Feedback Hyperjump

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

October 19, 2019

Abstract

Feedback is oracle computability when the oracle consists exactly of the con- and divergence information about computability relative to that same oracle. Here we study the feedback version of the hyperjump.
keywords: hyperjump, feedback, Kleene's $\mathcal{O}$
AMS 2010 MSC: 03D60,03D70

1 Introduction

Imagine a notion of computability which allows for an oracle. A natural choice of oracle is the halting problem, the set of halting programs. What if the programs in the oracle were exactly the halting programs relative to that same oracle?

That is the essence of feedback. There is not (yet) a general definition of feedback, one which is based on an unspecified notion of computation, perhaps a notion with some properties given axiomatically. What we do have are particular examples of feedback, including feedback Turing machines [1,2], feedback infinite time Turing machines [8], and feedback primitive recursion [1, 2]. Experience has shown that the way feedback is defined has to be adapted to each different setting. Hence it is useful and interesting to examine more instances of feedback. The purpose of this work is to introduce feedback hyperjump.

There are several aspects of this that could be of interest. There are naturally not one but two different kinds of feedback for the hyperjump. Called below strict and loose, the difference between them is that the loose version has a kind of built-in parallelism. If there were a general definition of feedback then there should be only one kind of feedback hyperjump, but as it turns out we see no prima facie reason to choose one of strict and loose over the other. Another aspect that bears mention is that, in both cases, the central concept is arguably that of well-foundedness, but it plays a two-faced role. In some more detail, with feedback the escape from paradox threatened by diagonalization is provided by the possibility of computations freezing. For instance, if a computation asks the

oracle about itself, then, in deciding what to answer, that same computation must be run, which will eventually ask the oracle about itself, ad infinitum. More generally, if a computation involves an infinite nested chain of oracle calls, then the oracle (depending on the setting) may not have a good answer and the computation could freeze. So the ill-foundedness of the tree of oracle calls (typically) leads to the computation freezing. On the other hand, considering $\mathcal{O}$ as a simple example of a hyperjump (the hyperjump of $\emptyset$), membership of n in $\mathcal{O}$ is given by the well-foundedness of the tree of ordinal notations less than n (less than in the sense of $<_{\mathcal{O}}$). Conversely, non-membership in $\mathcal{O}$ (in the interesting cases) is witnessed by the ill-foundedness of the induced tree of potential ordinal notations. With feedback, this is just the kind of information we want to capture. So ill-foundedness here will give us positive information. In the end, we will need to distinguish between two different kinds of trees, the tree of sub-computations and the tree of ordinal notations, the well- or ill-foundedness of each having very different consequences. Actually, this description is clean only for strict feedback hyperjump; for the loose version, as a kind of parallelism, even an ill-founded subcomputation tree can lead to a non-freezing computation. Ultimately the point for the moment is that we will be taking a very close look at the well-foundedness of trees associated with these computations. Finally, the results themselves might be of interest, as providing alternate descriptions of some ordinals which have already appeared in the literature.

To simplify the exposition, instead of defining the feedback hyperjump of an arbitrary real X, this will be done for only the empty set; the relativization to an arbitrary X is straightforward. The next section will review some of the basics of the regular hyperjump of 0, a.k.a. $\mathcal{O}$; all of this material is standard for the field, and serves only as a refresher and to introduce some of the notation and terminology we will use. The sections after that will study strict and loose feedback hyperjumps respectively.

To provide some historical context, feedback was clearly identified in [12] (pp. 406-407), although the topic was not pursued at that time. It was re-introduced in [8]. For an overview, see [2].

It should be mentioned that, as of this writing, not all of the proofs are complete. That notwithstanding, the author believes that they are far enough along to be convincing, and that the notions introduced are interesting enough to warrant public exposition.

2 Background on $\mathcal{O}$

For a much more thorough introduction to admissibility, $\mathcal{O}$, and such like, we refer the reader to [4] or [13]. Recall the mutual inductive definitions of Kleene's $\mathcal{O}$ and of the partial order $<_{\mathcal{O}}$ on $\mathcal{O}$. Regarding the former, $\mathcal{O}$ is the least set such that

1. $1 \in \mathcal{O}$,
2. if $n \in \mathcal{O}$ then $2^n \in \mathcal{O}$, and

3. if $\{e\}$ is total, and $\forall n\ \{e\}(n) <_{\mathcal{O}} \{e\}(n+1)$, then $3 \cdot 5^e \in \mathcal{O}$.

For the latter, $<_{\mathcal{O}}$ is the least transitive relation on $\mathcal{O}$ such that $n <_{\mathcal{O}} 2^n$ and $\{e\}(n) <_{\mathcal{O}} 3 \cdot 5^e$. (Notice that this is finer than the ordering on the ordinals represented by the members of $\mathcal{O}$.) Numbers not in $\mathcal{O}$ are incomparable with everything.

Given $n \in \mathcal{O}$, $\{k \mid k <_{\mathcal{O}} n\}$ is naturally ordered as a tree, with root n. The children of n are given by the primitive relations just mentioned, i.e. n is the unique child of 2^n, and if $n = 3 \cdot 5^e$ then the numbers $\{e\}(k)$ are the children of n. This tree is well-founded. In fact, that essentially characterizes the members of $\mathcal{O}$. That is, every $n \in \mathbb{N}$, whether in $\mathcal{O}$ or not, induces such a tree T_n, defined recursively, the well-foundedness of which, or not, determines membership in $\mathcal{O}$. We think of T_n, and related trees to be defined later, as the tree of ordinal notations, although it would be more accurate to call it the tree of potential ordinal notations, since some entries may not actually be ordinal notations (making of course n also not an ordinal notation); the latter name being more cumbersome, we stick with the former.

Definition 1. *1. T_1 is the tree consisting of the single node 1.*

2. For $n \neq 0$, T_{2^n} has root 2^n, which has a unique child n, which is the root of the subtree T_n.

3. $T_{3\cdot 5^e}$ has root $3 \cdot 5^e$, with children each $\{e\}(k)$ which is defined, which is the root of the subtree $T_{\{e\}(k)}$.

4. In all other cases, T_n consists of the single node n.

We say that the tree T_n is **ill-formed** *if*

1. either T_n contains a node not of the form 2^m or $3 \cdot 5^e$,

2. or T_n contains a node $3 \cdot 5^e$, and either $\{e\}$ is partial, or, for some k, $\{e\}(k) \notin T_{\{e\}(k+1)}$.

If T_n is not ill-formed then we say it is **well-formed**.

Proposition 1. *$n \in \mathcal{O}$ iff T_n is well-formed and well-founded.*

It is easy to see that there are n's with T_n well-formed and ill-founded: work in some non-standard model of some kind of set theory (say KP or anything stronger) with standard part ω_1^{CK}, and let n be a notation (as interpreted in that model) for some non-standard ordinal.

3 Strict Feedback Hyperjump

We will ultimately define the feedback oracle $\mathcal{SO}$, or **strict feedback** $\mathcal{O}$. With regular (as opposed to feedback) oracle computation, an oracle can be taken to be a set, and an oracle query returns YES or NO depending upon whether the number queried is in the oracle or not. With feedback oracle computation, this

will not work. One cannot avoid freezing, the possibility that the oracle just doesn't answer; this is how diagonalization is avoided. So a feedback oracle is taken to be a partial function, which on its domain returns either Y or N. A query which is not in the domain of an oracle is said to be a freezing query (relative to that oracle); if during the course of a computation the oracle is asked a freezing oracle query, it does not answer, and that computation freezes. In addition, our oracle will have to answer two different types of questions: not only "$n \in \mathcal{SO}$?", but also "$m <_{\mathcal{SO}} n$?". So a **feedback oracle** is a partial function from the set of queries of the form "$n \in \mathcal{SO}$?" and "$m <_{\mathcal{SO}} n$?" to $\{Y, N\}$.

Let P be a feedback oracle. By way of notation, $\{e\}^P$ is Turing computability relative to P.

The trees T_n defined in the previous section are sufficient to witness membership in $\mathcal{O}$; more crucially, they are necessary to witness non-membership in $\mathcal{O}$. Since non-membership in $\mathcal{SO}$ needs to be witnessed positively, we have need of the analogue U_n of T_n, also called the tree of ordinal notations, appropriate for the current setting. The definition of U_n is identical to that of T_n, except that the computations involved are feedback computations, with notation $\langle e \rangle$ and $\langle e \rangle(k)$, depending on whether the program e calls for an input. (Whether angle brackets $\langle\rangle$ are meant as feedback computation, as in $\langle e \rangle(k)$, or as forming a tuple, as in the ordered pair $\langle a, b \rangle$, should be clear from the context. Typically one argument $\langle e \rangle$ means feedback, and more than one a tuple.) Since we are in the midst of defining these very computations, in order to avoid circularity we must first define U_n^P, where P is a feedback oracle. We will then use this to define a one-step procedure from the set of feedback oracles to itself, and observe that this procedure is positive in P (i.e. $P \subseteq Q$ implies $U_n^P \subseteq U_n^Q$). Then on general principles there will be a least fixed point of that procedure, which we will call $\mathcal{SO}$. With $\mathcal{SO}$ in hand, the ultimate tree of interest U_n can be taken to be $U_n^{\mathcal{SO}}$.

Terminology. When building a tree T_a recursively from a parameter a, we will typically give the children b of the root, and then want to continue defining the descendants of b in T_a as essentially the members of T_b. We will give some precise definitions here, to fall back on as need be, although we may abuse notation for convenience (for instance sometimes identifying a piece of code, when convergent, with its output, or identifying a tuple of length 1 with its only entry). A tree is a set of tuples of natural numbers of positive length, closed under truncation. The label of a node of a tree is the last entry of the node as a tuple. For instance, the tree T_a will have root $\langle a \rangle$, which is labeled a. If σ is a node in T, **to append a tree** U **beneath** σ **in** T means to include in T all tuples of the form $\sigma^\frown\tau$, where τ is a node in U (and $^\frown$ is concatenation).

Definition 2. *1. U_1^P is the tree consisting of only the root, labeled 1.*

2. For $m \neq 0$, $U_{2^m}^P$ has root labeled 2^m, which has a unique child labeled m, and U_m^P is appended to $U_{2^m}^P$ beneath the root.

3. $U_{3\cdot 5^e}^P$ has root labeled $3 \cdot 5^e$, with a child labeled by the pair $\langle e, k \rangle$ for each

$k \in \mathbb{N}$[1]; if $\{e\}^P(k)$ is defined, then append $U^P_{\{e\}^P(k)}$ beneath the root, by abuse of notation identifying the label $\langle e, k\rangle$ with the label $\{e\}^P(k)$ of the root of $U^P_{\{e\}^P(k)}$; if $\{e\}^P(k)$ is not defined (either freezing or divergent) then the node labeled $\langle e, k\rangle$ has no children.

4. *In all other cases, U^P_n consists of a single node labeled n.*

We say that U^P_n **freezes** *if there is a node in U^P_n labeled $\langle e, k\rangle$ such that $\{e\}^P(k)$ freezes.*

U^P_n is **ill-formed** *if*

1. *either U^P_n contains a node not of the form 2^m or $3 \cdot 5^e$,*
2. *or U^P_n contains a node $3 \cdot 5^e$, and either $\{e\}^P$ is partial, or, for some k, the oracle call "$\{e\}^P(k) <_P \{e\}^P(k+1)$?" returns N.*

If U^P_n does not freeze and is not ill-formed then we say it is **well-formed**.

Proposition 2. *If U^P_n does not freeze and $Q \supseteq P$, then $U^Q_n = U^P_n$.*

Viewing a feedback oracle P as giving partial information about a fixed point feedback oracle, P induces its own version of answers to oracle queries, which we want to view as the successor feedback oracle to P, hence the notation P^+.

Definition 3.
1. *$P^+(n \in \mathcal{SO}?) = Y$ if U^P_n is well-formed and well-founded.*
2. *$P^+(n \in \mathcal{SO}?) = N$ if U^P_n either is not freezing and ill-formed, or is well-formed and ill-founded.*
3. *$P^+(m <_{\mathcal{SO}} n?) = Y$ if $P^+(n \in \mathcal{SO}?) = Y$ and $m \neq n$ is a node in U^P_n.*
4. *$P^+(m <_{\mathcal{SO}} n?) = N$ if $P^+(n \in \mathcal{SO}?) = N$, or if $P^+(n \in \mathcal{SO}?) = Y$ and either $m = n$ or m is not a node in U^P_n.*

Proposition 3. *$P^+(m <_{\mathcal{SO}} n?)$ returns a value iff $P^+(n \in \mathcal{SO}?)$ returns a value. Also, $P^+(n \in \mathcal{SO}?)$ returns a value iff U^P_n does not freeze.*

Proposition 4. *The definition of P^+ is positive in P. Hence if $P \subseteq Q$ then $P^+ \subseteq Q^+$.*

The preceding proposition justifies the following.

Definition 4. *$\mathcal{SO}$ or* **strict** *$\mathcal{O}$ is the least fixed point of the operation $P \mapsto P^+$. U_n is $U^{\mathcal{SO}}_n$.*

Sometimes we think of $\mathcal{SO}$ not as a feedback oracle but rather as a partial set of numbers, as captured by the following convention; which way to think about $\mathcal{SO}$ should always be clear from the context.

Notation:

[1]This child is to be thought of as a piece of syntax acting as a placeholder, and not, for instance, as feedback computation, for which angle brackets $\langle\rangle$ are also used.

- "$n \in \mathcal{SO}$" is an abbreviation for $\mathcal{SO}(n \in \mathcal{SO}?) = Y$.
- "$n \notin \mathcal{SO}$" is an abbreviation for $\mathcal{SO}(n \in \mathcal{SO}?) = N$.
- "$n \,?\, \mathcal{SO}$" is an abbreviation for $n \in \mathcal{SO}?$ not being in the domain of $\mathcal{SO}$.
- "$m <_{\mathcal{SO}} n$" is an abbreviation for $\mathcal{SO}(m <_{\mathcal{SO}} n?) = Y$.
- "$m \not<_{\mathcal{SO}} n$" is an abbreviation for $\mathcal{SO}(m <_{\mathcal{SO}} n?) = N$.
- "$m \,?_{\mathcal{SO}}\, n$" is an abbreviation for $m <_{\mathcal{SO}} n?$ not being in the domain of $\mathcal{SO}$.
- $\langle e \rangle$ is the computation $\{e\}^{\mathcal{SO}}$.
- Q is said to be a freezing query if it is freezing relative to $\mathcal{SO}$, i.e. not in the domain of $\mathcal{SO}$.

Theorem 5. *$\mathcal{SO}$ is a system of notation for ordinals through the least recursively inaccessible ordinal α, and is a complete Σ_1 set over L_α.*

In order to prove this, we will need witnesses within L_α for assertions like $n \in \mathcal{SO}$ and $n \notin \mathcal{SO}$. Of course this will involve the trees of ordinal notation U_n. (Whether they are well- or ill-founded will determine whether n is in or out of $\mathcal{SO}$.) It will also involve the definition of $\mathcal{SO}$ as a least fixed point. Least fixed points of positive inductive definitions can be viewed as developed in stages indexed by the ordinals. In our case, elements enter $\mathcal{SO}$ at later stages because of computations based on fragments of $\mathcal{SO}$ from earlier stages, which are themselves based on sub-computations from yet earlier stages. It turns out to be useful to organize these sub-computations into a tree, the well-foundedness of which, or not, will be crucial. Please note that the trees of sub-computations are not to be confused with the trees of ordinal notations.

Definition 5. *For e a natural number and Q an oracle query, the trees S_e and S_Q (S for sub-computations) are defined recursively.*

For S_e, the root is e. Start running the oracle Turing computation $\{e\}$[2]. If it makes an oracle query Q, then append S_Q beneath the root in S_e. If the oracle $\mathcal{SO}$ returns an answer to Q, then the computation $\{e\}$ continues. Similarly if at any time the run of $\{e\}$ makes an oracle call R, then S_R is appended beneath the root in S_e, with the root of S_R being a child of e to the right of all earlier oracle calls Q.

For S_Q, the root is Q. Whether Q is $n \in \mathcal{SO}?$ or $m <_{\mathcal{SO}} n?$, let n_Q be n. To answer Q one would need to consider U_{n_Q}. The only nodes in U_{n_Q} that require any computation are nodes labeled $\langle e, k \rangle$. Append $S_{\langle e,k \rangle}$ for each such $\langle e, k \rangle$ beneath the root in S_Q.

Proposition 6. *$\langle e \rangle$ is non-freezing iff S_e is well-founded, and Q is non-freezing iff S_Q is well-founded.*

[2]Sometimes we will have occasion to consider the computation $\{\bar{e}\}(k)$ instead. Then implicity $e = \langle \bar{e}, k \rangle$.

Proof. First we show inductively on ranks that if S_e resp. S_Q is well-founded then $\langle e \rangle$ resp. Q is non-freezing.

The immediate sub-trees of S_e are the S_Q's, where the Q's are e's oracle queries. If S_e is well-founded then each such S_Q has smaller rank, so Q is non-freezing, hence the run of $\langle e \rangle$ does not freeze.

To say that Q is non-freezing means that Q is in the domain of $\mathcal{SO}$. Since $\mathcal{SO}$ is a fixed point of the operation $P \mapsto P^+$, it suffices to show that Q is in the domain of $\mathcal{SO}^+$. By an earlier proposition, that holds iff $U_{n_Q}^{\mathcal{SO}}$ does not freeze. Toward that end, we must consider only nodes of $U_{n_Q}^{\mathcal{SO}}$ labeled $\langle e, k \rangle$. For any such e and k, $S_{\langle e,k \rangle}$ is an immediate sub-tree of S_Q. Hence it has lower rank, so inductively $\langle e \rangle(k)$ is non-freezing.

The converse hinges on $\mathcal{SO}$ being the least fixed point: there is no need to go beyond the realm in which the trees of sub-computations are well-founded. Toward this end, let P be $\mathcal{SO} \restriction \{Q \mid S_Q \text{ is well-founded}\}$. We will show that P is a fixed point, which suffices.

Let Q be a query. We need to show that if $P^+(Q)$ returns an answer then so does $P(Q)$. Because $P \subseteq \mathcal{SO}$ and $\mathcal{SO}$ is a fixed point, $P^+ \subseteq \mathcal{SO}$. That means that Q is in the domain of $\mathcal{SO}$. To get Q to be in the domain of P, we need only show that S_Q is well-founded. The immediate sub-trees of S_Q are all of the form $S_{\langle e,k \rangle}$, for a node labeled $\langle e, k \rangle$ from U_{n_Q}. Because $P^+(Q)$ is defined, by the proposition $U_{n_Q}^P$ does not freeze. Because $\mathcal{SO} \supseteq P$, $U_{n_Q} = U_{n_Q}^{\mathcal{SO}} = U_{n_Q}^P$. Hence all of the nodes $\langle e, k \rangle$ we must consider from U_{n_Q} are already in $U_{n_Q}^P$. Because $U_{n_Q}^P$ does not freeze, every such $\{e\}^P(k)$ does not freeze. That means that when running $\{e\}(k)$, every time an oracle query R is made, the oracle P responds. By the definition of P, S_R is well-founded. Since the immediate sub-trees of $S_{\langle e,k \rangle}$ are all of that form, $S_{\langle e,k \rangle}$ is well-founded. Hence S_Q is well-founded. □

Corollary 7. *U_n is not freezing iff $S_{n \in \mathcal{SO}^?}$ is well-founded.*

We are now ready to prove the main theorem, that $\mathcal{SO}$ is a complete Σ_1 set over the least recursively inaccessible L_α.

Proof. Sketch of proof: Let P be $\mathcal{SO}$ restricted to those queries Q with $S_Q \in L_\alpha$. We will show that P is a fixed point. It is then immediate that $\mathcal{SO}$ is Σ_1 definable over L_α, as $\mathcal{SO}(Q) = Y$ resp. N iff there are a tree S_Q which is well-founded and a computation witnessing the answer Y resp. N.

All of the statements of interest, when true, have witnesses. For instance, that $n \in \mathcal{SO}$ is witnessed by $S_{n \in \mathcal{SO}}$ (as well as a ranking function to the ordinals), to show that U_n is not freezing, and U_n itself (as well as witnesses that U_n is well-formed and well-founded). That $\langle e \rangle$ does not freeze is witnessed by S_e (along with its ranking function). Of course, the constructions of the objects U_n, S_e, S_Q serve at witnesses that those objects are what they are purported to be. Hence we can think of these objects as being generated as we ascend the L-hierarchy. This justifies the notation U_n^β, S_e^β, etc., as U_n resp. S_e as defined over L_β, using only the witnesses within L_β. We will show that over L_α no new computations are defined over L_α, and that $U_n = U_n^\alpha, S_e = S_e^\alpha, S_Q = S_Q^\alpha$.

If $\langle e \rangle$ converges then that is witnessed by a finite run of a Turing machine, along with the witnesses to finitely many oracle calls. If all of the oracle call witnesses are in L_α, so is this finite sequence.

If $\langle e \rangle$ diverges then that is witnessed by an ω-sequence which is the divergent run of $\langle e \rangle$, along with a sequence of witnesses to oracle calls of length at most ω. If each such witness is in L_α, then by the admissibility of L_α so is the ω-sequence. Then the run of $\langle e \rangle$ is definable over that latter sequence.

In building U_n, work on one level at a time. (The root is level 0, its children level 1, etc.) It is immediate to determine if a node has any children and what those children are, except for nodes labeled $3 \cdot 5^e$. Since U_n is (by assumption) not freezing, each child $\langle e, k \rangle$ has a witness as to whether $\langle e \rangle(k)$ converges or diverges. Since each level can be arranged in an ω-sequence, this induces a total function from ω to these witnesses. By admissibility this function is in L_α. Furthermore, we have to repeat this construction on all ω-many levels of U_n, which again by admissibility is in L_α.

S_e is essentially the witnesses to the oracle calls from above that $\langle e \rangle$ does not freeze.

For S_Q well-founded, U_{n_Q} was already seen to be in L_α. The immediate subtrees of S_Q are the $S_{\langle e,k \rangle}$'s for $\langle e, k \rangle$ a node in U_{n_Q}. In L_α there is a sequence of such nodes of length at most ω. If each $S_{\langle e,k \rangle}$ were in L_α, then again by admissibility the sequence of such is in L_α, which puts S_Q into L_α.

Finally, in order to answer an oracle question Q, the only time Q has an answer is when U_{n_Q} is not freezing. So then U_{n_Q} is in L_α, as above. Whether it is ill-formed or not is witnessed within L_α, again using the admissibility of L_α. When U_{n_Q} is well-formed, whether it is well-founded is also witnessed within L_α, this time using the fact that α is a limit of admissibles: if a tree is in L_α, then a ranking function for the tree's well-founded part is definable over the least admissible set containing the tree, and hence is in L_α.

The harder direction is to show that L_α is a lower bound, in that every real in L_α is $\mathcal{SO}$-computable and moreover that Σ_1 questions about L_α can be converted uniformly to questions about membership in $\mathcal{SO}$. It should be clear from the presence of the oracle calls that the $\mathcal{SO}$-computable reals are closed under hyperjump. For instance, $\mathcal{O}$ is computable: for n to be a candidate for membership in $\mathcal{O}$, when building U_n no computations along the way may consult with an oracle, so there is no possibility for U_n to freeze. Hence the oracle answer to $n \in \mathcal{SO}$? is the correct information for whether n is in $\mathcal{O}$. Since this argument relativizes, the computable reals are closed under the hyperjump. In particular, $\mathcal{O}$ exists, as do $\mathcal{O}', \mathcal{O}''$, etc. It is not hard to define the join of the $\mathcal{O}^{(n)}$'s: split the work tape up into ω-many infinite tapes, and dedicate the n^{th} tape to $\mathcal{O}^{(n)}$. So the computable reals go beyond the first limit of admissibles.

To show however that we can go past any inadmissible limit of admissibles, this needs in some form the Gandy Selection Theorem, that the computable predicates are closed under a search through ω. This turns a fact of inadmissibility – $\forall i \in \omega \; \exists A_i \; \phi(i, A_i)$ – into a computable sequence $\langle A_i \rangle_{i \in \omega}$. Gandy Selection can be proved via a Stage Comparison Theorem. $\square$

4 Loose Feedback Hyperjump

In the inductive generation of $\mathcal{O}$ and $\mathcal{SO}$, there is really no difference in the ways numbers get put into those sets, the ways numbers get accepted as ordinal notations. The difference between them is that $\mathcal{SO}$ contains negative information too, that $\mathcal{SO}$ will tell you when something is not a member of $\mathcal{SO}$. This negative information is essentially the ill-foundedness of a certain tree (T_n resp. U_n). For $\mathcal{O}$, there was no possibility of T_n being freezing, whereas some U_n's most certainly are. It is part of the definition of $\mathcal{SO}$ that for an oracle call "$n \in \mathcal{SO}$?" to be non-freezing the tree U_n must be non-freezing. The requirement that all nodes in U_n be non-freezing can be explained or justified by thinking of U_n being generated by a (transfinite) breadth-first search. That is, first evaluate all the computations on the first level of U_n, then all those on the second level, and so on. After all ω-many levels, one can see whether the tree is well-founded. If any of those computations freezes, then this procedure cannot be completed.

This outcome, that a single freezing node in U_n freezes the question $n \in \mathcal{SO}$?, is necessary if the answer is to be "yes", because a "yes" answer is to be taken as a guarantor of U_n's well-foundedness. Imagine by way of contrast that some node of an otherwise well-founded tree were freezing. That freezing happens when a particular oracle call is made. You can think of the machine at that point as waiting for a response. This waiting can be taken to be measured along the ordinals as indexing L, but it does not have to be. As an alternative, while we are studying in this paper the semantics of the least fixed point, we don't have to. Perhaps a larger fixed point is generated by some random (albeit appropriate) computation all of a sudden no longer freezing. Perhaps such sudden removal of blockages can be organized in a partial order, like a kind of (or actual!) Kripke model. Then time could be taken as movement along this partial order. There could be other interpretations of time. A conservative way of thinking about freezing is that one should use no information about a freezing node, not even that it is freezing. So back to our pseudo well-founded tree with one freezing node. If that node ever gets unstuck, depending on what happens after that, the tree could become ill-founded. This issue does not come up if instead U_n is not freezing: U_n cannot change at all, even as or if the oracle changes; hence a well-founded U_n will remain well-founded.

In contrast, such prudence is not necessary for ill-foundedness. Once a tree is ill-founded, even as the tree grows it will remain ill-founded. Allowing an ill-founded tree, even when freezing, to qualify as a witness that a number is not ordinal notation is what we are calling **loose feedback** $\mathcal{O}$, or $\mathcal{LO}$. This allowance can be explained or justified by thinking of U_n as being generated while ascending through L. As one climbs through the L_α's, α increasing, more computations become completed (i.e. converge or diverge), so more nodes get placed into U_n. If at any stage U_n is seen to be ill-founded, even if it still contains freezing nodes, then we can take that as a witness that $n \notin \mathcal{LO}$.

In comparing the negative information in $\mathcal{LO}$ with that of $\mathcal{SO}$, it comes down to a kind of parallelism. Normally in mathematical logic parallelism plays no role, since it can be simulated by sequential computation via dovetailing. This

does not work with feedback around: once a freezing oracle call is made, the entire computation stops. This was used in [9] to define parallel feedback (Turing) computability, by which an ω-sequence of machines was run in parallel, which was shown to be stronger than (sequential) feedback Turing computability. Parallelism was also defined for infinite time Turing machines [8] (and ultimately analyzed in [16], even if the framework there is Kleene's higher types [6], as the results are translatable to feedback ITTMs). In the current setting, it's as though we're searching for an infinite branch in a tree, even if another part of the tree is freezing. Instead of running ω-many machines in parallel, what we have here can be called **tree parallelism**.

We will make use of the same trees U_n^P as in the previous section. Given an oracle, there is no change from before about the induced tree of ordinal notations. The difference from before is the inductive step on feedback oracles, there called P^+, here $P^{\&}$.

Definition 6. *1.* $P^{\&}(n \in \mathcal{SO}?) = Y$ *if* U_n^P *is well-formed and well-founded.*

2. $P^{\&}(n \in \mathcal{SO}?) = N$ *if* U_n^P *is either ill-formed or ill-founded.*

3. $P^{\&}(m <_{\mathcal{SO}} n?) = Y$ *if* $P^{\&}(n \in \mathcal{SO}?) = Y$ *and* $m \neq n$ *is a node in* U_n^P.

4. $P^{\&}(m <_{\mathcal{SO}} n?) = N$ *if* $P^{\&}(n \in \mathcal{SO}?) = N$, *or if* $P^{\&}(n \in \mathcal{SO}?) = Y$ *and either* $m = n$ *or* m *is not a node in* U_n^P.

Note that the only difference between P^+ and $P^{\&}$ is in clause 2.[3]

Proposition 8. *The definition of* $P^{\&}$ *is positive in* P. *Hence if* $P \subseteq Q$ *then* $P^{\&} \subseteq Q^{\&}$.

The preceding proposition justifies the following.

Definition 7. $\mathcal{LO}$ *or* **loose** $\mathcal{O}$ *is the least fixed point of the operation* $P \mapsto P^{\&}$. V_n *is* $U_n^{\mathcal{LO}}$.

While we're at it, we will also define the trees of sub-computations $S_e^{\&}$ and $S_Q^{\&}$. Formally speaking, they are defined the same way S_e and S_Q were in the previous section, only with reference to $\mathcal{SO}$ replaced by $\mathcal{LO}$. This affects $S_e^{\&}$ directly: if $\{e\}$ makes an oracle call, it is more likely to get an answer from $\mathcal{LO}$

[3]It bears mention that there are several options for dealing with this clause. In all cases, the evidence that n is not an ordinal notation is that its tree U_n^P of smaller ordinal notations is bad somehow, either ill-formed or ill-founded. For P^+, we took this in the strictest possible sense: U_n^P had to be non-freezing in order to qualify as evidence. For $P^{\&}$, there is no such requirement on U_n^P ever; once we have any evidence that U_n^P will not be acceptable, we take it. In contrast with both of these, one could work in the middle. That is, the reasons that U_n^P activate clause 2 are that it has a node not of the right form, or that the function named by a node is partial, or that the function named by a node is not increasing (in the sense of $<_P$), or that the tree has an infinite descending path; the requirement that U_n^P be non-freezing could, in principle, be levied on some and not all of these conditions. We find the two extreme cases isolated here to be the most natural ones; we believe that the only condition of any real importance is the well-foundedness of U_n^P, and that varying the others will make no difference; determining this is left for future work.

than $\mathcal{SO}$. Then this affects $S^{\&}_Q$, which is defined in terms of $S^{\&}_e$ the way S_Q is defined in terms of S_e.

Definition 8. *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. But this is not the first time it has come up. Extending work in [11], it was shown in [7] that such ordinals are exactly the Σ^1_1 reflecting ordinals. (In this context, the superscript 1 refers not to reals but to subsets of the structure over which the formula is being evaluated.) 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; see [3]. 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.) In [11] it was also shown that the least Σ^1_1 reflecting ordinal is also the closure point of Σ^1_1 monotone inductive definitions. (Here the superscript does refer to reals.) Furthermore, that is the same least ordinal which provides winning strategies for all Σ^0_2 games (Solovay, see [10] 7C.10 or [15]). (If Player I has a winning strategy, then there is one in L_γ; if II does, then there is one in L_{γ^+}.) As though that weren't enough, [14] shows the equivalence of closure under Σ^1_1 monotone inductive definitions with the Σ^1_1 Ramsey property. (For all Σ^1_1 partitions P of ω there is an infinite set $H \subseteq \omega$ such that the infinite subsets of H are either all in P or all not in P.) An ordinal α is Gandy if the α-computable well-orderings are cofinal through α^+; γ is the least non-Gandy ordinal [5]. Closest of all to the work being discussed here, γ is also the closure ordinal of context-dependent deterministic parallel feedback Turing computability [2, 9]. With all of these applications, this definition counts as natural.

Theorem 9. *$\mathcal{LO}$ is a system of notation for ordinals through the least ordinal γ which is Π_1 gap-reflecting on admissibles, and is a complete Σ_1 set over L_γ.*

Proof. It is easier to show that γ is an upper bound. The notations $V^\beta_n, S^{\&\beta}_e, S^{\&\beta}_Q$ mean $V_n, S^{\&}_e, S^{\&}_Q$ as interpreted in L_β. In the definition of $P^{\&}$, clauses 1, 3, and 4 are the same as for P^+. So by the arguments for the previous section $\mathcal{LO}^\gamma$ is closed under those clauses by the admissibility of γ. Similarly for the ill-formedness condition of clause 2: if V^γ_n is ill-formed, then so is some V^β_n ($\beta < \gamma$). Now suppose V^γ_n were ill-founded. Because V^γ_n is definable over L_γ, its ill-foundedness is a Π_1 statement over L_{γ^+} with parameter γ. Therefore, by the choice of γ, there is a smaller β with $L_{\beta^+} \models$ "V^β_n is ill-founded." So there is already a witness to n not being in $\mathcal{LO}$ in L_γ.

To show that γ is a lower bound, we will interpret, or simulate, parallel feedback Turing computability [2, 9] within (computability relative to) $\mathcal{LO}$. Since the former has already been shown to compute everything in L_γ, this suffices. The reason this reduction would hold is that the same structures are involved with both of them. In more detail, the $\mathcal{LO}$ computations are run by the trees V_n of ordinal notations and $S^{\&}_e, S^{\&}_Q$ of sub-computations. For a computation not to freeze, it is not necessary that V_n not freeze (as opposed to U_n), much less be well-founded. For $S^{\&}_e$ and $S^{\&}_Q$, it is not necessary that they be well-founded (as opposed to S_e and S_Q), just well-founded in the right way: an infinite path through V_n determines infinitely many sub-trees (rooted on the first level) of S_Q, where Q is $n \in \mathcal{LO}?$, and they all must be well-founded. Now consider the trees that come up in parallel feedback. Most prominent is $C^{(e,n)}$, the tree of runs. In order for the parallel feedback computation $\langle e \rangle(n)$ not to freeze, it is not necessary that $C^{(e,n)}$ not freeze; rather, $C^{(e,n)}$ could have a terminal node, which is the uninteresting case in all the proofs, or it is ill-founded. This is the same behavior as the V_n's. The work on parallel feedback did not discuss the tree of sub-computations, because it no longer had to be well-founded; in fact, the only well-foundedness that matters is that of an infinite set of sub-trees as determined by some infinite path through $C^{(e,n)}$. The stopping conditions are the same in both cases. That is why ultimately each can code the other. $\square$

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] Nathanael Ackerman, Cameron Freer, and Robert Lubarsky, "An Introduction to Feedback Turing Computability," **Annals of Pure and Applied Logic**, special issue on LFCS '16, submitted; also available at http://math.fau.edu/lubarsky/pubs.html

[3] A. Arnold and D. Niwinski, **Rudiments of μ-Calculus, Studies in Logic and the Foundations of Mathematics**, v. 146, North Holland, 2001

[4] Jon Barwise, **Admissible Sets and Structures, Perspectives in Mathematical Logic**, Springer-Verlag, Berlin 1975

[5] Richard Gostanian, "The Next Admissible Ordinal," **Annals of Mathematical Logic**, v. 17 (1979), pp. 171-203

[6] Stephen Cole Kleene, "Recursive functionals and quantifiers of finite types. I," **Transactions of the American Mathematical Society**, v. 91 (1959), pp. 1–53

[7] Robert Lubarsky, "μ-definable sets of integers," **The Journal of Symbolic Logic**, v. 58 (1) (1993), pp. 291–313

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

[9] Robert Lubarsky, "Parallel Feedback Turing Computability," in **Proceedings of LFCS 2016, Lecture Notes in Computer Science 9537** (Sergei Artemov and Anil Nerode, eds.), pp. 236-250

[10] Yiannis Moschovakis, **Descriptive Set Theory**, First edition North Holland (1987); second edition AMS (2009)

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

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

[13] Gerald Sacks, **Higher Recursion Theory, Perspectives in Mathematical Logic**, Springer-Verlag, Berlin 1990, pp. xvi+344

[14] Kazuyuki Tanaka, "The Galvin-Prikry Theorem and Set Existence Axioms," **Annals of Pure and Applied Logic** 42 (1), pp. 81-104 (1989)

[15] Kazuyuki Tanaka, "Weak Axioms of Determinacy and Subsystems of Analysis II (Σ^0_2 Games)," **Annals of Pure and Applied Logic** 52 (1-2), pp. 181-193 (1991)

2008

[16] Philip Welch, "$G_{\delta\sigma}$-games and generalized computation," to appear