of such a tree is witnessed by the non-existence of such a function in any admissible set containing the tree. In the case at hand, that is a Π_1 statement in $L_{\gamma+}$ with parameter γ. By the choice of γ, this reflects down to some smaller β. So $T^{(e,n)}_{\beta}$, for some smaller β, was already seen to be ill-founded. So there can be no new computation values at stage γ, and hence not beyond either.

For the converse, let β be strictly less than γ; by lemma 4.3, we can assume that β is a limit of admissibles. Assume inductively that for each $\alpha < \beta$ there is an e such that $\langle e \rangle(\cdot)$ is the characteristic function of the Σ_1 truth set of L_α. Let ϕ witness that β is not Π_1 gap-reflecting on admissibles: so ϕ is Π_1, and $L_{\beta+} \models \phi(\beta)$, but if $\alpha < \beta$ then $L_{\alpha+} \not\models \phi(\alpha)$. We must show that (the characteristic function of) the Σ_1 truth set of L_β is computable.

As in lemma 4.4, start with (a code for the Σ_1 truth set of) $L_{\omega_1^{CK}}$. At any stage along the way, there will be a well-founded model of $V = L_\alpha$, as well as two finite sets (both empty, at the beginning) of sentences. The intent of this construction is that, if it continues for ω-many steps, the union of the L_α's so chosen will be L_β, all of the sentences in the first set will be true in L_β, and the second set will provide a term model of $V = L_{\beta+}$.

The action at any stage is much as in the previous lemma. First, check for the consistency of a theory, to be described below. If an inconsistency is found, freeze. Else we are going to continue building the ultimate model. This involves interleaving steps to make sure that the union of the chosen L_α's, L_δ, is admissible (and $\delta \leq \beta$), with steps to insure that $L_{\delta^+} \models \phi(\delta)$ (guaranteeing $\delta \geq \beta$). We assume a dovetailing, fixed at the beginning, of all (countably many) formulas ψ with parameters. For the formulas in the first set, the parameters are the sets in the L_α's chosen along the way. For the formulas in the second set, the parameters include, in addition to the members of the L_α's, also constants c_i for the term model, as well as a dedicated constant we will ambiguously call δ, since the ordinal δ is its intended interpretation.

At any even stage $2n$, consider the n^{th} formula of the form $\forall a \in A\ \exists y\ \psi(a, y, \vec{b})$, where ψ is Σ_1 and the parameters are from the L_α at hand. In parallel, choose it to be either true or false. Moreover, you must provide a well-founded model of $V = L_\alpha$, extending the previously chosen model by at least one admissible. Furthermore, if you had deemed the formula to be true, then it must hold in the chosen L_α; if false, then you must also choose a specific $a \in A$, and include in the first set of sentences "$\forall y\ \neg\psi(a, y, \vec{b})$". Notice that this step includes as a degenerate case those instances in which ψ does not depend on a, thereby forcing us to decide all Σ_1 and Π_1 formulas. Finally, it must be the case that $\alpha < \beta$, which can be verified computably, since it needs only a well founded model of $V = L_{\alpha^+}$ (which exists by the inductive hypothesis and the choice of β) which also satisfies "$\neg\phi(\alpha) \wedge \forall \nu < \alpha\ L_{\nu^+} \not\models \phi(\nu)$".

At an odd stage $2n+1$, consider similar to the above the n^{th} formula of the form $\forall a \in A\ \exists y\ \psi(a, y, \vec{b})$, where ψ is Σ_1, only this time the parameters are for the second set (that means the parameters are from an already chosen L_α and the c_i's and δ). Include in the second set either "$\forall a \in A\ \exists y \in \tau\ \psi(a, y, \vec{b})$", for some term τ, or "$\tau \in A \wedge \forall y\ \neg\psi(\tau, y, \vec{b})$," for some term τ. Of course, this step is meant to include all possible degenerate cases, such as Σ_1 assertions, even quantifier-free sentences. Also, if "$\tau < \delta$" for some term τ is ever included in the second set, then, extending L_α if need be, for some $\epsilon < \alpha$ the sentence "$\tau = \epsilon$" is included in the second set.

With regard to the theory referenced above but there left unspecified, at any stage along the way it will be "$V = L_{\delta^+}$ is admissible, and δ is admissible, and $\alpha < \delta$ (where L_α is the model we have at this stage), and everything in the first set is true in L_δ, and everything in the second set is true in V."

For this computation, the tree of runs has neither halting nor divergence nodes (since, whenever it does not freeze, it makes another oracle call). It is ill-founded, since there is a run of the computation which does not halt, namely one using the truth about L_β and L_{β^+} to make decisions along the way. We would like to show that along any infinite path in the tree of runs, the induced δ equals β.

Consider the term model induced by the second set. There is an isomorphism between the term δ and the union of the α's chosen along the way: on the one hand, the assertion "$\alpha < \delta$" was included in the theory along the way, and on the other, anything ever deemed less than δ was forced to be less than some α. So we can consider the term model as including some (standard) ordinal δ. Also, this δ is at most β, since each α is less than β. The next observation is that this term model satisfies "$V = L_{\delta^+}$ is admissible," by the Henkinization (choice of explicit witnesses) performed on the second set. Of course, the term model might well be ill-founded. But its well-founded part has ordinal height the real δ^+. By the downward persistence of Π_1 sentences, since $\phi(\delta)$ holds in the term model, it holds in the actual $L_{\delta+}$. By the choice of ϕ, δ is at least as big as β.

We must turn this procedure into a way of getting the characteristic function for the truth set of L_β. For any Σ_1 sentence χ, run the procedure as above, with χ and $\neg\chi$ each separately, in parallel, included in the first set. The false option is inconsistent and so any such computation will freeze, so the answer you will get is the true option, along with the information that the procedure diverges. ▮

Corollary 4.7 *For $\beta < \gamma$, the order-types of the $\Sigma_1(L_\beta)$-definable well-orderings of ω are the ordinals less than β^+.*

This is a generalization of the earlier result that the order-types of the Π^1_1 well-orderings are cofinal in ω_2^{CK}. Sacks [6], giving this special case as an exercise (p. 51, 7.10), attributes it to Richard Platek, who never published a proof. Although Platek may have been the first to notice this (Sacks in personal correspondence dates it from the '60s), Tanaka [7] seems to have discovered it independently.

The corollary as stated is not the optimal result, since the conclusion holds for any β which is Σ_1 projectible, by arguments similar to Tanaka's. It's just that this more general result is no longer a corollary to the theorem.

Proof: For simplicity, assume that β is a limit of admissibles. The construction of the theorem is of an ill-founded tree T_β, Σ_1 definable over L_β, such that any infinite path yields a term model of $V = L$ with ordinal standard part β^+. If the well-founded nodes all had rank less than some $\beta' < \beta^+$, then they could all be distinguished from the non-well-founded nodes definably over $L_{\beta'}$. So an infinite path, and hence such a term model, is also definable over $L_{\beta'}$. It is then easy (which we can here take to mean "definable over $L_{\beta'}$") to read off all the reals in this model. This includes reals with L-rank cofinal in β^+. This is a contradiction. Hence, for any $\beta' < \beta$, there is a node in T_β with that rank. The nodes of T_β are labeled with pairs (e, n). They also have associated with them two finite sets of formulas. The formulas are just finite pieces of syntax, except for the parameters from L_β's. But L_β is the Σ_1 Skolem hull of ω, which provides an integer name for each of its members (for instance, a Σ_1 formula that it uniquely satisfies). So each formula can be coded by a natural number. All told, each node can be represented by a natural number. This produces an ordering of a subset of ω with rank β'. To get this to be a well-ordering, it suffices to take the Kleene-Brouwer ordering of that tree. ▮

Happily, the work done also enables us to determine at least an upper bound for the non-deterministic computations.

Theorem 4.8 *Any relation computable via a non-deterministic parallel feedback Turing machine, as in the previous section, is* $\Sigma_1(L_\gamma)$.

Proof: By much the same argument as before. The only possible values come from halting nodes, divergent nodes, and the ill-foundedness of trees. A node is seen to halt at a successor stage, and γ is not a successor ordinal. A node is seen to diverge at a stage of the form $\alpha + \omega$, and γ is not of that form. As for the last possibility, the tree $T_\gamma^{(e,n)}$ is Δ_1 definable in L_{γ^+} with parameter γ. If it's not well-founded, that fact is Π_1 expressible in L_{γ^+}. By the choice of γ, a smaller $T_\alpha^{(e,n)}$ was already ill-founded, so divergence was already a value for $\langle e\rangle(n)$. Hence there are no new possible values for any computation at or after stage γ. ▮

References

[1] Nathanael Ackerman, Cameron Freer, and Robert Lubarsky, "Feedback Turing Computability, and Turing Computability as Feedback," Proceedings of LICS 2015, Kyoto, Japan; also available at http://math.fau.edu/lubarsky/pubs.html

[2] Robert Lubarsky, "ITTMs with Feedback," in **Ways of Proof Theory** (Ralf Schindler, ed.), Ontos, 2010, pp. 341-354; also available at http://math.fau.edu/lubarsky/pubs.html

[3] Robert Lubarsky, "μ-definable Set of Integers," **Journal of Symbolic Logic**, v. 58 (1), March 1993, pp. 291 - 313

[4] Wayne Richter and Peter Aczel, "Inductive Definitions and Reflecting Properties of Admissible Ordinals," in **Generalized Recursion Theory** (Fenstad and Hinman, eds.), North-Holland, 1974, pp. 301-381

[5] Hartley Rogers, **Theory of Recursive Functions and Effective Computability**, McGraw-Hill, 1967

[6] Gerald Sacks, **Higher Recursion Theory**, Springer, 1990

[7] Hisao Tanaka, "On Analytic Well-Orderings," **Journal of Symbolic Logic**, v. 35 (2), June 1970, pp. 198 - 204