Higher-order feedback computation

Juan P. Aguilera[1], Robert S. Lubarsky[2], and Leonardo Pacheco[3][0000−0002−7703−7990]

[1] TU Wien, Austria
aguilera@logic.at
[2] Florida Atlantic University, USA
rlubarsk@fau.edu
[3] TU Wien, Austria
leonardo.pacheco@tuwien.ac.at

Abstract. Feedback Turing machines are Turing machines which can query a halting oracle which has information on the convergence or divergence of *feedback* computations. To avoid a contradiction by diagonalization, feedback Turing machines have two ways of not converging: they can diverge as standard Turing machines, or they can freeze. A natural question to ask is: what about feedback Turing machines which can ask if computations of the same type converge, diverge, or freeze? We define αth order feedback Turing machines for each computable ordinal α. We also describe feedback computable and semi-computable sets using inductive definitions and Gale–Stewart games.

Keywords: Turing computation · Feedback computation · Fixed-point operators.

1 Introduction

Feedback Turing machines are Turing machines which can query a halting oracle $h :\subseteq \omega \times \omega \to \{\downarrow, \uparrow\}$, which has information on the convergence or divergence of *feedback* computations. That is, given the code e for a feedback Turing machine and an input n the oracle answers if the computation $\{e\}^h(n)$ converges or diverges. To avoid a contradiction by diagonalization, feedback Turing machines have two ways of not converging: they can diverge as standard Turing machines, or they can freeze. A feedback Turing machine freezes when it asks the halting oracle h about a pair $\langle e, n\rangle$ not in the domain of h.

Feedback Turing machines were first studied by Ackerman, Freer and Lubarsky [2, 3]. They proved that the feedback computable sets are the Δ^1_1 sets and the feedback semi-computable sets are the Π^1_1 sets. We can also show that the feedback semi-computable sets are the winning regions of Gale–Stewart games with Σ^0_1 payoff [14]. It is quite curious that some of the key results of [2] were announced in Rogers' textbook on recursion theory [15], almost 50 years before proofs were published.

Lubarsky [11] defined feedback infinite time Turing machines and their subcomputation trees. He showed that feedback writable, feedback eventually writable,

and feedback accidentally writable reals coincide; this does not happen for standard infinite time Turing machines. By a result of Welch [16], the feedback infinite time Turing machine semi-computable sets are the winning regions of Σ^0_3 Gale–Stewart games. On the other hand, one can also add feedback to weaker notions of computability. Ackerman *et al.* [2, 3] studied feedback primitive recursion. Feedback primitive recursive sets coincide with the computable sets. That is, computability via Turing machines itself is a kind of feedback computation. Ackerman *et al.* also lifted the results of feedback computability over ω to feedback computability over 2^ω in [1].

A natural question to ask is: what about feedback Turing machines which can ask if computations of the same type converge, diverge, or freeze? These new machines are second-order feedback machines. Note that we must now have a new and stronger notion of freezing to avoid a contradiction by diagonalization. Having defined second-order feedback computation, it is now natural to ask: what about third-, fourth-, and higher-order feedback?

We define αth order feedback Turing machines for each computable ordinal α. We also describe feedback computable and semi-computable sets using inductive definitions and Gale–Stewart games. Specifically, we prove the following level-by-level correspondence:

Theorem 1. *For all $\alpha < \omega_1^{\mathrm{ck}}$, the following classes of sets of integers coincide:*

1. *the $(\alpha+1)$-feedback semi-computable sets;*
2. *the $\Sigma^\mu_{\alpha+1}$-definable sets; and*
3. *the $\eth(\Sigma^0_2)_\alpha$ sets.*

Proof. Items 1 and 2 are equivalent by Theorems 2 and 3. Items 2 and 3 are equivalent by [9, 8].

We now turn our attention to μ-arithmetic. μ-arithmetic is obtained by adding least and greatest fixed-point operators to first-order arithmetic. It was inspired by Kozen's [10] modal μ-calculus, an extension of modal logic by fixed-point operators. Lubarsky [12] characterized the μ-definable sets using n-reflecting ordinals. This characterization was later used by Bradfield [7] to show the strictness of the modal μ-calculus' alternation hierarchy, a problem which stayed open for almost a decade. We should note that we are restricted to taking fixed points of positive formulas in the μ-arithmetic; a closely related system is Möllerfeld's σ-arithmetic [13], which lifts this restriction. As the μ- and the σ-arithmetics define the same sets of natural numbers, we restrict ourselves to the μ-arithmetic.

Gale–Stewart games are strong and flexible tools in descriptive set theory. In a Gale–Stewart game $G(A)$ with payoff $A \subseteq \omega^\omega$, two players alternate picking natural numbers to for a sequence α. The first player wins the game $G(A)$ iff they have a strategy which guarantees the generated sequences are inside A, no matter what the second player does. Given a set $A \subseteq \omega^\omega$, the set $\eth A \subseteq \omega^{<\omega}$ is the set of finite sequence $s \in \omega^{<\omega}$ such that the first player has a winning strategy for $G(A)$ starting from s. In other words, $\eth A$ is the set of winning positions for the first player in $G(A)$. Bradfield [6] proved that the μ-definable sets are the

winning positions of games whose payoff sets are finite boolean combinations of Σ^0_2 sets. This was later extended by Bradfield, Duparc, and Quickert [9, 8] to transfinite μ-formulas and transfinite boolean combinations.

Closely related to higher-order feedback computability are the feedback hyperjump studied by Aguilera and Lubarsky [4] and feedback ${}^2E$-computability studied by Aguilera and Soto [5]. These concepts have two variations: a strict one and a loose one. These refer to how the subcomputation trees are defined: in the loose variations, ill-founded subcomputation trees can be witnesses to non-freezing computations. We will see a similar phenomenon in subcomputation trees for higher-order feedback computations.

Corollary 1. *The following are equivalent:*

1. *2-feedback semi-computability;*
2. *loose feedback ${}^2E$-semi-computability; and*
3. *computable reducibility to the loose feedback hyperjump $\mathcal{LO}$.*

Proof. By [5] and [4].

Outline In Section 2, we define α-feedback computability. In Section 3, we define subcomputation trees for α-feedback computations. In Section 4, we define the transfinite μ-arithmetic and its alternation hierarchy. In Sections 5 and 6, we prove the equivalence between feedback computability and μ-definability.

2 Higher-order feedback computability

Fix $\alpha < \omega_1^{\text{ck}}$ and a computable notation for α. We define αth order feedback Turing machines. We omit the reference to the ordinal when not ambiguous and abbreviate "αth order feedback Turing machines" by "feedback machines". Write $-1 \leq \beta < \alpha$ to mean $\beta = -1$ or $\beta < \alpha$. We will use symbols $\uparrow_\beta$ as notation for the outputs of the freezing oracles. Note that $\uparrow_{-1}$ will be used to indicate convergent computations; we also write $\uparrow_{-1}$ as $\downarrow$. Similarly, $\uparrow_0$ will indicate divergent (and non-freezing) computations.

We can extend any standard encoding of Turing machines as natural numbers to α-feedback Turing machines. We use the fixed encoding of α as a computable ordering to add commands to query the freezing oracles. The encoding of freezing queries are no different from encodings of oracle queries in a relativized computation. Kleene's Recursion Theorem also holds for α-feedback Turing machines by the standard proof, which will be useful for us later.

Intuitively, an α-feedback machine can query freezing oracles

$$f_\beta : F_\beta \to \{\uparrow_{\beta'} | \ -1 \leq \beta' \leq \beta\},$$

with $F_\beta \subseteq \omega \times \omega$ and $\beta < \alpha$. We call f_β the β-freezing oracle. The domain F_β of f_β contains the indices and inputs of computations which $\leq \beta$-freeze. Given the code $e \in \omega$ of a feedback machine and some input $n \in \omega$, we denote the computation

$\{e\}^{\{f_\beta\}_{\beta<\alpha}}(n)$ by $\langle e\rangle^\alpha(n)$. The oracle f_β returns $\uparrow_\gamma \in \{\uparrow_{\beta'} | -1 \leq \beta' < \beta\}$ iff $\langle e\rangle^\alpha(n)$ γ-freezes.

A computation $(\beta+1)$-freezes iff it queries f_β about some pair not in F_β. If λ is a limit, a computation λ-freezes iff it makes a query about $\langle e, n\rangle \notin \bigcup_{\alpha<\lambda} F_\alpha$.[4] We also say that a computation 0-freezes when it is divergent and that it -1-freezes when it is convergent. We write $\langle e\rangle^\alpha(n) \uparrow_\beta$ iff $\langle e\rangle^\alpha(n)$ β-freezes.

Formally, we define:

Definition 1. *Let $\alpha < \omega_1^{\mathrm{ck}}$. For all $\beta < \alpha$, let $F_\beta \subseteq \omega \times \omega$ be the least relation such that the function $f_\beta : F_\beta \to \{\uparrow_{\beta'} | -1 \leq \beta' < \beta\}$ is such that $\langle e, n\rangle \in F_\beta$ and $f_\beta(e, n) = \uparrow_\gamma$ iff $\langle e\rangle^\alpha(n)$ makes no β'-freezing query outside of $F_{\beta'}$ and γ-freezes.*

A set $A \subseteq \omega$ is α-feedback computable iff there is an α-feedback machine with index e such that:

$$\langle e\rangle^\alpha(n) = \begin{cases} 1, \text{ if } n \in A \\ 0, \text{ if } n \notin A \end{cases}$$

A set $A \subseteq \omega$ is α-feedback semi-computable iff there is an α-feedback machine with index e such that

$$\langle e\rangle^\alpha(n) \downarrow \text{ iff } n \in A.$$

Using Theorem 1, we can show that $A \subseteq \omega$ is α-feedback computable iff A and $\omega \setminus A$ are α-feedback semi-computable.

One should be careful that the freezing oracles depend on the fixed α. For a more precise notation, we could write f_β as f_β^α. We do not do so as we always work with a fixed α. To see why the freezing oracles depend on α, consider the 0-freezing oracle f_0^1 for 1-feedback machines and the 0-freezing oracle f_0^2 for 2-feedback machines are different partial functions. f_0^2 has information about halting computations which have freezing subcomputations via queries to f_1^2.

Note that our 1-feedback machines are equivalent to the feedback machines in Ackerman *et al.* [3]. Furthermore, our 0-freezing oracle f_0 for 1-feedback machines is equivalent to their halting oracle. We could also call f_0 the halting oracle, but we prefer not to do so for uniformity of notation. Note also that 0-feedback machines are just standard Turing machines.

Before proceeding, we show that the freezing oracles $\{f_\beta\}_{\beta<\alpha}$ are well-defined using simultaneous inductive definitions. We will come back to this proposition when we show that feedback semi-computable sets are definable in μ-arithmetic.

Proposition 1. *Fix $\alpha < \omega_1^{\mathrm{ck}}$. For all $\beta < \alpha$, there is a smallest relation $F_\beta \subseteq \omega \times \omega$ and a function $f_\beta : F_\beta \to \{\uparrow_{\beta'} | -1 \leq \beta' < \beta\}$ such that $\langle e, n\rangle \in F_\beta$ and $f_\beta(e, n) = \uparrow_\gamma$ iff $\langle e\rangle^\alpha(n)$ makes no β'-freezing query outside of $F_{\beta'}$ and γ-freezes.*

Proof. We define the F_β and f_β by simultaneous inductive definitions.

Given $\beta < \alpha$, Γ_β is an auxiliary function taking sequences $\{g_{\beta'}\}_{\beta'<\alpha}$ of freezing oracles to the set of indices and inputs which β-freeze:

$$\Gamma_\beta(\{g_{\beta'}\}_{\beta'<\alpha}) = \{\langle e, n\rangle \mid \{e\}^{\{g_{\beta'}\}_{\beta'<\alpha}}(n) \ \beta\text{-freezes}\}.$$

[4] See the definition of *subcomputation trees* below.

We define operators $h_{(\beta,\cdot)}$ using the $\Gamma_{\beta'}$ with $\beta' \leq \beta$:

$$h^{-1}_{(\beta,\{g_\gamma\}_{\gamma<\alpha})}(\uparrow_{\beta'}) = \Gamma_{\beta'}(\{g_\gamma\}_{\beta'<\alpha}) \text{ for } \beta' \leq \beta.$$

Each $h_{(\beta,\cdot)}$ can be seen as an operator on sequences of partial functions from ω to ω. Furthermore, $h_{(\beta,\cdot)}$ is monotone: given sequences of partial functions $\{g_\gamma\}_{\gamma<\alpha}$ and $\{g'_\gamma\}_{\gamma<\alpha}$ such that $g_\gamma \subseteq g'_\gamma$ for all $\gamma < \alpha$, then $h_{(\beta,\{g_\gamma\}_{\gamma<\alpha})} \subseteq h_{(\beta,\{g'_\gamma\}_{\gamma<\alpha})}$. Let $\{f_\beta\}_{\beta<\alpha}$ be the sequence of the smallest partial functions such that of $h_{(\beta,\{f_\beta\}_{\beta<\alpha})} = f_\beta$. That is, if $h_{(\beta,\{f'_\beta\}_{\beta<\alpha})} = f'_\beta$, then $f_\beta \subseteq f'_\beta$ for all $\beta < \alpha$.

3 Subcomputation trees

Fix $e, n \in \omega$ and $\alpha < \omega_1^{\text{ck}}$. We define a subcomputation tree $T_{e,n}$ to witness the convergence, divergence, or freezing of the α-feedback computation $\langle e\rangle^\alpha(n)$. Our trees are similar to the subcomputation trees for 1-feedback Turing computation found in [3].

We will also consider a trimmed version $T^{\text{trim}}_{e,n}$ of the subcomputation trees. As the subtrees for 1-feedback computation, $T^{\text{trim}}_{e,n}$ will be wellfounded iff $\langle e\rangle^\alpha(n)$ is convergent or divergent, and $T^{\text{trim}}_{e,n}$ will be non-wellfounded iff $\langle e\rangle^\alpha(n)$ is β-freezing for some $\beta \geq 1$. We will trim $T_{e,n}$ because higher-order queries allow non-wellfounded trees to witness converging and diverging feedback computations.

After we finish the construction of $T_{e,n}$ and $T^{\text{trim}}_{e,n}$, we will have:

Proposition 2. *Let $e, n \in \omega$ and $T^{\text{trim}}_{e,n}$ be the trimmed subcomputation tree of $\langle e\rangle^\alpha(n)$. Then:*

1. *The computation $\langle e\rangle^\alpha(n)$ is non-freezing iff $T^{\text{trim}}_{e,n}$ is well-founded.*
2. *The computation $\langle e\rangle^\alpha(n)$ is β-freezing iff $T^{\text{trim}}_{e,n}$ has an infinite path $\rho = \{\langle e_i, n_i\rangle\}_{i\in\omega}$ such that*

   $$\beta = \limsup\{\beta_i + 1 \mid \text{the edge between } \langle e_i, n_i\rangle \text{ and } \langle e_{i+1}, n_{i+1}\rangle \text{ is labeled } \beta_i\}.$$

   Furthermore, the path ρ is the rightmost infinite path in the tree $T^{\text{trim}}_{e,n}$.

Fix an index e and an input n, we build the subcomputation tree of $\langle e\rangle^\alpha(n)$ by stages. The subcomputation tree $T_{e,n}$ will be a labeled subtree of $\omega^{<\omega}$. We label each node of $T_{e,n}$ with a pair $\langle e', n'\rangle$ consisting of an index e' for a feedback Turing machine and an input n'. At each node $\langle e', n'\rangle$, we simulate a feedback machine with index e' starting on input n'. At all times there is a node of $T_{e,n}$ that is in control of the computation. On successor stages, we will run one instruction in the computation being simulated at the control node. When querying the freezing oracle f_β about $\langle e'', n''\rangle$, we pass the control to a new child node labeled $\langle e'', n''\rangle$. We label the edge between $\langle e', n'\rangle$ and $\langle e'', n''\rangle$ with the ordinal $\beta < \alpha$.

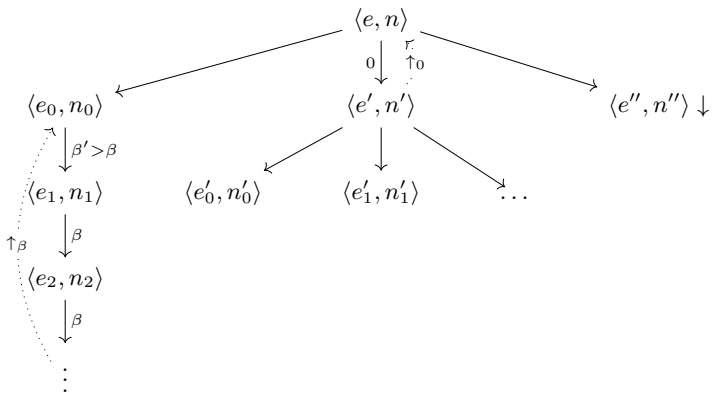

Fig. 1. The tree $T_{\langle e,n\rangle}$ of a converging computation $\langle e, n\rangle$ with diverging and freezing subcomputations.

First stage We add root node to $T_{e,n}$. Label the root node by $\langle e, n\rangle$; it is initially in control of the computation. Go to the next construction stage.

Successor stages Suppose the control is currently at the node $\langle e', n'\rangle$ of $T_{e,n}$. What we do in this stage of the construction depends on what is the next instruction on the computation being simulated at $\langle e', n'\rangle$.

Suppose the next instruction is not a query to a freezing oracle. Run the instruction. If the computation does not converge, we go to the next stage; the control stays at the same node. If the computation converges, there are two possibilities. If the control node is the root node, the whole computation halts. If the control node is not the root node, we pass control to its parent node; the parent then gets the answer $\downarrow$ to its freezing query and we go to the next stage.

Suppose the next instruction is a query to a freezing oracle h_β about a computation $\langle e'', n''\rangle$. We create a new child node to the right of all existing children. We label the child node by $\langle e'', n''\rangle$ and the path between parent and child by β. The the child node now controls the computation. Go to the next stage.

Limit stages Suppose that we are on a limit stage of the construction of $T_{e,n}$. In this stage, we decide if some subcomputation of $\langle e\rangle^\alpha(n)$ diverged or froze.

Suppose that the control goes back to a node $\langle e', n'\rangle$ infinitely many times in a final segment of our construction (or that the control stays at a same node $\langle e', n'\rangle$). In this case, the computation at the node $\langle e', n'\rangle$ diverges. If $\langle e', n'\rangle$ is the root node, then the whole computation diverges. If $\langle e', n'\rangle$ is not the root node, pass the control to its parent and answer the parent's freezing query with $\uparrow_0$; we then go to the next stage.

Suppose there is no node $\langle e', n'\rangle$ such that the control goes back to $\langle e', n'\rangle$ infinitely many times in a final segment of the construction up to this point. Let $\rho = \{\langle e_i, n_i\rangle\}_{i\in\omega}$ be the rightmost infinite path in the tree. The control of the computation will have been in nodes of ρ on infinitely many stages. Let

β_i be the ordinal labeling the edge between $\langle e_i, n_i\rangle$ and $\langle e_{i+1}, n_{i+1}\rangle$. Let $\beta := \limsup\{\beta_i + 1 \mid i \in \omega\}$. We pass the control to the lowest node $\langle e_i, n_i\rangle$ in the path, if it exists, such that the edge between $\langle e_i, n_i\rangle$ and $\langle e_{i+1}, n_{i+1}\rangle$ is labeled by some $\beta' \geq \beta$. We then answer $\uparrow^\beta$ to the freezing query done in the computation at $\langle e_i, n_i\rangle$. If there is no such node, then the whole computation β-freezes.

Trimmed subcomputation trees We now define the trimmed subcomputation tree $T^{\mathrm{trim}}_{e,n}$ for $\langle e\rangle^\alpha(n)$, given the subcomputation tree $T_{e,n}$. Let $\rho = \{\langle e_i, n_i\rangle\}_{i\in\omega}$ be an infinite path in $T_{e,n}$. Let β_i be the ordinal labeling the edge between $\langle e_i, n_i\rangle$ and $\langle e_{i+1}, n_{i+1}\rangle$ and $\beta := \limsup\{\beta_i + 1 \mid i \in \omega\}$. Let $\langle e_i, n_i\rangle$ lowest in the path such that the edge between $\langle e_i, n_i\rangle$ and $\langle e_{i+1}, n_{i+1}\rangle$ is labeled by some $\beta' \geq \beta$. For all $j > i$, $\langle e_j, n_j\rangle$ is not in $T^{\mathrm{trim}}_{e,n}$. A node $\langle e', n'\rangle$ of $T_{e,n}$ is in $T^{\mathrm{trim}}_{e,n}$ iff it was not excluded by this procedure.

4 The μ-arithmetic

The language $\mathcal{L}_\mu$ of μ-arithmetic is obtained by adding countably many set variables and the fixed-point operators μ and ν to the language $\mathcal{L}_1$ of first-order arithmetic. Therefore, the μ-arithmetic has two types of terms: number and set terms.

Number terms are build up from constants 0 and 1, number variables, addition, and multiplication:

$$t := 0 \mid 1 \mid x \mid t + t \mid t \times t.$$

We define μ-formulas and μ-terms by simultaneous induction. Set terms are set variables or fixed-points:

$$T := X \mid \mu x X.\varphi \mid \nu x X.\varphi,$$

Atomic formulas are of the form $t = t'$ and $t \in T$, where t, t are number terms and T is a set term. Formulas are defined as follows:

$$\varphi := t = t \mid t \in T \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x.\varphi \mid \forall x.\varphi \mid \bigvee_{i<\omega} \varphi_i \mid \bigwedge_{i<\omega} \varphi_i.$$

The set terms $\mu x X.\varphi$ and $\nu x X.\varphi$ are well-formed iff X is positive in φ. That is, each occurrence of X in φ is in the scope of an even number of negations (possibly none). The formulas $\bigvee_{i<\omega} \varphi_i$ and $\bigwedge_{i<\omega} \varphi_i$ are only defined if there is a computable enumeration of the formulas φ_i and there is a finite collection $\mathcal{X}$ of number and set variables such that the free variables of each φ_i are in $\mathcal{X}$. This restriction allows us to, given a transfinite formula with only positive occurrences of variables, obtain a closed set term with finitely many applications of fixed-point operators.

We use η to denote either μ or ν. Note that the fixed-point operator $\eta x X$ in a set term $\eta x X.\varphi$ binds both the occurrences of the number variable x and

the occurrences of the set variable X. We usually denote the variables in a fixed point operator $\eta x X$ by lowercase and uppercase versions of the same letter.

We can encode the μ-formulas as usual. We only need to take care when encoding transfinite disjunctions and conjunctions. As we only to consider conjunctions and disjunctions of recursively enumerable many formulas, we can code this with indexes for programs enumerating codes for the formulas in these conjunctions and disjunctions. A note of caution: a number may encode some non-well-formed formula with non-wellfounded syntax tree, but this will not be a problem since we will never quantify over all the codes.

We interpret the μ-formulas over the set of natural numbers, where the first-order objects and set membership have their standard interpretations. We only need to define the interpretation of the fixed-point operators. If X is positive in $\varphi(x, X)$, define the operator $\Gamma_\varphi : \mathcal{P}(\omega) \to \mathcal{P}(\omega)$ by:

$$\Gamma_\varphi(A) := \{n \in \omega \mid \varphi(n, A)\}.$$

We can show by induction on the structure of φ that, if X is positive in φ, then Γ_φ is monotone: if $A \subseteq B$, then $\Gamma_\varphi(A) \subseteq \Gamma_\varphi(B)$. By the Knaster–Tarski Theorem, Γ_φ has least and greatest fixed-points. Denote by $\mu x X.\varphi$ the least fixed-point of Γ_φ and by $\nu x X.\varphi$ the greatest fixed-point of Γ_φ.

We can use games to define a different but equivalent alternative for the μ-arithmetic's semantics. We sketch it here at is is quite useful to understand the meaning of μ-formulas. Given a μ-formula φ and interpretations for its free set variables, Verifier and Refuter play a game to decide whether φ hold. When discussing formula ψ, one of the players will have to propose a new formula to discuss. For example, when discussing $\exists x.\psi(x)$, Verifier has to choose some $\psi(n)$; similarly, when discussing $\bigwedge_{i\in\omega} \psi_i$, Refuter has to choose some ψ_i. If discussing $\neg\psi$, the players switch roles and discuss ψ. When discussing $\tau \in \eta x X.\psi$, the players discuss $\psi(\tau, \eta x X.\psi)$. When discussing $\tau \in X$, the players go to $\tau \in \eta x X.\psi$, where $\eta x X.\psi$ the smallest formula where X is bound; in this case, we say that X was regenerated. We summarize the possible plays in Table 1 below.

At a position of the form $t = s$, Verifier wins iff the equality is true. Let ρ be an infinite play and $X_0, \ldots, X_n$ be the variables regenerated infinitely often and $\eta_0.\psi_0, \ldots \eta_n.\psi_n$ be the associated formulas. Let η_i be the fixed-point operator with biggest scope. Verifier wins ρ iff η_i is a ν.

For context, the evaluation game for any transfinite μ-formula is determined because it can be written as a Δ^0_3 Gale–Stewart game; and so it is determined by Borel determinacy. See [9] for a proof that Verifier wins the evaluation game for φ iff φ is true.

The alternation hierarchy classifies the μ-formulas according to the alternation of least and greatest fixed-point operators. For all computable α, the αth level of the alternation hierarchy is defined by:

- $\Sigma^\mu_0 = \Pi^\mu_0 :=$ all first-order formulas (with set variables);
- $\Sigma^\mu_{\alpha+1} :=$ the closure of $\Sigma^\mu_\alpha \cup \Pi^\mu_\alpha$ under $\wedge, \vee, \exists, \forall$ and $\tau \in \mu x X.\varphi$;
- $\Pi^\mu_{\alpha+1} :=$ the closure of $\Sigma^\mu_\alpha \cup \Pi^\mu_\alpha$ under $\wedge, \vee, \exists, \forall$ and $\tau \in \nu x X.\varphi$;

Table 1. The rules of evaluation game for μ-arithmetic.

Verifier		Refuter	
Position	Admissible moves	Position	Admissible moves
$\psi_1 \vee \psi_2$	$\{\psi_1, \psi_2\}$	$\psi_1 \wedge \psi_2$	$\{\psi_1, \psi_2\}$
$\exists x.\psi(x)$	$\{\psi(n) \mid n \in \omega\}$	$\forall x.\psi(n)$	$\{\psi(n) \mid n \in \omega\}$
$\bigvee_{i\in\omega} \psi_i$	$\{\psi_i \mid i \in \omega\}$	$\bigwedge_{i\in\omega} \psi_i$	$\{\psi_i, \mid i \in \omega\}$
$\tau \in \mu x X.\psi(x, X)$	$\{\psi(\tau, \mu x X.\psi)\}$	$\tau \in \nu x X.\psi(x, X)$	$\{\psi(\tau, \nu x X.\psi)\}$

- Σ^μ_λ := the closure of $\bigcup_{\alpha<\lambda} \Sigma^\mu_\alpha$ under recursive enumerable disjunctions $\bigvee_{i<\omega}$, when λ is a limit;
- Π^μ_λ := the closure of $\bigcup_{\alpha<\lambda} \Sigma^\mu_\alpha$ under recursive enumerable conjunctions $\bigwedge_{i<\omega}$, when λ is a limit.

We say that $A \subseteq \omega$ is $\Sigma^\mu_{\alpha+1}$-definable iff there is a $\Sigma^\mu_{\alpha+1}$-formula $n \in \mu x X.\varphi$ such that $A = \{n \in \omega \mid n \in \mu x X.\varphi\}$. If λ is a limit ordinal, $A \subseteq \omega$ is Σ^μ_λ-definable iff there is a Σ^μ_λ-formula $\bigvee_{i\in\omega} n \in \mu x X.\psi_i$ such that $A = \{n \in \omega \mid \bigvee_{i\in\omega} n \in \mu x X.\psi_i\}$.

It will also be useful to consider formulas in a kind of prenex normal form. Lubarsky [12] showed that all finite μ-formulas can be put in the form:

$$\tau_n \in \mu x_n X_n.\tau_{n-1} \in \nu x_{n-1} X_{n-1}.\tau_{n-2} \in \mu x_{n-2} X_{n-2}. \ldots \tau_1 \in \eta x_1 X_1.\varphi,$$

with φ first-order formula.

Bradfield *et al.* [9, 8] extended Lubarsky's normal form to transfinite formulas. A transfinite μ-formula is in normal form iff it is:

- a finite μ-formula in normal form;
- an infinite disjunction or conjunction of μ-formulas in normal form; or
- a formula of the form $\tau \in \eta x X.\varphi$ where φ is in normal form.

Bradfield *et al.* show that all transfinite μ-formulas are equivalent to one in normal form by induction of the construction of formulas and Lubarsky's result.

5 μ-definability implies feedback computability

In this section, we define an evaluation function eval for Σ^μ_α-formulas using α-feedback machines. The function eval receives the code of a formula φ (along with some auxiliary input) and outputs 1 iff φ is true. To evaluate a given formula we will decompose it and check it by parts.

Feedback will be used in two places. First, to check quantifiers, infinite disjunctions, and infinite conjunctions. For example, we can imagine a program which evaluates $\varphi(n)$ for all $n \in \omega$ and stops when it finds some n such that $\varphi(n)$ fails; the formula $\forall x.\varphi(x)$ is true iff this program does not stop. We can verify this with a 0-freezing query. Second, to evaluate fixed-point formulas. eval

β-freezing on input φ will be (roughly) equivalent to φ being false; a $(\beta+1)$-freezing query tells us if φ is true.

We will make heavy use of Kleene's Recursion Theorem in the proof. It holds for α-feedback computability, by the standard proof.

Theorem 2. *Let $A \subseteq \omega$ and $\alpha < \omega_1^{\mathrm{ck}}$. If A is Σ^μ_α-definable then A is α-feedback semi-computable.*

See Appendix A for a proof of Theorem 2.

6 Feedback computability implies μ-definability

In this section, we show that α-feedback semi-computable sets are Σ^μ_α-definable. To do so, we show that the graph of the freezing oracle f_α is Σ^μ_α-definable. The heart of this proof is Proposition 1, but we need to take care when stating the inductive definitions: μ-formulas can only have finitely many free set variables. We show how we can overcome this technicality using an encoding for α and only two set variables.

Theorem 3. *Let $A \subseteq \omega$ and $\alpha < \omega_1^{\mathrm{ck}}$. If A is αth-order feedback semi-computable then A is Σ^μ_α-definable.*

Proof. Fix $\alpha < \omega_1^{\mathrm{ck}}$. We prove that the graph of f_α is Σ^μ_α-definable for α a successor ordinal. The case for limit ordinals is similar; we will indicate the changes in the appropriate place.

We use computation histories for computations $\langle e\rangle^\alpha(n)$. A computation history encodes a finite initial segment of a computation. Such encoding is possible because at each step, the computation only needs a finite amount of memory. Any such encoding is good as long that we can decide if some natural number encodes a computation history or not. We also require that, from a computation history h, we can recover the index e, the initial input n and the step-by-step computation of $\langle e\rangle^\alpha(n)$ up to a finite time. Note that a freezing oracle query counts as only one step here. A computation will be halting iff it has a finite computation history which halts (this history will have no extension). A computation diverges iff there are computation histories of unbounded length. For $\beta \leq \alpha$, a computation β-freezes iff there is a sequence of histories $\{h_i\}_{i\in\omega}$ such that: $e_0 = e$ and $n_0 = n$; h_i is a computation history for $\langle e_i\rangle^\alpha(n_i)$ ending in a query to f_{β_i} about $\langle e_{i+1}\rangle^\alpha(n_{i+1})$; and $\beta = \limsup\{\beta_i + 1 \mid i \in \omega\}$.

We will first give a wrong proof: we define the graph of the freezing oracle f_β by a $\Sigma^\mu_{\beta+1}$-formula χ_β with free variables $X_{\beta+1}, X_{\beta+2}, \ldots, X_\alpha$. For $\beta \geq \omega\cdot 2$, $\chi_\beta(x)$ is not a well-formed μ-formula. We explain later how to modify the χ_β and obtain proper μ-formulas.

The graph of the freezing oracle f_0 is defined by the formula $\chi_0(x)$ defined as follows. $\chi_0(x)$ is of the form $x \in \mu x_0 X_0.\varphi_0(x_0)$. $\varphi_0(x_0)$ is true if $x_0 = \langle e, n, \downarrow\rangle$ and there is a computation history witnessing that $\langle e\rangle^\alpha(n)$ halts; or if $x_0 = \langle e, n, \uparrow_0\rangle$ and for all k there is a computation history of length k for $\langle e\rangle^\alpha(n)$ which is non-halting. Note that X_β is free in χ_0 for all $0 < \beta \leq \alpha$, since the only fixed-point operator in χ_0 is $\mu x_0 X_0$.

Suppose the formula χ_β defines the graph of f_β. We now define the graph of the freezing oracle $f_{\beta+1}$ with a formula $\chi_{\beta+1}$. The formula $\chi_{\beta+1}(x)$ is of the form $x \in \mu x_{\beta+1} X_{\beta+1}.\varphi_{\beta+1}(x_{\beta+1})$. Here, $\varphi_{\beta+1}(x_{\beta+1})$ is true if either $\chi_\beta(x_{\beta+1})$ holds, or $x_{\beta+1} = \langle e, n, \uparrow_{\{} \beta+1\}\rangle$ and there is a sequence of histories $\{h_i\}_{i\in\omega}$ witnessing that $\langle e\rangle^\alpha(n)$ β'-freezes for β' with $\beta \leq \beta' < \alpha$. The last disjunct can be computed by a greatest fixed-point: start with the set of all finite sequences of histories and trim off the sequences which cannot be the initial segments of such an witnessing history $\{h_i\}_{i\in\omega}$; the resulting tree is non-empty only if such sequence of histories exist.

If we have defined χ_β for all $\beta < \lambda$, we define $\chi_\lambda(x)$ by $\bigvee_{\beta<\lambda} \chi_\beta(x)$. When proving that λ-feedback computable are Σ^μ_λ-definable, we define χ_λ similarly, but omit the references for $\beta' > \beta$ in each χ_β.

We use α's encoding on ω to substitute references for X_β with references to X_α. We can do so because a query to f_β is a query to f_α where we ignore the output if it is $\uparrow_{\beta'}$ for some $\beta' > \beta$. The corrected formula is a well-formed μ-formula equivalent to the non-well-formed formula. This finishes the definition of the freezing oracle f_α.

Now, suppose A is αth order feedback semi-computable via a machine with index e. For all $\beta < \alpha$, we can recover f_β from f_α using the computable encoding of α. Thus $n \in A$ iff there is a computation history for e starting with input n where the computation halts; here, the computation histories can consult the freezing oracles $\{f_\beta\}_{\beta\leq\alpha}$.

Acknowledgments. This study was funded by FWF project TAI-797.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Ackerman, N.L., Freer, C.E., Lubarsky, R.S.: Feedback computability on cantor space **15**(2), 7:1–7:18
2. Ackerman, N.L., Freer, C.E., Lubarsky, R.S.: Feedback Turing Computability, and Turing Computability as Feedback. In: 30th Annual ACM/IEEE Symposium on Logic in Computer Science. pp. 523–534. https://doi.org/10.1109/LICS.2015.55
3. Ackerman, N.L., Freer, C.E., Lubarsky, R.S.: An introduction to feedback turing computability **30**(1), 27–60. https://doi.org/10.1093/logcom/exaa002
4. Aguilera, J.P., Lubarsky, R.S.: Feedback hyperjump **31**(1), 20–39. https://doi.org/10.1093/logcom/exaa085
5. Aguilera, J.P., Soto, M.: Type-2 feedback computability
6. Bradfield, J.C.: Fixpoints, games and the difference hierarchy **37**(1), 1–15. https://doi.org/10.1051/ita:2003011
7. Bradfield, J.C.: The modal mu-calculus alternation hierarchy is strict **195**(2), 133–153. https://doi.org/10.1016/S0304-3975(97)00217-X
8. Bradfield, J.C., Duparc, J., Quickert, S.: Fixpoint alternation and the Wadge hierarchy, https://www.julianbradfield.org/Research/fixwadge.pdf

9. Bradfield, J.C., Duparc, J., Quickert, S.: Transfinite extension of the mu-calculus. In: Ong, L. (ed.) Computer Science Logic, vol. 3634, pp. 384–396. Springer Berlin Heidelberg. https://doi.org/10.1007/11538363_27
10. Kozen, D.: Results on the propositional μ-calculus **27**(3), 333–354. https://doi.org/10.1016/0304-3975(82)90125-6
11. Lubarsky, R.S.: ITTMs with feedback. In: Schlinder, R. (ed.) Ways of Proof Theory, pp. 341–354
12. Lubarsky, R.S.: μ-definable sets of integers **58**(1), 291–313. https://doi.org/10.2307/2275338
13. Möllerfeld, M.: Generalized inductive definitions: The μ-calculus and Π^1_2-comprehension
14. Moschovakis, Y.: Descriptive Set Theory, Mathematical Surveys and Monographs, vol. 155. American Mathematical Society. https://doi.org/10.1090/surv/155
15. Rogers, H.: Theory of Recursive Functions and Effective Computability
16. Welch, P.: $G_{\delta\sigma}$-games and generalized computation, https://arxiv.org/abs/1509.09135

A Proof of Theorem 2

Let $A \subseteq \omega$ and $\alpha < \omega_1^{\mathrm{ck}}$. We show that, if A is Σ^μ_α-definable, then A is α-feedback semi-computable.

We begin by defining an evaluation function `eval` for Σ^μ_α-formulas. We then prove by induction on $\beta \leq \alpha$ that the Σ^μ_β-definable sets are α-feedback semi-computable.

For this proof, we work with a fixed set of set variables $\{X_i \mid i \in \omega\}$. We also assume that the μ-formulas are in normal form. We do not consider formulas with free *number* variables. `eval` is defined by recursion on the structure of the Σ^μ_α-formulas: we begin at the first order formulas and go up level-by-level.

The function $\texttt{eval}(\varphi, s)$ takes as input a formula φ, and a sequence s of natural numbers. The sequence s is a sequence of indices of (possibly partial) characteristic function of sets. If $i < \mathrm{length}(s)$ then s_i denotes the index in the ith position of s. If $i \geq \mathrm{length}(s)$, then s_i is the index for the characteristic function of the empty set. s will be useful when evaluating the fixed-point operators.

We define `eval` for first-order formulas, along with auxiliary functions `exists` and `forall`:

- $\texttt{eval}(t = t', s) := \begin{cases} 1, \text{ if } t = t' \\ 0, \text{ otherwise} \end{cases}$
- $\texttt{eval}(t \in X_i, s) := \begin{cases} 1, \text{ if } \langle s_i \rangle^\alpha(t) = 1 \\ 0, \text{ otherwise} \end{cases}$
- $\texttt{eval}(\neg\psi, s) := \begin{cases} 1, \text{ if } \texttt{eval}(\psi, s) = 0 \\ 0, \text{ otherwise} \end{cases}$
- $\texttt{eval}(\psi \wedge \theta, s) := \begin{cases} 1, \text{ if } \texttt{eval}(\psi, s) = \texttt{eval}(\theta, s) = 1 \\ 0, \text{ otherwise} \end{cases}$
- $\texttt{eval}(\psi \vee \theta, s) := \begin{cases} 1, \text{ if } \texttt{eval}(\psi, s) = 1 \text{ or } \texttt{eval}(\theta, s) = 1 \\ 0, \text{ otherwise} \end{cases}$

- $\texttt{forall}(\psi(x), s, i) := \begin{cases} 0, & \text{if } \texttt{eval}(\psi(i), s) = 0 \\ \texttt{forall}(\psi(x), s, i+1), & \text{otherwise} \end{cases}$
- $\texttt{eval}(\forall x.\psi, s) := \begin{cases} 1, \text{ if } \texttt{forall}(\psi(x), s, 0) \text{ diverges} \\ 0, \text{ otherwise} \end{cases}$
- $\texttt{exists}(\psi(x), s, i) := \begin{cases} 1, & \text{if } \texttt{eval}(\psi(i), s) = 1 \\ \texttt{exists}(\psi(x), s, i+1), & \text{otherwise} \end{cases}$
- $\texttt{eval}(\exists x.\psi, s) := \begin{cases} 1, \text{ if } \texttt{exists}(\psi(x), s, 0) \text{ converges} \\ 0, \text{ otherwise} \end{cases}$

On `forall` and `exists`, $\psi(i)$ is obtained by substituting the indicated number variable x by i.

We similarly define `eval` on infinitary formulas using auxiliary functions `conjunction` and `disjunction`:

- $\texttt{disjunction}(\bigvee_{i\in\omega} \psi_i, s, i) := \begin{cases} 1, & \text{if } \texttt{eval}(\psi_i, s) = 1 \\ \texttt{disjunction}(\bigvee_{i\in\omega} \psi_i, s, i+1), & \text{otherwise} \end{cases}$
- $\texttt{eval}(\bigvee_{i\in\omega} \psi_i, s) := \begin{cases} 1, \text{ if } \texttt{disjunction}(\bigvee_{i\in\omega} \psi_i, s, 0) \text{ converges} \\ 0, \text{ otherwise} \end{cases}$
- $\texttt{conjunction}(\bigwedge_{i\in\omega} \psi_i, s, i) := \begin{cases} 0, & \text{if } \texttt{eval}(\psi(i), s) = 0 \\ \texttt{conjunction}(\bigwedge_{i\in\omega}, s, i+1), & \text{otherwise} \end{cases}$
- $\texttt{eval}(\bigwedge_{i\in\omega} \psi_i, s) := \begin{cases} 0, \text{ if } \texttt{conjunction}(\bigwedge_{i\in\omega} \psi_i, s, 0) \text{ converges} \\ 1, \text{ otherwise} \end{cases}$

Remember that as we only allow recursively enumerable conjunctions and disjunctions, we can recover ψ_i from a code of $\bigwedge_{i\in\omega} \psi_i$ or $\bigvee_{i\in\omega} \psi_i$.

We now extend `eval` to formulas with fixed-points. Suppose $t \in \mu x X.\psi$ is a $\Sigma^\mu_{\beta+1}$-formula, define:

$$\texttt{eval}(t \in \mu x_i X_i.\psi, s) := \begin{cases} 1, & \text{if } \texttt{eval}(\psi(t), s[X_i := \emptyset]) = 1 \\ & \text{or } \texttt{eval}(\psi(t), s[X_i := \mu x_i X_i.\psi]) = 1 \\ \uparrow_\beta, & \text{otherwise} \end{cases}$$

Where $s[X := \emptyset]$ is obtained by putting an index for the empty set in the ith position of s, and $s[X_i := \mu x X.\psi]$ is obtained by putting an index for $\lambda n.\texttt{eval}(n \in \mu x_i X_i.\psi)$ in the ith position of s. If s becomes a longer sequence by this procedure, fill the unused positions of s with indexes for the empty set.

If we have extended `eval` to Σ^μ_β-formulas, extend it to Π^μ_β-formulas by:

$$\texttt{eval}(t \in \nu x X.\psi, s) := \begin{cases} 1, \text{ if } \texttt{eval}(t \in \mu x X.\neg\psi(\neg X), s) \ \beta\text{-freezes} \\ 0, \text{ otherwise} \end{cases}$$

This finishes the definition of `eval`.

We prove by bounded induction on $\beta \leq \alpha$ that the Σ^μ_β-definable sets are α-feedback semi-computable. We slightly strengthen the induction hypothesis to show that, for $\beta < \alpha$, Π^μ_β-definable sets are α-feedback computable.