Degrees of isolated paths of trees of countable width

Reese Johnston, University of Wisconsin-Madison

http://www.math.wisc.edu/~rwjohnston

January 28, 2016
Definitions

- Given a language $\mathcal{L}$ containing the symbol $\in$, an $\mathcal{L}$-formula $\varphi$ is $\Delta^0_0$ if all quantifiers appearing in $\varphi$ are bounded (i.e., of the form $\forall x \in y$ or $\exists x \in y$).

- $\varphi$ is $\Sigma^0_1$ if it is of the form $\exists x \psi$, where $\psi$ is $\Delta^0_0$.

- We operate within the universe $L_{\omega_1}$. A set $X \subseteq L_{\omega_1}$ is c.e. if $X$ is definable by a $\Sigma^0_1(L_{\omega_1})$ formula (a $\Sigma^0_1$ formula with parameters in $L_{\omega_1}$). $X$ is computable if both $X$ and $\overline{X}$ are c.e.
For the most part, we will rely on an uncountable version of the Church-Turing Thesis; we think of this as running a program that is allowed to manipulate countably infinite objects and run for any countable number of stages.
For our purposes, a tree is a computable subset of $2^{<\omega_1}$ that is downward-closed under the usual ordering. A path through a tree $T$ is a set $X$ so that $X \upharpoonright \alpha \in T$ for each $\alpha < \omega_1$. In the standard setting, every tree has a low path, and isolated paths are computable. In the $\omega_1$ setting, this is not the case.
For our purposes, a tree is a computable subset of $2^{<\omega_1}$ that is downward-closed under the usual ordering. A path through a tree $T$ is a set $X$ so that $X \upharpoonright \alpha \in T$ for each $\alpha < \omega_1$.

In the standard setting, every tree has a low path, and isolated paths are computable. In the $\omega_1$ setting, this is not the case.
Trees

For our purposes, a tree is a computable subset of $2^{<\omega_1}$ that is downward-closed under the usual ordering. A path through a tree $T$ is a set $X$ so that $X \upharpoonright \alpha \in T$ for each $\alpha < \omega_1$.

In the standard setting, every tree has a low path, and isolated paths are computable. In the $\omega_1$ setting, this is not the case. Binary trees in the $\omega_1$ setting behave more like infinitely branching trees in the standard setting.
Trees

For our purposes, a tree is a computable subset of $2^{<\omega_1}$ that is downward-closed under the usual ordering. A path through a tree $T$ is a set $X$ so that $X \upharpoonright \alpha \in T$ for each $\alpha < \omega_1$.

In the standard setting, every tree has a low path, and isolated paths are computable. In the $\omega_1$ setting, this is not the case.

Binary trees in the $\omega_1$ setting behave more like infinitely branching trees in the standard setting.

A tree $T$ has countable width if $T \cap 2^\alpha$ is countable for every $\alpha$. 
Theorem

There is a computable tree $T$ of countable width so that $T$ has exactly one path, and that path is equivalent to $\emptyset'$. 

Lemma

There is a computable Aronszajn tree; i.e., a tree of countable width and uncountable height, having no path.
Rough Proof of Theorem

We build the tree level by level, keeping track of a current guess at the final path $P$. $\emptyset'$ will be encoded into $P$. 
Rough Proof of Theorem

We build the tree level by level, keeping track of a current guess at the final path $P$. $\emptyset'$ will be encoded into $P$. Whenever we commit to encoding a bit of $\emptyset'$, we set up a branch point. Over one branch, we let $P$ continue, with the bit of $\emptyset'$ coded in. Over the other branch, we start building a computable Aronszajn tree.
Rough Proof of Theorem

We build the tree level by level, keeping track of a current guess at the final path $P$. $\emptyset'$ will be encoded into $P$. Whenever we commit to encoding a bit of $\emptyset'$, we set up a branch point. Over one branch, we let $P$ continue, with the bit of $\emptyset'$ coded in. Over the other branch, we start building a computable Aronszajn tree. If, at any stage, the approximation to $\emptyset'$ changes, we abandon the first branch and switch $P$ to the first available node in the Aronszajn tree being grown over the other branch.
Question
How far up can this go?

Upper Bound
Certainly an isolated path of any computable tree (countable width or otherwise) can only be $\Delta^1_1$. 
Hyperarithmetic Hierarchy

(Definition 1): Mimicking the standard setting, take the hyperarithmetic sets to be those below $0^{(\alpha)}$, where $\alpha$ ranges over computable ordinals.
Hyperarithmetic Hierarchy

(Definition 1): Mimicking the standard setting, take the hyperarithmetic sets to be those below $0^{(\alpha)}$, where $\alpha$ ranges over computable ordinals.

Because well-ordering is a $\Pi^0_1$ property in $\omega_1$, by this definition there are nonhyperarithmetic sets that are hyperarithmetic in $\emptyset'$. 
Hyperarithmetic Hierarchy

(Definition 2): Take $HYP$ to be the minimal class containing $0$ and closed under the map $d \to d^{(\alpha)}$, where $\alpha$ is any ordinal computable in $d$. It turns out that it is possible to describe the first ordinal that doesn't appear in $HYP$ in a $\Delta^1_1$ way. So we need to look further afield.
(Definition 2): Take $HYP$ to be the minimal class containing $0$ and closed under the map $d \to d^{(\alpha)}$, where $\alpha$ is any ordinal computable in $d$.

It turns out that it is possible to describe the first ordinal that doesn’t appear in $HYP$ in a $\Delta^1_1$ way. So we need to look further afield.
Let $L_0 = HC$, and otherwise define the $L$ hierarchy as usual. For any $\alpha < \omega_2$ and $n < \omega$, let $M_{\omega\alpha+n} = \Delta_n(L_\alpha)$.

Say $\alpha$ is an $M$-index if $(M_{\alpha+1} \setminus M_\alpha) \cap 2^{\omega_1} \neq \emptyset$. A master code for $\alpha$ is an $A \subseteq \omega_1$ so that $M_{\alpha+1} \cap 2^{\omega_1}$ is exactly the sets computable from $A$.

Let $0^{(\alpha)}$ be the degree of the master code for the $\alpha$th $M$-index.
Definition

Let $\rho_0^0$ be the first nonhyperarithmetic ordinal (under the expanded definition). For any $\alpha > 0$, let $\rho_\alpha^0$ be the first ordinal not hyperarithmetic in $0^{(\rho_\beta^0)}$ for any $\beta < \alpha$.

For any $\alpha, \gamma > 0$, let $\rho_\alpha^\gamma$ be the $\alpha$th ordinal $\delta$ so that $\delta = \rho_\delta^\epsilon$ for every $\epsilon < \gamma$.

Theorem

Let $\eta$ be the least ordinal so that $\eta = \rho_0^\eta$. Then, for every $\beta < \eta$, there is a tree of countable width $T$ with unique path $X$ so that $X \geq_T 0^{(\beta)}$. 
A *companion map* for a tree \( T \) is a function \( f : \omega_1 \rightarrow T \) with the following properties.

- \( f \) is continuous; that is, for any limit ordinal \( \alpha \),
  \[ f(\alpha) = \lim_{\beta<\alpha} f(\beta). \]
- \( f \) is order-respecting; if \( f(\alpha) \prec f(\beta) \), then \( \alpha < \beta \).
- The range of \( f \) is unbounded along every path through \( T \).

An *augmented tree* is a tree \( T \) equipped with a companion map. An augmented tree is computable if both the tree and its companion map are computable.
Lemma I

Let $T$ be a computable augmented tree with $X$ its unique path. Then there exists, uniformly in an index for $T$, a computable augmented tree $T'$ with unique path $Y$ so that $Y \equiv_T X'$. Furthermore, if $T$ has countable width, so does $T'$.

Lemma II

Let $U$ be a computable augmented tree with unique path $X$. Let $F : U \rightarrow \omega_1$ be a computable function so that for every $\sigma$ along $X$, $F(\sigma)$ is an index for a computable augmented tree with a unique path $X_\sigma$. Then there exists, uniformly in an index for $U$ and $F$, a computable augmented tree $T$ with unique path $Y$ so that $Y$ uniformly computes $X$ and every $X_\sigma$. Furthermore, if $U$ has countable width and the tree with index $F(\sigma)$ has countable width for every $\sigma$ along $X$, then $T$ also has countable width.
The Idea

We can use Lemmas I and II to mimic the hyperarithmetic hierarchy; Lemma I handles successor steps, while Lemma II handles limit steps computable in preexisting paths. But this only gets us up to (and not including) \( \rho_0 \). We would like to be able to use Lemma II to “join” across all of the paths produced this way, but the set of these is too complicated.
The Idea

We can use Lemmas I and II to mimic the hyperarithmetic hierarchy; Lemma I handles successor steps, while Lemma II handles limit steps computable in preexisting paths. But this only gets us up to (and not including) $\rho_0^0$. We would like to be able to use Lemma II to “join” across all of the paths produced this way, but the set of these is too complicated.
What if we could restrict the application of Lemma II to make it easier?
Kleene’s $O$ in $\omega_1$

$R$ ("Relaxed $O$") is the smallest set satisfying the following closure properties.

- $0 \in R$, $H_0 = \emptyset$, and $|0| = 0$.
- If $a \in R$, then $b = \langle \text{succ}, a \rangle \in R$, $H_b = H'_a$, and $|b| = |a| + 1$.
- Let $\{e\}$ be a functional, $a \in R$. Then $b = \langle \omega_1, a, e \rangle \in R$. Let $X = \{x | \{e\}^{H_a}(x) \downarrow \}$. $H_b = \bigoplus_{x \in X} H_{\{e\}^{H_a}(x)}$, and $|b| = \sup_{x \in X} |\{e\}(x)|$. 

This covers everything $O$ does, but has the advantage of being computable. The downside is that it’s very difficult to do anything with $R$. 

Reese Johnston, University of Wisconsin-Madison

Degrees of isolated paths of trees of countable width
Kleene’s $\mathcal{O}$ in $\omega_1$

$\mathcal{R}$ ("Relaxed $\mathcal{O}$") is the smallest set satisfying the following closure properties.

- $0 \in \mathcal{R}$, $H_0 = \emptyset$, and $|0| = 0$.
- If $a \in \mathcal{R}$, then $b = \langle \text{succ}, a \rangle \in \mathcal{R}$, $H_b = H'_a$, and $|b| = |a| + 1$.
- Let $\{e\}$ be a functional, $a \in \mathcal{R}$. Then $b = \langle \omega_1, a, e \rangle \in \mathcal{R}$. Let $X = \{x|\{e\}^{H_a}(x) \downarrow\}$. $H_b = \bigoplus_{x \in X} H_{\{e\}^{H_a}(x)}$, and $|b| = \sup_{x \in X} |\{e\}(x)|$.

This covers everything $\mathcal{O}$ does, but has the advantage of being computable. The downside is that it’s very difficult to do anything with $\mathcal{R}$.
For any $a \in \mathcal{R}$, let $a \downarrow$ be the set of members of $\mathcal{R}$ that appear in the expansion of $a$ (including $a$). Let $\mathcal{O}_a$ be the set of members of $\mathcal{R}$ constructible using only $\omega_1$-sequences computable in $H_b$ for $b \in a \downarrow$. $\mathcal{O}_a$ is computable in $H'_a$. 
Getting past HYP

There is a computable function \((a, e) \rightarrow e'\) so that, for any pair, the following holds:

- \(\{e'\}^{H''_a}\) is total and has range entirely contained in \(O_a\), and
- \(\{e'\}^{H''_a}(x) = \{e\}^{H_a}(x)\) whenever this exists and is in \(O_a\).
The notation set $C^X$

Let $X$ be the unique path in some fixed computable augmented tree $T^X$. We define a set $C^X$, together with functions $T : C^X \rightarrow \omega_1$ and $o : C^X \rightarrow R$, inductively as follows:

1. $0 \in C^X$. $T(0) = T^X$, and $o(0) = 0$.

2. If $a \in C^X$, then $b = \langle \text{succ}, a \rangle \in C^X$. $T(b) = T(a)'$, as constructed by Lemma 1. $o(b) = \langle \text{succ}, o(a) \rangle$.

3. If $a \in C^X$ and $e \in \omega_1$, then $b = \langle \omega_1, a, e \rangle \in C^X$. Let $e'$ be the image of $(o(a), e)$ under the previously specified map. Then $T(b)$ is the tree constructed by Lemma 2, taking $T(a)''$ as the base tree (applying Lemma 1 twice to $T(a)$) and applying $\{e'\}$ along branches to compute $F$. 
The notation set $C^X$

$C^X$ is computable. By the recursion theorem, $T$ and $o$ are also computable.
The notation set $C^X$

$C^X$ is computable. By the recursion theorem, $T$ and $o$ are also computable.
We can then apply Lemma 2 to a “tree” with a computable path, taking $F$ to be the function enumerating the members of $C^X$ along each branch. The result is a tree $T$ with a unique path $Y$, so that $Y$ computes every set hyperarithmetic in $X$. 

Taking $X = \emptyset$, this $Y$ is $0^{(\rho_0)}$. 
The notation set $\mathcal{C}^X$

$\mathcal{C}^X$ is computable. By the recursion theorem, $T$ and $o$ are also computable.

We can then apply Lemma 2 to a “tree” with a computable path, taking $F$ to be the function enumerating the members of $\mathcal{C}^X$ along each branch. The result is a tree $T$ with a unique path $Y$, so that $Y$ computes every set hyperarithmetic in $X$.

Taking $X = \emptyset$, this $Y$ is $0^{(\rho_0^0)}$. 
Thank you.