

# Self embeddings of computable trees

Stephen Binns  
Bjørn Kjos-Hanssen  
Manuel Lerman  
James H. Schmerl  
Reed Solomon  
Department of Mathematics  
University of Connecticut

January 6, 2006

## Abstract

It is shown that  $0''$  is sufficient to compute a nontrivial self-embedding of any infinite computable tree. Furthermore, we show that there exists a tree for which  $0''$  is necessary to compute such an embedding. We prove that every infinite computable tree must have either a computable infinite chain or an infinite  $\Pi_1^0$  antichain, and that this is the best result possible. A corollary is that  $\text{WKL}_0$  is too weak to prove that every binary branching tree has an infinite chain or an infinite antichain.

## 1 Introduction

We explore here some issues in the realm of computable trees. In particular we look at work done by Downey, Miller and Lempp [ ] on the effectiveness of the Dushnik-Miller theorem and Herrmann[1] on chains and antichains in computable partial orders.

The Dushnik-Miller theorem, a theorem of classical mathematics, states that every linear order has a nontrivial self-embedding. Downey et al. prove that there exist computable linear orders with no computable self-embeddings. In fact a Turing complete c.e. set can be

encoded into a linear order so that any nontrivial self-embedding computes  $0'$ . We investigate the same questions as they apply to nontrivial self-embeddings of countable trees rather than linear orders.

There are many ways to define the term *tree*. For our purposes a tree will be a partial order  $\preceq$  on a subset of  $\mathbb{N}$  with a minimum element (denoted  $\lambda$ ) and such that every set of the form  $\{m : m \preceq n\}$  is finite. Trees of this sort are classically isomorphic to a subtree of  $\omega^{<\omega}$ . Computable trees are trees whose order relation is computable. Note that the domain of a tree is computable as a consequence because the domain is just  $\{n : \lambda \preceq n\}$ . Also note that while every computable tree is isomorphic to a subtree of  $\omega^{<\omega}$ , it is not necessarily computably isomorphic to such a subtree. This can be seen by observing the fact that the successor relation of a computable tree is not necessarily computable, while the successor relation of any subtree of  $\omega^{<\omega}$  is computable.

There are also two natural ways to define the term *nontrivial*. For linear orders a self-embedding is not the identity map if and only if it is not surjective. For trees this is clearly not the case as it is easy to come up with a surjective, non-identity map of (for example)  $\omega^{<\omega}$ . We could therefore define the term nontrivial in two different ways. For our purposes we choose nontrivial to mean not surjective. We also define a self-embedding to be *weakly nontrivial* if it is not the identity map. We will primarily concern ourselves with nontrivial self-embeddings and prove results for weakly nontrivial self-embeddings only to present the sharpest possible results.

The basic result we prove in Section 2. That is that every infinite computable tree has a self embedding computable from  $0''$ . (A simple but unstated consequence of the proof is the analogue for trees of the Dushnik-Miller theorem.) The proof describes three different types of trees - those with an  $\omega$ -node; those with no  $\omega$ -node but with an isolated path; and those with neither of the previous but which embed  $2^{<\omega}$  (an  $\omega$ -node is an element of the tree with infinitely many immediate successors). We show that for the first two categories  $0'$  is sufficient to compute a self-embedding and for the third category we use  $0''$ . This categorisation of trees proves to be useful throughout the paper. The rest of the subsection is devoted to showing that these are the best possible results. That is we construct trees of types 1 and 2 such that any nontrivial self-embedding of one of these trees computes  $0'$  and a tree of type 3 all of whose nontrivial self-embeddings compute  $0''$ . With these results we completely answer all basic questions analogous

to the Downey Lempp Miller result.

We then turn our attention to a related question. Is the property of having no computable self-embedding a property of the isomorphism type of the tree? That is, does there exist a computable tree  $T$  such that no tree  $S$  classically isomorphic to  $T$  has a computable nontrivial self-embedding? We answer in the affirmative for all three types of tree. These second set of results are somewhat less complete than the first set. We do not not prove for example that there is a type 1 tree such that any classically isomorphic tree has the property that any nontrivial self-embedding computes  $0'$ . There is further work to be done here and we make some observations on this at the end of section 2.

In Section 3 we establish some other computability results for computable trees. In particular we examine a theorem by Herrmann on partial orders in the special case where the partial order is a tree. Herrmann in [1] shows that every infinite partial order has either an infinite  $\Delta_2^0$  chain or an infinite  $\Pi_1^0$  antichain. We show a better result can be obtained if trees rather than general partial orders are considered. As a consequence for reverse mathematics it is shown that the system  $WKL_0$  is not strong enough to prove that every infinite binary branching tree has either an infinite chain or an infinite antichain.

## 2 Definitions

**Definition 2.1.** *A tree is a partial order  $\preceq$  on a subset of  $\mathbb{N}$  with a least element and such that for all  $n$   $\{m : m \preceq n\}$  is finite and linearly ordered by  $\preceq$ . The domain of  $\preceq$  will be denoted  $T$  and elements of  $T$  will be referred to as nodes. The  $\preceq$ -least element of  $T$  will be called the root and will be denoted  $\lambda$ .*

We will be concerned in this paper with computable trees. That is trees for which the set  $\{\langle n, m \rangle : n \preceq m\}$  is computable. The symbols  $\succeq$ ,  $\succ$  and  $\prec$  are as usual.

**Definition 2.2.** *Let  $T$  be a tree.*

- $T(i) := \{n : n \succeq i\}$  — the successor tree of  $i$  with the inherited order.
- $T_s := \{n \in T : n \leq s\}$  with the inherited order.
- $S(m, n) \equiv m$  is a successor of  $n \equiv \forall k[m \succeq k \succ n \implies k = m]$

- The branching function of  $T$   $\text{br} : T \rightarrow \mathbb{N} \cup \{\infty\}$  is the function that takes each node of  $T$  to the number of its successors.
- If  $n \in T$  and  $\lambda = n_0 \preceq n_1 \preceq \cdots \preceq n_m = n$  where  $S(n_{i+1}, n_i)$  for all  $i$ , then  $m$  is the height of  $n$ , denoted  $\text{ht}(n)$ . The height of a tree,  $\text{ht}(T)$ , is the supremum of the height of its nodes.
- A branching node is a node with more than one successor.
- A leaf is a node with no successors.
- $i$  is an  $\omega$ -node if  $\{n : S(n, i)\}$  is infinite
- $i$  is an infinite node if  $T(i)$  is infinite
- $i$  is a maximal infinite node if  $i$  is an infinite node and for all  $j$

$$j \succ i \implies T(j) \text{ is finite.}$$

Note that every maximal infinite node is an  $\omega$ -node, each of whose successor trees is finite.

- A path through  $n \in T$  is an infinite sequence  $\langle x_i \rangle$  of elements of  $T$  such that  $x_0 = \lambda$ ,  $S(x_{i+1}, x_i)$  for all  $i$  and such that  $n = x_j$  for some  $j$ . A path  $X$  is isolated if there is an  $n \in T$  such that  $X$  is the only path through  $n$ .
- An embedding from  $T$  to  $T'$  is an order-preserving injection. The notation  $T \hookrightarrow T'$  indicates such an embedding exists. There are two possible ways to conceive of an embedding's being nontrivial. We say a self embedding  $\varphi$  from  $T$  to  $T$  is nontrivial if it is not surjective. We say  $\varphi$  is weakly nontrivial if it is not the identity map.

**Lemma 2.3.** Every tree  $T$  embeds into  $2^{<\omega}$  and the embedding is computable in  $0'$ .

*Proof.* Given an arbitrary  $n \in T$  we will compute the image of  $n$  as follows.  $0'$  computes the successor relation of  $T$  (which is explicitly  $\Pi_1^0$ ) so we can use  $0'$  to compute the sequence

$$\lambda = n_0 \prec n_1 \prec \cdots \prec n_k = n,$$

where  $S(n_{i+1}, n_i)$  for all  $i$ . The image of  $n$  will then be

$$1^{n_0} * 0 * 1^{n_1} * 0 * \dots * 1^{n_k} * 0.$$

(where  $1^m$  denotes the string of  $m$  ones). It is straightforward to confirm that this gives an embedding. □

**Lemma 2.4.** *If  $T$  is an infinite tree and it has no maximal infinite node then it has a path.*

*Proof.* Suppose  $T$  is an infinite tree and has no maximal infinite node. If  $T$  has no  $\omega$ -node then it is finitely branching and has a path by König's Lemma. So it must have an  $\omega$ -node  $n_0$ , and a successor  $n_1$  of  $n_0$  such that  $T(n_1)$  is infinite (otherwise  $n_0$  would be a maximal infinite node). If  $T(n_1)$  has a path then so does  $T$  and we are done. So  $T(n_1)$  must have an  $\omega$ -node  $n_2$  which has a successor  $n_3$  such that  $T(n_3)$  is infinite. Iterating in this way gives us a path  $n_0 \prec n_1 \prec n_2 \prec \dots$ .  $\square$

**Theorem 2.5.** *Every infinite computable tree has a nontrivial self embedding computable in  $0''$ .*

*Proof.* The proof will be in cases. The first two cases need only  $0'$ .

*Case 1:*  $T$  has a maximal infinite node  $n$ . Let  $n_0 < n_1 < \dots$  be the successors of  $n$  (computable from  $0'$ ). Kruskal's theorem gives us a  $k$  such that

$$\forall m \geq k \exists^\infty s \geq m \ T(n_m) \hookrightarrow T(n_s).$$

Fix this  $k$ . We will define a  $0'$ -computable embedding  $\varphi$  such that  $\forall m \ \varphi(m) \neq m$  if and only if  $m \succeq n_l$  for some  $l \geq k$ . Furthermore, for all  $l \geq k$ ,  $\varphi(n_l) = n_j$  for some  $j > l$ .

To define  $\varphi$ , fix  $j \geq k$  and suppose that  $\varphi(n_i)$  has been defined for all  $i < j$ . Use  $0'$  to find an  $s$  and  $t$  such that

1.  $n_t > \max\{\varphi(n_i) : i < j\}$ ,
2.  $T_s(n_j)$  embeds into  $T_s(n_t)$ ,
3.  $\forall s' > s \ T_{s'}(n_j) = T_s(n_j)$ .

Such an  $s$  and  $t$  exist as  $n$  is a maximal infinite node and hence  $T(n_i)$  is finite for each  $i$ . We then extend  $\varphi$  to include the embedding of  $T_s(n_j)$  into  $T_s(n_t)$ .

*Case 2:*  $T$  has no maximal infinite node and an isolated path  $X$ .

Fix  $i \in T$  such that there is only one infinite path extending  $i$ . By Lemma 2.4 if  $j \succ i$  is any node on  $X$  there is exactly one successor of  $j$ , say  $j'$ , such that  $T(j')$  is infinite, viz the successor that is on  $X$ . Therefore the only possible  $\omega$ -nodes extending  $i$  lie on  $X$ . But now if  $m \succ i$  is an  $\omega$ -node, all but one of its successor trees are finite and we are essentially in case 1 (if  $\{n_l : l \in \omega\}$  are the successor nodes of  $m$ , and  $k$  is as in case 1, we take  $k'$  such that  $T(n_l)$  is finite for all  $l \geq k'$  and then use  $\max\{k, k'\}$  instead of  $k$  in case 1). So we can assume that there are no  $\omega$ -nodes above  $i$ .

But now we can compute  $X$  using  $0'$ . Suppose  $X$  is known up to the node  $x \succeq i$ . Using  $0'$  we find an  $s$  such that every successor of  $x$  is on the tree  $T_s(x)$ . That is, we find an  $s$  such that

$$\forall n \succ x \exists m \in T_s(x) \ m \neq x \text{ and } n \succeq m.$$

Such an  $s$  must exist as  $T(x)$  is finitely branching by our assumption above. Now search for a successor  $x'$  of  $x$  and a  $t$  such that

$$\forall v \geq t \ [v \succ x \implies v \succeq x'].$$

Such an  $x'$  and  $t$  must exist as  $x$  has exactly one path through it. So  $x'$  lies on the path  $X$  which is thus computable using iteration from  $0'$ .

Now consider the sequence  $i = x_0 \prec x_1 \prec x_2 \prec \dots$  such that for all  $i$ ,  $x_i$  is on the path  $X$  and  $S(x_{i+1}, x_i)$ . We now can apply Kruskal's theorem to the sequence of finite trees  $T(x_i) \setminus T(x_{i+1})$  and use an argument similar to case 1.

*Case 3:*  $T$  has no maximal infinite node and no isolated path.

We use  $0''$  to embed  $2^{<\omega}$  into  $T$  and then we use  $0'$  to embed  $T$  into  $2^{<\omega}$  as per Lemma 2.3. We define an embedding  $\varphi : 2^{<\omega} \rightarrow T$  by recursion.

Let  $m$  be a successor to  $\lambda$  such that  $T(m)$  is infinite.  $T(m)$  must then have no maximal infinite node and no isolated path. Let  $\varphi(\emptyset) = m$ . Now let  $\sigma \in 2^{<\omega}$  and suppose inductively that for all  $\tau, \tau' \subseteq \sigma$ ,  $\tau \subseteq \tau'$  if and only if  $\varphi(\tau) \preceq \varphi(\tau')$ . Let  $\varphi(\sigma) = n$  and assume also for the induction that  $T(n)$  is infinite. Using  $0''$  find two incomparable extensions of  $n$ , say  $n_0$  and  $n_1$  such that  $T(n_0)$  and  $T(n_1)$  are both infinite.  $T$  has no isolated paths or maximal infinite nodes, so these nodes must exist. Set  $\varphi(\sigma * 0) = n_0$  and  $\varphi(\sigma * 1) = n_1$ . It is easy to check that the inductive assumptions hold for  $\sigma * 0$  and  $\sigma * 1$ ,  $T(n_0)$  and  $T(n_1)$ . As  $\lambda$  is not in the range of  $\phi$ , the embedding is nontrivial.  $\square$

In cases 1 and 2 we may not need  $0'$  to compute a self-embedding. An examination of the proofs of these cases will show that the self-embeddings can be computed if one knows the branching function and the successor relation of  $T$ . This is not the situation in case 3 because both the branching function and the successor relation are computable from  $0'$  while there are trees as in case 3 any of whose self-embeddings compute  $0''$ , as Theorem 2.14 makes clear.

We now show that in these cases the results are the best possible. First that in cases 1 and 2 there is a tree satisfying the respective

case all of whose nontrivial self embeddings compute  $0'$ . Finally we show that there is a tree satisfying Case 3 all of whose nontrivial self embeddings compute  $0''$ .

**Theorem 2.6.** *There is a tree  $T$  with a maximal infinite node, such that any weakly nontrivial self embedding computes  $0'$ .*

*Proof.* We describe the tree before explicitly constructing it.  $0$  will be the root of  $T$  and the set of successors of  $0$  will be the set of positive even numbers,  $\mathbb{E}^+$ . Each subtree  $T(n)$  with  $n \in \mathbb{E}^+$  is a tree with no branching nodes. We construct a sequence of positive even numbers  $2 = m_0 < m_1 < m_2 < \dots$  which along with  $T$  will have the following properties:

- I. For all  $i \in \mathbb{N}$  and  $p, q \in \mathbb{E}^+$ , if  $m_i \leq p < q < m_{i+1}$  then  $\text{ht}(T(p)) > \text{ht}(T(q))$ ;
- II. If  $K = \bigcup_{s=0}^{\infty} K_s$  is a computable enumeration of a complete c.e. set, then for all  $i$   $K[i] = K_{m_i}[i]$ .

These two properties are sufficient to guarantee that any nontrivial embedding computes  $K$ . Let  $\varphi$  be any such embedding. As  $\varphi$  is weakly nontrivial, there must be an  $n \in \mathbb{N}$  such that  $\varphi(n) \neq n$ . Fix any such  $n$ . For all  $m > 0$  let  $\lfloor m \rfloor$  be the unique element of  $\mathbb{E}^+$  such that  $m \in T(\lfloor m \rfloor)$ . We define an increasing function  $\psi$ , computable from  $\varphi$  as follows:

$$\psi(0) = \lfloor n \rfloor$$

$$\psi(s+1) = \lfloor \varphi^k(n) \rfloor \text{ where } k = k(s+1) \text{ is the least natural number such that } \lfloor \varphi^k(n) \rfloor \geq \psi(s).$$

We prove by induction that  $\psi(s) \geq m_s$  for all  $s \in \mathbb{N}$ . We will then have by II above that  $K[i] = K_{\psi(i)}[i]$  and hence that  $\psi \geq_T K$ . But  $\varphi \geq_T \psi$  and therefore  $\varphi \geq_T K$ .

The base of the induction is  $\psi(0) = \lfloor n \rfloor \geq 2 = m_0$ . Now suppose that  $\psi(s) \geq m_s$ . Let  $j$  be such that  $m_j \leq \psi(s) < m_{j+1}$ . Thus  $j \geq s$  as  $\langle m_s \rangle$  is an increasing sequence. But if  $j > s$ , then we are done as  $\psi$  is increasing. So we can assume that  $m_s \leq \psi(s) < m_{s+1}$ .

Property I above ensures that for all  $l$  if  $\psi(s) < l < m_{s+1}$ , then  $T(\psi(s)) \not\leq_T T(l)$ . Therefore, for all  $t \geq k = k(s)$ ,

$$\lfloor \varphi^t(n) \rfloor > \psi(s) \implies \lfloor \varphi^t(n) \rfloor \geq m_{s+1}.$$

But  $k(s+1) \geq k(s)$  and so in particular  $\psi(s+1) \geq m_{s+1}$  as required.

It remains to give the construction of such a  $T$  satisfying I and II. We build it in stages. At stage  $s$  we build  $T^s$  and  $T$  will be  $\bigcup_s T^s$ .  $T^0$  will consist of all the even nodes as above as well as an infinite/cofinite computable set of odd numbered nodes arranged so that for all  $e \in \mathbb{E}^+$   $\text{ht}(T(e)) = e$ .

We use a movable marker argument to create the sequence  $\langle m_i \rangle$ . As we do this we also ensure that I. and II. are satisfied. We describe a uniformly computable sequence  $\langle m_{i,s} \rangle$  with the properties

- i.  $\forall i \ m_{i,0} = 2i$ ,
- ii.  $\forall i, s \ m_{i,s} < m_{i+1,s}$ ,
- iii.  $\forall i, s \ m_{i,s} \leq m_{i,s+1}$ ,
- iv.  $\forall i \ \lim_s m_{i,s}$  exists.

For each  $i$   $m_i$  is defined to be  $\lim_s m_{i,s}$ . We enumerate  $K$  one element at a time. Suppose  $s$  is a stage at which  $i \in K_{s+1} \setminus K_s$  and let  $k \geq i$  be the smallest number such that  $m_{k,s} \geq s+1$ . Then we set

$$m_{j,s+1} = \begin{cases} m_{j,s} & \text{if } j < i \\ m_{k+t,s} & \text{if } j = i+t, (t \in \mathbb{N}). \end{cases}$$

At the same time it is necessary to adjust the subtrees  $T^s(e)$  with  $e \in \mathbb{E}^+$ . We leave all successor trees unchanged except perhaps those  $T^s(e)$  with  $m_{i-1,s} \leq e < m_{k,s} = m_{i,s+1}$  (if  $i = 0$  take  $m_{i-1} = 2$ ). To the subtrees  $T(e)$  with  $m_{i-1,s} \leq e < m_{i,s+1}$ , we add the minimum number of nodes to the top of each subtree, retaining the property that there are no branching nodes and ensuring that property I is preserved.

The argument that the tree  $T$  and the sequence  $\langle m_i \rangle$  have the required properties is now just the typical movable marker argument, made explicit in the following lemmas.

**Lemma 2.7.** *For every  $j$   $\lim_s m_{j,s}$  exists.*

*Proof.* Let  $s$  be such that  $K_s[j] = K[j]$ . Then for all  $t \geq s$ , if  $i \in K_{s+1} \setminus K_s$ , then  $i > j$ , and so for all  $t \geq s$   $m_{j,t} = m_{j,t+1}$ .  $\square$

**Lemma 2.8.** *Every successor tree  $T(e)$  with  $e \in \mathbb{E}^+$  is finite.*

*Proof.* Fix  $e \in \mathbb{E}^+$ . Let  $i$  be such that  $m_i \geq e$ .  $T^{s+1}(e) \neq T^s(e)$  only if  $s$  is a stage such that  $m_{i,s} \neq m_{i,s+1}$ . So if  $t$  is a stage such that  $m_{i,t} = m_i$ , which exists by the previous lemma, then  $T^u(e) = T(e)$  for all  $u \geq t$ . As only a finite number of nodes are added at any stage,  $T(e)$  is finite.  $\square$

**Lemma 2.9.** *T has properties I. and II. above.*

*Proof.* The fact that  $T$  has property I follows immediately from the fact that the property is explicitly retained at each stage in the construction and the previous two lemmas.

II. Fix any  $j \in \mathbb{N}$ . If  $s$  is the the largest stage at which  $i \in K_{s+1} \setminus K_s$  for some  $i \leq j$ , then as per the construction:

$$m_j = m_{j,s+1} \geq m_{i,s+1} = m_{k,s} \geq s + 1,$$

where  $k$  is as in the construction above. But  $K_{s+1}[j] = K[j]$  by our choice of  $s$  so  $K_{m_j}[j] = K[j]$ .  $\square$

This establishes the result.  $\square$

Notice that in the the previous construction the successor relation ( $S$ ) is computable but the branching function ( $\text{br}$ ) is not. This latter fact is because the set of leaves of  $T$  (that is, the set  $\{n : \text{br}(n) = 0\}$ ) is not computable. In fact, because the self-embedding can be computed from  $\text{br} \oplus S$ , it must be that  $\text{br} \equiv_T 0'$ .

This is not an essential part of the construction and it is easy to reverse the situation and build a tree isomorphic to the one in the theorem whose branching function is computable but whose successor relation computes  $0'$ . To do this, instead of adding nodes to the top of each subtree, simply add the same number of nodes immediately above the root of each subtree.

**Lemma 2.10.** *If  $T$  is a tree ordering on  $\mathbb{N}$  such that*

- a)  *$T$  is finitely branching*
- b)  $\forall m, n \quad m \succeq n \implies m \geq n$
- c)  $\forall n \quad n \succeq$  *the immediate predecessor of  $n + 1$*

*then  $T$  has exactly one infinite path  $X$ . Furthermore, for all  $n \in \mathbb{N}$   $X(n) = \max\{m : \text{ht}(m) \leq n\}$ .*

*Proof.* Let  $k_n = \max\{m : \text{ht}(m) \leq n\}$  which exists by a). For any fixed  $n$  we will show by induction that  $l \succeq k_n$  for all  $l \geq k_n$ . Thus for any  $n$   $k_n$  is an infinite node and has an extension of height  $n$ . So by b) it is of height  $n$  and the only infinite node of height  $n$ , and  $k_0, k_1, k_2, \dots$  is the only path through  $T$ .

Fix  $n \in \mathbb{N}$ . For the base of the induction, we have trivially that  $k_n \succeq k_n$ . Suppose then that  $l \geq k_n$  and  $l \succeq k_n$ . We show that

$l + 1 \succeq k_n$ . Let  $p$  be the immediate predecessor of  $l + 1$ . By c)  $l \succeq p$  and hence  $p$  is comparable to  $k_n$ . So either  $p \succeq k_n$  in which case  $l + 1 \succeq k_n$  and we are done, or  $p \prec k_n$  in which case  $\text{ht}(l + 1) \leq n$  (as  $\text{ht}(k_n) \leq n$ ) contradicting the fact that  $k_n = \max\{m : \text{ht}(m) \leq n\}$  as  $l + 1 > l \geq k_n$ .  $\square$

**Theorem 2.11.** *There is a computable tree  $T$  with an isolated path and no maximal infinite node such that any nontrivial self-embedding computes  $0'$ .*

*Proof.* We construct  $T$  to have exactly one infinite path  $X$  such that  $\text{deg}_T(X) \geq 0'$ . This will prove sufficient to establish the theorem.

$\langle T, \preceq \rangle$  will be built computably in stages denoted  $\langle T_s, \preceq_s \rangle$  with  $T_s = \{0, 1, 2, \dots, s\}$  and  $\preceq_s = \preceq \cap (T_s \times T_s)$  for all  $s$ .  $T$  will be  $\bigcup_s T_s = \mathbb{N}$  and  $\preceq$  will be  $\bigcup_s \preceq_s$ . At each stage  $s$  we designate an element  $n_s$  of  $T_s$  to be the immediate predecessor of  $s + 1$  and  $\preceq_{s+1}$  is defined to be the transitive closure of  $\preceq_s \cup \{(n_s, s + 1)\}$ . Note that in any tree constructed like this  $\preceq$  will be a subset of  $\leq$  and from this it is straightforward to see that  $T$  will be computable.

Let  $\preceq_0 = \{(0, 0)\}$  and at stage  $s$  let

$$n_s = \begin{cases} \max\{m \preceq s : K_{s+1}[\text{ht}(m)] = K_s[\text{ht}(m)]\} & \text{if this exists,} \\ 0 & \text{otherwise.} \end{cases}$$

**Lemma 2.12.**  *$T$  has exactly one infinite path  $X$  which computes  $K$ .*

*Proof.* Conditions b) and c) of Lemma 2.10 are satisfied immediately by the construction of  $T$ . To see that a) is satisfied we show that  $\forall m \in \mathbb{N} \exists s \in \mathbb{N} \forall p > s \text{ ht}(p) \geq m$ . Fix  $m \in \mathbb{N}$ . Let  $s$  be such that  $K_s[m] = K[m]$ . Let  $t \geq s$ . Then we have  $K_{t+1}[m] = K_t[m]$  and  $\text{ht}(n_t) \geq m$ . But for all such  $t$ ,  $t + 1 \succ n_t$  so  $\text{ht}(t + 1) > \text{ht}(n_t) \geq m$ .

To prove that  $X \geq_T K$  we show that for all  $n$ ,  $K_{X(n)}[n] = K[n]$ . Suppose not. Let  $s \geq X(n)$  be maximum such that  $K_{s+1}[n] \neq K_s[n]$ . Then  $s + 1 \succ n_s$  and  $n_s \prec X(n)$ . But then by Lemma 2.10, the unique path through  $T$  must pass through  $s + 1$  and not  $X(n)$ . Contradiction.  $\square$

**Lemma 2.13.** *Any nontrivial self embedding  $\varphi$  of  $T$  computes  $K$ .*

*Proof.* Let  $\varphi$  be as in the lemma and  $m$  some node on  $X$  such that  $\varphi(m) \neq m$  (that such a node exists is a consequence of there being

no nontrivial self-embeddings of finite trees).  $\varphi(m)$  must also lie on  $X$  as  $T$  has only one infinite path. By induction one sees that for all  $n$   $\varphi^n(m) \succeq X(n)$ . But then  $\varphi^n(m) \geq X(n)$  and so for all  $n$   $K_{\varphi^n(m)}[n] = K_{X(n)}[n] = K[n]$ . Therefore  $\varphi \geq_T K$ .

□

□

The result can be improved slightly by replacing *nontrivial* with *weakly nontrivial* in 2.11. To show this we construct  $T'$  from  $T$ .  $T'$  will have, like  $T$ , a single isolated path  $X'$  that computes  $K$ .  $T'$  will be modified however to ensure that any weakly nontrivial self-embedding must move a node on  $X'$ .

We say a node  $n$  on  $T$  is *just off*  $X$  if  $n$  is not on  $X$  but the immediate predecessor is on  $X$ . Any weakly nontrivial self-embedding of  $T$  that fixes every node on  $X$  must be weakly nontrivial on some finite tree  $T(n)$  with  $n$  just off  $X$ . Any finite tree can be properly extended to a finite tree that has no weakly nontrivial self embedding by adding nodes extending the leaves so that no two leaves have the same height. This is what we do to ensure that  $T(n)$  has no nontrivial self embedding. Extensions may be added at different stages in the construction but we ensure that each leaf is extended only a finite amount.

We first repeat the construction of  $T$  using only the even numbers - adding node  $2(s+1)$  as the immediate successor to  $2n_s$  at stage  $s$ . For all  $n$   $X'(n) = 2X(n)$ . We now describe the placement of every odd number on  $T'$ .

We begin with  $T'_0 = T_0$ . As before at stage  $s$  we determine  $2n_s$  and place  $2(s+1)$  as its immediate successor. At stage  $s$  we also find all leaves extending  $2n_s$  except  $2(s+1)$  (if the leaves are even they will be numerically less than  $2(s+1)$ ), and we keep track of how many odd nodes we have added by any stage so we can determine these leaves computably). We properly extend all such leaves with successive odd numbers so that any two distinct leaves have different heights.  $T'_{s+1}$  is this extension of  $T_{s+1}$ . We need only show now that every leaf on  $T$  is extended only finitely and that all the odd numbers are used.

As per Lemma 2.10  $X'(n)$  will be the numerically greatest even number of height less than or equal to  $n$ .  $X'(n)$  is added to  $T'$  at stage  $s = X'(n)/2$  and for all  $t \geq s$ ,  $2n_t \succeq X'(n)$ . So no more extensions will be added to the leaves above any node just off  $X$  whose height is

less than or equal to  $n$ . So each leaf is extended finitely only a finite number of times.

To see that all the odd numbers are used we merely need to note that there are infinitely many nodes just off  $X'$  (otherwise  $X'$  would be computable) and that we have decreed that each leaf extending such a node must be properly extended.

**Theorem 2.14.** *There is an infinite computable binary branching tree  $S$  such that  $S$  has no isolated paths and such that if  $\delta : S \rightarrow S$  is a nontrivial self-embedding, then  $0'' \leq_T \delta$ .*

Before proving Theorem 2.14, we outline the main steps of the proof. We begin by giving a particular computable approximation to  $0''$  which is conducive to our coding methods. Next, we define a c.e. subtree  $T \subseteq 2^{<\omega}$  such that  $0''$  is coded into the branching levels of  $T$ . ( $T \subseteq 2^{<\omega}$  is a tree in this context if it is closed under initial segments. By a c.e. subtree  $T \subseteq 2^{<\omega}$  we mean that  $T$  is a c.e. set of elements of  $2^{<\omega}$  which forms a tree under the relation  $\subseteq$ .) We say  $n$  is a *branching level* of  $T$  if there is a string  $\sigma \in T$  such that  $|\sigma| = n$  and both  $T(\sigma * 0)$  and  $T(\sigma * 1)$  are infinite. We use this c.e. subtree  $T$  of  $2^{<\omega}$  because it makes the notation easier when proving properties of  $T$  such that where the branching levels occur and the fact that  $T$  has no isolated paths.

We show that from any nontrivial self-embedding of  $T$  we can compute a function dominating the branching levels and that any such function computes  $0''$ . Finally, we show how to define a computable tree  $S \cong T$  for which the successor relation is computable. Because the branching levels of  $T$  are invariant under isomorphisms, we have  $0''$  coded into the branching levels of  $S$ . From any nontrivial self-embedding of  $S$ , we can decode  $0''$  as long as we can determine the height of each node in  $S$ . However, since the successor relation is computable in  $S$ , we can effectively determine the height of any node.

We begin by developing our computable approximation to  $0''$ . Fix a uniformly c.e. sequence of c.e. sets  $A_n$  for  $n \in \mathbb{N}$  such that  $\{n \mid A_n \text{ is finite}\} \equiv_T 0''$ . Without loss of generality, we assume that in the uniform enumeration of the  $A_n$  sequence, exactly one set gets an element at each stage. Let  $f(n) =$  the least  $s$  such that the sets among  $A_0, \dots, A_n$  which are finite have been completely enumerated by stage  $s$ .

**Lemma 2.15.**  $0'' \leq_T f \oplus 0'$ .

*Proof.* To determine whether  $n \in 0''$ , ask  $0'$  whether  $A_n$  gets an element after stage  $f(n)$ . The answer to this question is no if and only if  $n \in 0''$ .  $\square$

We want to define a computable approximation  $f(n, s)$  to the function  $f(n)$  so that  $f(n) = \liminf_s f(n, s)$  and various other properties hold. To define  $f(n, s)$ , proceed as follows. If the sets  $A_0, \dots, A_n$  are all empty at stage  $s$ , then set  $f(n, s) = 0$ . If at least one of these sets is nonempty but none of them receives a new element at stage  $s$ , then let  $f(n, s) = t$  where  $t < s$  is the last stage at which one of these sets received an element.

If we are not in one of these two cases, then at stage  $s$ , exactly one set among  $A_0, \dots, A_n$  gets a new element. Let  $i_{n,s} \leq n$  be such that  $A_{i_{n,s}}$  gets a new element at stage  $s$  and let  $t_{n,s} < s$  be the last stage at which  $A_{i_{n,s}}$  received an element. (If  $s$  is the first stage at which  $A_{i_{n,s}}$  gets an element, then set  $t_{n,s} = 0$ .) Let

$$I_{n,s} = \{j \leq n \mid A_j \text{ has received an element since } t_{n,s}\}.$$

$I_{n,s}$  represents our current guess at which sets among  $A_0, \dots, A_n$  are infinite. Let  $f(n, s) = t$  where  $t < s$  is the greatest stage such that there exists a  $j \leq n$  for which  $j \notin I_{n,s}$  and  $A_j$  gets an element at stage  $t$ . (If the sets  $A_j$  for  $j \notin I_{n,s}$  are all empty or if  $I_{n,s} = \{0, 1, \dots, n\}$ , then set  $f(n, s) = 0$ .) That is, to calculate  $f(n, s)$  we look at the sets  $A_j$  for  $j \leq n$  which we currently think are not infinite and take the last stage at which one of these sets received a new element. The function  $f(n, s)$  is a total computable function.

**Lemma 2.16.** *The function  $f(n, s)$  satisfies the following properties.*

1.  $f(n) = \liminf_s f(n, s)$ .
2. For every  $k > f(n)$ , there is a stage  $s_k$  such that for all  $t \geq s_k$  either  $f(n, t) = f(n)$  or  $f(n, t) > k$ .

*Proof.* Fix  $n$  and break into two cases. If  $A_0, \dots, A_n$  are all finite, then let  $u$  be the last stage at which any of these sets gets an element. Because  $f(n, s) = u$  for all  $s > u$ , we have both Property 1 and 2 in this case.

Otherwise, there is at least one set among  $A_0, \dots, A_n$  which is infinite. Let  $I$  be the set of all  $i \leq n$  such that  $A_i$  is infinite and let  $u_0 = f(n)$  be the last stage such that some  $A_j$  with  $j \leq n$  and  $j \notin I$  receives an element. Let  $u_1 > u_0$  be a stage such that each  $A_i$  with  $i \in I$  has received at least one element between stages  $u_0$  and  $u_1$ .

Consider any stage  $s > u_1$  and split into two cases. First, if none of the sets  $A_i$  for  $i \leq n$  receives an element at stage  $s$ , then  $f(n, s) > u_0$  since  $I \neq \emptyset$  and each  $A_i$  for  $i \in I$  received an element after stage  $u_0$ . Second, if one of the  $A_i$  sets for  $i \in I$  does receive an element at stage  $s$ , then  $t_{n,s} > u_0$  since each such set receives an element between stages  $u_0$  and  $u_1$ . Furthermore,  $I_{n,s} \subseteq I$  since none of the sets  $A_j$  for  $j \notin I$  receives an element after stage  $u_0$ .

If  $I_{n,s} = I$ , then  $f(n, s) = u_0 = f(n)$ . If  $I_{n,s} \subsetneq I$ , then  $f(n, s) > u_0$  since there is an  $A_i$  for which  $i \in I \setminus I_{n,s}$  and this  $A_i$  received an element after stage  $u_0$ . Therefore, for all  $s > u_1$ ,  $f(n, s) \geq u_0 = f(n)$ .

Define a sequence of stages  $v_0 < v_1 < v_2 < \dots$  such that  $u_1 < v_0$  and at each stage  $v_k$ ,  $I_{n,v_k} = I$ . (We can define such a sequence because each  $A_i$  with  $i \in I$  is infinite. Therefore, for any stage  $v_k$ , there is a next greatest stage  $v_{k+1}$  at which some  $A_i$ ,  $i \in I$  receives an element and every other  $A_j$  with  $j \in I$  has received an element since the last time  $A_i$  received an element.) Since  $I_{n,v_k} = I$ , we have  $f(n, v_k) = u_0 = f(0)$ . Therefore, we have established Property 1.

On the other hand, we can extend our sequence of stages  $u_0 < u_1 < u_2 < \dots$  so that each  $A_i$ ,  $i \in I$ , receives an element between stages  $u_k$  and  $u_{k+1}$ . Consider any  $s > u_{k+1}$ . If none of the sets  $A_i$ ,  $i \in I$ , receive an element at  $s$ , then  $f(n, s) > u_k$  since  $I \neq \emptyset$  and each  $A_i$ ,  $i \in I$ , has received an element since  $u_k$ . If some  $A_i$ ,  $i \in I$ , does receive an element at stage  $s$ , then either  $I_{n,s} = I$  (in which case  $f(n, s) = u_0 = f(n)$ ) or  $I_{n,s} \subsetneq I$  (in which case  $f(n, s) > u_k$  since  $A_{i_{n,s}}$  has received an element since stage  $u_k$ ). Therefore, we have established Property 2.  $\square$

We define a computable function  $g(n, s)$  from  $f(n, s)$  that has one further property. Fix  $s$  and we define  $g(n, s)$  by induction on  $n$ . Let  $g(0, s) = f(0, s)$ . Assume  $g(i, t)$  has been defined for all  $i \leq n$  and  $t \leq s$ . Let  $k_{n,s}$  be the number of stages  $t < s$  for which  $g(i, t) = g(i, s)$  for all  $i \leq n$  and let  $m_{n,s} =$  the maximum value of  $g(i, s)$  for  $i \leq n$ . Let  $l_{n,s} = k_{n,s} + m_{n,s}$ . Define  $g(n+1, s) = f(n+1, l_{n,s})$ .

**Lemma 2.17.** *The function  $g(n, s)$  satisfies the following properties.*

1.  $f(n) = \liminf_s g(n, s)$ .
2. For every  $k > f(n)$ , there is a stage  $s_k$  such that for all  $t \geq s_k$  either  $g(n, t) = f(n)$  or  $g(n, t) > k$ .
3. For every  $n$ , there are infinitely many stages  $s$  at which  $g(i, s) = f(i)$  for all  $i \leq n$ .

*Proof.* We proceed by induction on  $n$ . By Lemma 2.16, these properties hold for  $n = 0$ . Assume these properties hold for  $i \leq n$  and we prove them for  $g(n+1, s)$ . Applying Property 3 to  $n$ , let  $u_0 < u_1 < \dots$  list all the stages at which  $g(i, s) = f(i)$  for all  $i \leq n$ . Let  $M =$  the maximum of  $f(i)$  for  $i \leq n$ . At each stage  $u_k$ , we have  $m_{n, u_k} = M$  and  $k_{n, u_k} = k$ , so by definition  $g(n+1, u_k) = f(n+1, M+k)$ . Therefore, as  $k \rightarrow \infty$ ,  $g(n+1, u_k)$  takes on all the values of  $f(n+1, t)$  for  $t > M$ .

Let  $t > M$  be a stage for which  $f(n+1, t) = f(n+1)$ . (By Property 2 of Lemma 2.16 there are infinitely many such stages.) Let  $k = t - M$ . At stage  $u_k$ , we have  $g(i, u_k) = f(i)$  for all  $i \leq n$  by definition of  $u_k$  and we have  $g(n+1, u_k) = f(n+1, M+k) = f(n+1, t) = f(n+1)$ . Therefore, Property 3 of this lemma holds for  $n+1$ .

For any  $a \in \mathbb{N}$ , let  $s_a > u_a$  be a stage such that for every  $s > s_a$  and every  $i \leq n$ , either  $g(i, s) = f(i)$  or  $g(i, s) > a$ . (The existence of  $s_a$  follows from Property 2 of this lemma applied inductively to  $i \leq n$ .) Consider any  $s > s_a$ . By definition,  $l_{n, s} = k_{n, s} + m_{n, s}$ . We claim that  $l_{n, s} \geq a$ . There are two cases to consider. First, suppose  $g(i, s) = f(i)$  for all  $i \leq n$ . In this case,  $m_{n, s} = M$  and because  $s_a > u_a$ , there have been at least  $a$  many stages  $t < s$  for which  $g(i, t) = g(i, s) = f(i)$  for all  $i \leq n$ . Therefore,  $k_{n, s} \geq a$ , so  $l_{n, s} \geq a$ . Second, suppose that for some  $i \leq n$  we have  $g(i, s) \neq f(i)$ . By the choice of  $s_a$ ,  $g(i, s) > a$ , so  $m_{n, s} > a$  and  $l_{n, s} > a$ . Therefore, in either case  $l_{n, s} \geq a$  and so  $g(n+1, s) = f(n+1, t)$  for some  $t \geq a$ .

The previous paragraph established that for all  $a$ , there is a stage  $s_a$  such that for all  $s > s_a$ , there is a  $t \geq a$  for which  $g(n+1, s) = f(n+1, t)$ . Combining this fact with Property 2 of Lemma 2.16 and with the fact that  $g(n+1, u_k) = f(n+1)$  for all  $u_k$  yields Properties 1 and 2 of this lemma.  $\square$

We now put together the last two pieces of our approximating function. Let  $h(n) =$  the least stage  $s$  for which  $K_s \upharpoonright n+1 = K \upharpoonright n+1$ . ( $K = \{e \mid \varphi_e(e) \text{ halts}\}$  denotes the usual halting set for the partial computable functions.) Because  $h(n)$  is a  $\Delta_2^0$  function, it has a computable approximation  $h(n, s)$  such that  $\lim_s h(n, s) = h(n)$ . Finally, let  $a(n, s)$  be the computable function defined by  $a(n, s) = \max\{g(n, s), h(n, s)\}$ .

**Lemma 2.18.** *The computable function  $a(n, s)$  satisfies the following properties.*

1.  $a(n) = \liminf_s a(n, s)$  exists and for all  $n$ ,  $a(n) \geq f(n), h(n)$ .

2. For all  $n$  and for every  $k > a(n)$ , there is a stage  $s_k$  such that for all  $t \geq s_k$ , either  $a(n, t) = a(n)$  or  $a(n, t) > k$ .
3. For every  $n$ , there are infinitely many stages  $s$  at which  $a(i, s) = a(i)$  for all  $i \leq n$ .
4. For any function  $b$  which dominates  $a$ ,  $0'' \leq_T b$ .

*Proof.* Properties 1 through 3 follow from Lemma 2.17 and the fact that  $h(n) = \lim_s h(n, s)$ . Property 4 follows from the fact that if  $b$  dominates  $a$ , then  $b$  dominates both  $h$  and  $f$ . The fact that  $b$  dominates  $h$  gives  $0' \leq_T b$ . Combining this fact with Lemma 2.15 gives  $0'' \leq_T b$ .  $\square$

We next define a c.e. subtree  $T \subseteq 2^{<\omega}$  such that the branching levels of  $T$  dominate the function  $a(n)$ . We begin with some notation. For any  $s$ , any  $n \leq s$  and any string  $\sigma \in 2^{<\omega}$  with  $|\sigma| = n + 1$ , we define

$$\tau_{n,s}^\sigma = 0^{a(0,s)} * \sigma(0) * 0^{a(1,s)} * \sigma(1) * \dots * 0^{a(n,s)} * \sigma(n).$$

For any  $n$  and any string  $\sigma \in 2^{<\omega}$  such that  $|\sigma| = n + 1$ , we let

$$\tau_n^\sigma = 0^{a(0)} * \sigma(0) * 0^{a(1)} * \sigma(1) * \dots * 0^{a(n)} * \sigma(n).$$

For any nonempty string  $\alpha \in 2^{<\omega}$ , let  $\alpha'$  denote the string obtained by removing the last element of  $\alpha$ . Notice that

$$(\tau_{n,s}^\sigma)' = 0^{a(0,s)} * \sigma(0) * 0^{a(1,s)} * \sigma(1) * \dots * 0^{a(n,s)}$$

is a string of length  $n + \sum_{i=0}^n a(i, s)$  and that  $(\tau_n^\sigma)'$  is a string of length  $n + \sum_{i=0}^n a(i)$ .

The subtree  $T \subseteq 2^{<\omega}$  is enumerated in stages as a sequence of finite trees  $T_0 \subseteq T_1 \subseteq T_2 \subseteq \dots$ . Set  $T_0 = \emptyset$ . To define  $T_{s+1}$ , consider each string  $\sigma \in 2^{<\omega}$  which has length  $s + 1$ . Let  $\alpha_\sigma$  be the lexicographically least element of  $2^{<\omega}$  which extends  $\tau_{s,s}^\sigma$  and which is not in  $T_s$ . Add  $\alpha_\sigma$  and all of its initial segments to  $T_s$ .  $T_{s+1}$  is the tree formed by adding these strings when  $\sigma$  ranges over all elements of  $2^{<\omega}$  of length  $s + 1$ . Our desired tree is  $T = \bigcup_s T_s$ .

**Lemma 2.19.** *For each  $n$  and each  $\sigma \in 2^{<\omega}$  of length  $n + 1$ , the node  $(\tau_n^\sigma)'$  is a branching node of  $T$ .*

*Proof.* Let  $u_0 < u_1 < \dots$  be the stages such that  $a(i, u_k) = a(i)$  for all  $i \leq n$ . For each such stage,  $(\tau_n^\sigma)' = (\tau_{n, u_k}^\sigma)'$ , so both  $T_{u_k}((\tau_n^\sigma)' * 0)$  and  $T_{u_k}((\tau_n^\sigma)' * 1)$  gain extra elements at stage  $u_k + 1$ . Therefore, these trees are infinite and  $(\tau_n^\sigma)'$  is a branching node in  $T$ .  $\square$

**Lemma 2.20.** *If  $\xi$  is a branching node of  $T$ , then there is an  $n$  such that  $|\xi| = n + \sum_{i=0}^n a(i)$ .*

*Proof.* Suppose  $\xi \in T$  is such that there is an  $n$  such that

$$n + \sum_{i=0}^n a(i) < |\xi| < n + 1 + \sum_{i=0}^{n+1} a(i). \quad (1)$$

By Lemma 2.18, let  $u$  be a stage such that for all  $s > u$  and all  $i \leq n + 1$ , either  $a(i, s) = a(i)$  or  $a(i, s) > |\xi|$ . Fix any stage  $s > u$ .

We claim that there is a  $j \leq n$  such that

$$j + \sum_{i=0}^j a(i, s) < |\xi| < j + 1 + \sum_{i=0}^{j+1} a(i, s). \quad (2)$$

The proof of the claim breaks into two cases. If  $a(i, s) = a(i)$  for all  $i \leq n$ , then the claim with  $j = n$  follows from Equation 1. Otherwise, let  $j < n$  be the least number such that  $a(j + 1, s) \neq a(j + 1)$ . In this case,

$$j + \sum_{i=0}^j a(i, s) = j + \sum_{i=0}^j a(i) < n + \sum_{i=0}^n a(i) < |\xi|.$$

Because  $a(j + 1, s) > |\xi|$ , we have  $j + 1 + \sum_{i=0}^{j+1} a(i, s) > |\xi|$  and hence Equation 2 holds in this case as well.

By Equation 2, at stage  $s$  there is a unique  $\sigma \in 2^{<\omega}$  with length  $j + 1$  such that  $\tau_{j, s}^\sigma \subseteq \xi$ . Furthermore, for  $a \in \{0, 1\}$  we have  $\xi \subsetneq (\tau_{j+1, s}^{\sigma * a})'$ . That is,

$$0^{a(0, s)} * \sigma(0) * \dots * 0^{a(j, s)} * \sigma(j) \subset \xi \subsetneq 0^{a(0, s)} * \sigma(0) * \dots * 0^{a(j, s)} * \sigma(j) * 0^{a(j+1, s)}.$$

It follows that at stage  $s + 1$ ,  $T_s(\xi * 0)$  gets a new element but  $T_s(\xi * 1)$  does not. Because this property holds for any  $s > u$ ,  $T(\xi * 1) = T_u(\xi * 1)$  is finite, so  $\xi$  is not a branching node of  $T$ .

To finish the proof, we need to show that if  $\xi \in T$  and  $|\xi| < a(0)$ , then  $\xi$  is not a branching node. The proof of this fact is similar (but simpler) than the argument above and we leave it to the reader to verify.  $\square$

From Lemmas 2.19 and 2.20, we obtain the following fact.

**Lemma 2.21.** *The  $n^{\text{th}}$  branching level of  $T$  is given by the formula  $b(n) = n + \sum_{i=0}^n a(i)$ .*

**Lemma 2.22.** *If  $T(\xi)$  is infinite, then  $\xi \subseteq \tau_n^\sigma$  for some  $n$  and  $\sigma$  with  $|\sigma| = n + 1$ .*

*Proof.* Suppose that  $\xi \not\subseteq \tau_n^\sigma$  for any  $n$  and  $\sigma$ . We show that  $T(\xi)$  is finite. First, notice that  $\xi$  must contain at least one value of 1 or else  $\xi \subseteq \tau_n^\sigma$  for sufficiently large  $n$  by choosing  $\sigma$  to contain all zeros.

Second, notice that if  $\xi$  does not have  $0^{a(0)}$  as an initial segment, then this property follows trivially. That is, fix a stage  $u$  such that for all  $s > u$ ,  $a(0, s) \geq a(0)$ . At any stage  $s > u$ , we add nodes only above strings  $\tau_{s,s}^\sigma$  and each string  $\tau_{s,s}^\sigma$  begin with  $0^{a(0,s)}$ . Because this string is not an initial segment of  $\xi$ ,  $T_s(\xi)$  does not get a new element at stage  $s + 1$ . Therefore,  $T_u(\xi) = T(\xi)$  and hence  $T(\xi)$  is finite.

It remains to consider the case when  $0^{a(0)}$  is an initial segment of  $\xi$  and  $\xi$  contains at least one value of 1. Let  $j$  be the largest value such that there is are strings  $\alpha$  (with  $|\alpha| = n + 1$ ) and  $\mu$  such that

$$\xi = 0^{a(0)} * \alpha(0) * \dots * 0^{a(j)} * \alpha(j) * \mu.$$

Fix such  $j$ ,  $\alpha$  and  $\mu$ . Because  $\xi \not\subseteq \tau_n^\sigma$  for any  $n$  and  $\sigma$ , the string  $\mu$  must contain at least one value of 1. Write  $\mu = \mu_0 * 1 * \mu_1$  where  $\mu_0$  is such that  $\mu_0(k) = 0$  for all  $k < |\mu_0|$ . Because  $j$  is chosen maximal,  $|\mu_0| < a(j + 1)$ .

Let  $u$  be a stage such that for all  $s > u$  and for all  $i \leq j + 1$ ,  $a(i, s) = a(i)$  or  $a(i, s) > |\xi|$ . The lemma follows from the claim that  $T_u(\xi) = T(\xi)$ . To prove this claim, fix any  $s > u$  and we show that  $T_s(\xi)$  does not gain a new element at stage  $s + 1$ . We split into two cases. First, suppose that for all  $i \leq j$ ,  $a(i, s) = a(i)$ . The only way for  $T_s(\xi)$  to gain a new element at stage  $s + 1$  is if there is a string  $\sigma$  of length  $s + 1$  such that  $\xi \subseteq \tau_{s,s}^\sigma$ . Because  $a(i, s) = a(i)$  for  $i \leq j$ , this string  $\sigma$  must satisfy  $\sigma(i) = \alpha(i)$  for all  $i \leq j$ . It follows that  $0^{a(0)} * \alpha(0) * \dots * 0^{a(j)} * \alpha(j) * 0^{a(j+1,s)}$  is an initial segment of  $\tau_{s,s}^\sigma$ . However, regardless of whether  $a(j + 1, s) = a(j)$  or not, we have  $|\mu_0| < a(j + 1, s)$ . Hence the string  $0^{a(0)} * \alpha(0) * \dots * 0^{a(j)} * \alpha(j) * 0^{a(j+1,s)}$  is not an initial segment of  $\xi$ . Therefore,  $T_s(\xi)$  does not get a new element in this case.

The other case is when there is an  $i < j$  for which  $a(i, s) \neq a(i)$ . Let  $k$  denote the least such  $i$ . The argument is similar.  $T_s(\xi)$  can

gain a new element only if there is a  $\sigma$  such that  $\xi \subseteq \tau_{s,s}^\sigma$ . Because  $a(i, s) = a(i)$  for all  $i < k$ , we have  $\sigma(i) = \alpha(i)$  for  $i < k$  and hence  $0^{a(0)} * \alpha(0) * \dots * 0^{a(k-1)} * \alpha(k-1) * 0^{a(k,s)}$  is an initial segment of  $\tau_{s,s}^\sigma$ . Because  $a(k, s) > |\xi|$ , this string is not an initial segment of  $\xi$  and hence  $T_s(\xi)$  does not get a new element in this case.  $\square$

**Lemma 2.23.** *The tree  $T$  has no isolated paths.*

*Proof.* This lemma follows immediately from Lemmas 2.22 and 2.19.  $\square$

**Lemma 2.24.** *If  $\delta : T \rightarrow T$  is a nontrivial self-embedding, then there is a string  $\xi$  such that  $|\delta(\xi)| > |\xi|$ .*

*Proof.* Suppose there is no such string  $\xi$ . Because  $|\xi| \leq |\delta(\xi)|$  for any self-embedding  $\delta$ , it follows that for each  $n$ ,  $\delta$  restricted to the strings of length  $n$  in  $T$  is a permutation. Therefore  $\delta$  is onto and hence is not nontrivial.  $\square$

**Lemma 2.25.** *If  $\delta : T \rightarrow T$  is a nontrivial self-embedding then there is a  $k \in \mathbb{N}$  and a node  $\xi$  such that  $\xi \subsetneq \delta^k(\xi)$ .*

*Proof.* By Lemma 2.24, let  $\mu_0$  be a node such that  $|\mu_0| < |\delta(\mu_0)|$ . If  $\mu_0 \subseteq \delta(\mu_0)$  then  $\mu_0 \subsetneq \delta(\mu_0)$  and we can let  $\xi = \mu_0$  and  $k = 1$  to verify the lemma. Otherwise, assume that  $\mu_0 \not\subseteq \delta(\mu_0)$ . Let  $\mu_1$  be such that  $|\mu_1| = |\mu_0|$  and  $\mu_1 \subseteq \delta(\mu_0)$ . Notice that  $\mu_1 \neq \mu_0$  and  $\delta(\mu_0) \neq \mu_0$ .

We proceed by induction. Assume that  $n \geq 1$  and we have defined a sequence of pairwise distinct nodes  $\mu_0, \mu_1, \dots, \mu_n$  such that  $|\mu_i| = |\mu_0|$  for all  $i \leq n$  and  $\mu_{i+1} \subseteq \delta(\mu_i)$  and  $\mu_i \neq \delta(\mu_i)$  for all  $i < n$ . (The last two sentences of the previous paragraph establishes the required properties when  $n = 1$ .)

We claim that in this situation,  $\mu_n \neq \delta(\mu_n)$ . Suppose that  $\mu_n = \delta(\mu_n)$ . Because  $|\mu_n| = |\mu_{n-1}|$  and  $\mu_n \neq \mu_{n-1}$ ,  $\mu_n$  and  $\mu_{n-1}$  are incomparable nodes. However,  $\delta(\mu_n) = \mu_n \subseteq \delta(\mu_{n-1})$ . Therefore  $\delta(\mu_n)$  and  $\delta(\mu_{n-1})$  are comparable contradicting the fact that  $\delta$  is a self-embedding.

Next, we let  $\mu_{n+1}$  be such that  $|\mu_{n+1}| = |\mu_0|$  and  $\mu_{n+1} \subseteq \delta(\mu_n)$ . We claim that if  $\mu_{n+1} = \mu_i$  for some  $i \leq n$ , then the conclusion of the lemma is true. Otherwise we add  $\mu_{n+1}$  to the list above and continue by induction. Because there are only finitely many nodes at level  $|\mu_0|$ , we must eventually find an  $n$  such that  $\mu_{n+1} = \mu_i$  for some  $i \leq n$ . Hence, the lemma follows from the claim in this paragraph.

Suppose that  $\mu_{n+1} = \mu_i$  for some  $i \leq n$  and let  $l \geq 1$  be such that  $i = (n + 1) - l$ . In this situation we have  $\mu_i = \mu_{n+1} \subseteq \delta^l(\mu_i)$ . We claim that  $\mu_i \neq \delta^l(\mu_i)$  (and hence we have established the lemma with  $\xi = \mu_i$  and  $k = l$ ). We break into three cases.

First, if  $i = 0$ , then we have  $\mu_0 \subseteq \delta^l(\mu_0)$ . But,  $|\mu_0| < |\delta(\mu_0)|$  implies  $|\mu_0| < |\delta^l(\mu_0)|$  so we have  $\mu_0 \subsetneq \delta^l(\mu_0)$  as required.

Second, if  $l = 1$ , we have  $\mu_i \subseteq \delta(\mu_i)$ . Because  $i \leq n$ , we know  $\mu_i \neq \delta(\mu_i)$ , so  $\mu_i \subsetneq \delta(\mu_i)$  as required.

Third, we consider the case when  $i > 0$  and  $l > 1$ . For a contradiction, assume that  $\mu_i = \delta^l(\mu_i)$ . We have  $\mu_i = \mu_{n+1} \subseteq \delta(\mu_n)$  and  $\mu_i \subseteq \delta(\mu_{i-1})$ . By our induction hypothesis,  $\mu_n$  and  $\mu_{i-1}$  are incomparable nodes. Furthermore, we have

$$\mu_i = \mu_{n+1} \subseteq \delta(\mu_n) \subseteq \delta^2(\mu_{n-1}) \subseteq \cdots \subseteq \delta^l(\mu_i) = \mu_i.$$

Therefore,  $\delta(\mu_n) = \mu_i$  so  $\delta(\mu_n)$  and  $\delta(\mu_{i-1})$  are comparable nodes, violating the fact that  $\delta$  is a self-embedding.  $\square$

For any nontrivial self-embedding  $\delta : T \rightarrow T$ , we can fix  $k$  and  $\xi$  as in Lemma 2.25 and let  $\gamma = \delta^k : T \rightarrow T$ .  $\gamma$  is a nontrivial self-embedding of  $T$  such that  $\xi \subsetneq \gamma(\xi) \subsetneq \gamma^2(\xi) \subsetneq \cdots$ . It will also be useful to consider the nontrivial self-embedding  $\iota : T \rightarrow T$  given by  $\iota = \gamma^2$ . Notice that both  $\iota$  and  $\gamma$  are obtained from  $\delta$  by finitely many parameters.

**Lemma 2.26.** *Let  $\delta$  be any nontrivial self-embedding  $\delta : T \rightarrow T$  and let  $\gamma$  and  $\iota$  be defined from  $\delta$  as above. There are nodes  $\alpha$  and  $\beta_0$  such that  $\alpha \subsetneq \beta_0 \subsetneq \iota(\alpha)$  and  $\beta_0$  is a branching node.*

*Proof.* Fix  $\xi$  as in the paragraph before this lemma. Because  $\xi, \gamma(\xi), \gamma^2(\xi), \dots$  traces out a path in  $T$  and because  $T$  has no isolated paths, there must be a  $k \geq 1$  and a branching node  $\beta_0$  such that  $\gamma^k(\xi) \subseteq \beta_0 \subsetneq \gamma^{k+1}(\xi)$ . Let  $\alpha = \gamma^{k-1}(\xi)$ . Because  $\alpha = \gamma^{k-1}(\xi) \subsetneq \gamma^k(\xi) \subseteq \beta_0$ , we have  $\alpha \subsetneq \beta_0$ . Because  $\iota = \gamma^2$ , we have  $\beta_0 \subsetneq \gamma^{k+1}(\xi) = \gamma^2(\gamma^{k-1}(\xi)) = \iota(\alpha)$ .  $\square$

**Lemma 2.27.** *Let  $\iota : T \rightarrow T$  be a nontrivial self-embedding such that there are nodes  $\alpha$  and  $\beta_0$  such that  $\alpha \subsetneq \beta_0 \subsetneq \iota(\alpha)$  and  $\beta_0$  is a branching node. Then there is a branching node  $\beta_1$  such that  $\iota(\alpha) \subsetneq \beta_1 \subsetneq \iota^2(\alpha)$ .*

*Proof.* Fix  $\alpha$  and  $\beta_0$ . Because  $\alpha \subsetneq \beta_0 \subsetneq \iota(\alpha)$ , we have

$$\begin{aligned} \alpha \subsetneq \beta_0 \subsetneq \iota(\alpha) \subsetneq \iota(\beta_0) \subsetneq \iota(\beta_0 * 0) \\ \alpha \subsetneq \beta_0 \subsetneq \iota(\alpha) \subsetneq \iota(\beta_0) \subsetneq \iota(\beta_0 * 1). \end{aligned}$$

Let  $\beta_1$  be the infimum of  $\iota(\beta_0 * 0)$  and  $\iota(\beta_0 * 1)$ . Because these two nodes are incomparable,  $\beta_1$  is strictly contained in both of them. From the offset containments above, it is clear than  $\beta_0 \subsetneq \iota(\alpha) \subsetneq \beta_1$ . Let  $i_0 \in \{0, 1\}$  be such that  $\beta_0 * i_0 \subseteq \iota(\alpha)$ . Because  $\iota(\beta_0 * i_0) \subseteq \iota^2(\alpha)$  and  $\beta_1 \subsetneq \iota(\beta_0 * i_0)$  we have  $\beta_1 \subsetneq \iota^2(\alpha)$ .  $\square$

**Lemma 2.28.** *Let  $\delta : T \rightarrow T$  be any nontrivial self-embedding. There is a nontrivial self-embedding  $\iota : T \rightarrow T$  defined from finitely many parameters in  $T$  and a node  $\alpha$  such that the sequence  $c(n) = |\iota^{n+1}(\alpha)|$  dominates the branching level function  $b(n)$  of  $T$  (see Lemma 2.21). (We define  $\iota^0(\alpha) = \alpha$ .)*

*Proof.* Define  $\iota$  from  $\delta$  as above and let  $\alpha$  and  $\beta_0$  be as in Lemma 2.27. Applying Lemma 2.27 inductively, we obtain a sequence of branching node  $\beta_0 \subsetneq \beta_1 \subsetneq \beta_2 \subsetneq \dots$  such that  $\iota^n(\alpha) \subsetneq \beta_n \subsetneq \iota^{n+1}(\alpha)$ . Therefore, there are at least  $n$  many branching levels below  $|\iota^{n+1}(\alpha)|$ .  $\square$

To prove Theorem 2.14, we need to transform the c.e. subtree  $T \subseteq 2^{<\omega}$  into a computable tree  $S$ . This transformation is easily done in a general setting.

**Lemma 2.29.** *For any c.e. subtree  $\hat{T} \subseteq 2^{<\omega}$ , there is a computable tree  $\hat{S}$  such that  $\hat{S} \cong \hat{T}$ . Furthermore, we can assume that the successor relation is computable in  $\hat{S}$ .*

*Proof.* If  $\hat{T}$  is finite, this lemma follows trivially. Assume  $\hat{T}$  is infinite and  $\hat{T}$  is the range of the total computable 1-1 function  $\varphi_e$ . Let  $\hat{S}$  have domain  $\mathbb{N}$  and let  $\leq_{\hat{S}}$  be defined by  $n \leq_{\hat{S}} m \Leftrightarrow \varphi_e(n) \subseteq \varphi_e(m)$ . Then  $\varphi_e$  is an isomorphism from  $(\hat{S}, \leq_{\hat{S}})$  to  $(\hat{T}, \subseteq)$  as required. Furthermore,  $m$  is a successor of  $n$  in  $\hat{S}$  if and only if  $\varphi_e(m)' = \varphi_e(n)$ .  $\square$

We now present the proof of Theorem 2.14. Let  $T$  be the c.e. subtree of  $2^{<\omega}$  we have constructed and let  $S \cong T$  be the computable tree with a computable successor relation given by Lemma 2.29. Let  $b$  denote the branching level function for  $S$  (which is the same as the branching level function for  $T$  since  $S \cong T$ ). By Lemma 2.21,  $b$  dominates  $a$  and hence by Lemma 2.18,  $0'' \leq_T b$ . Fix any nontrivial self-embedding  $\delta : S \rightarrow S$ . By Lemma 2.28, there is a nontrivial self-embedding  $\iota : S \rightarrow S$  (defined from finitely many parameters in  $S$ ) and a node  $\alpha$  such that the function  $c(n) = |\iota^{n+1}(\alpha)|$  dominates  $b$  (and hence by Lemma 2.18,  $0'' \leq_T c$ ). Because we only need finitely many parameters to obtain  $\iota$  from  $\delta$ , we have  $\iota \leq_T \delta$ . Furthermore,

because the successor function is computable in  $S$ , we can determine the height of any node in  $S$ . Therefore,  $0'' \leq_T c \leq_T \iota \leq_T \delta$  as required.  $\square$

## 2.1 Coding into isomorphism types

We now turn our attention to a related problem in computable tree theory. Consider the following: because the property of  $T$  used to encode  $0''$  in the proof of Theorem 2.14 (namely the branching levels) is invariant under isomorphism, we get the following result.

**Theorem 2.30.** *There is a computable tree  $S$  such that for any  $\hat{S} \cong S$  and any nontrivial self-embedding  $\delta : \hat{S} \rightarrow \hat{S}$ ,  $0'' \leq_T 0' \oplus \delta$ . In particular,  $\hat{S}$  does not have any  $\Delta_2^0$  nontrivial self-embeddings and  $\hat{S}$  does not have any nontrivial self-embeddings which are strictly between  $0'$  and  $0''$ .*

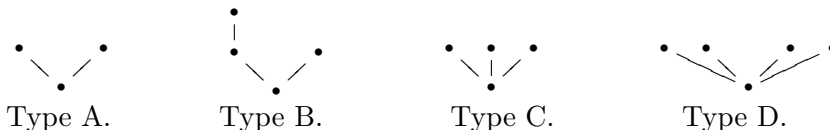
*Proof.* Fix a computable tree  $S \cong T$  where  $T$  is the c.e. subtree  $T \subseteq 2^{<\omega}$  constructed above. Fix any  $\hat{S} \cong S$  and any nontrivial self-embedding  $\delta$  of  $\hat{S}$ . Let  $\iota$  and  $c$  be the functions given by Lemma 2.28 for  $\hat{S}$  and  $\delta$ . The only change from the proof of Theorem 2.14 is that we do not know that the successor relation in  $\hat{S}$  is computable. However, since  $0' \oplus \delta$  can compute both  $\iota$  and the successor relation in  $\hat{S}$ , we have  $0'' \leq_T c \leq_T 0' \oplus \delta$ .  $\square$

We can also ask these types of questions for the previous results. For example, as we saw from Theorem 2.6 there is a tree  $T$  with a maximal infinite node with the property that  $T$  has no computable self-embedding. However it is not clear from the theorem whether this was a property of the specific tree ordering of  $\mathbb{N}$ , or of the isomorphism type of  $T$ . That is, the question arises whether there is another computable tree  $T'$ , classically isomorphic to  $T$ , such that  $T'$  has a computable nontrivial self-embedding. We answer this now.

**Theorem 2.31.** *There is a computable tree with a maximal infinite node such that no classically isomorphic computable tree has a computable nontrivial self-embedding.*

*Proof.* If  $T$  is a tree, a *component* of  $T$  is a tree of the form  $T(n)$  where  $n$  is an immediate successor of the root of  $T$ . We construct a tree  $T$  that will have a single  $\omega$ -node at its root. We describe the tree by constructing the infinite sequence  $\langle T_i \rangle$  of components of  $T$ .  $T_i^t$  will

be the  $i^{\text{th}}$  component enumerated by stage  $t$ . We will suppress the superscript  $t$  when it is implied. Each  $T_i$  will be of height 2 or 3 and will have finitely many components, each of which will be one of the following four types:



Let  $\langle \phi_e \rangle_e$  and  $\langle f_e \rangle_e$  be two effective enumerations of the partial computable functions. We satisfy all requirements of the form:

$R_{\langle e, i \rangle} \equiv \phi_e$  does not enumerate a computable tree isomorphic to  $T$   
or  $f_i$  is not a non-trivial computable self-embedding of  $\phi_e$ .

By Posner's Lemma [], it suffices to satisfy all requirements of the form  $R_e = R_{\langle e, e \rangle}$ . We ensure while we are satisfying these requirements that for all  $m < n$

$$||T_m|| < ||T_n||, \tag{3}$$

$$T_m \text{ has strictly fewer type D components than } T_n. \tag{4}$$

To satisfy the requirements we use an infinite injury argument. We follow the exposition in [?] Chapter XIV. The basic module below will be used to satisfy a general requirement  $R_e$ . There are countably many copies of the basic module indexed by the set  $\{0, 1, 2, 3\}^{<\omega}$  which is ordered lexicographically (*to the left of* means lexicographically less than). A module with index  $\alpha$  will be working to satisfy  $R_{|\alpha|}$ . We say a basic module is *active at stage  $s$*  if it is attempting to satisfy its requirement. Parameters and internal states of modules (whether active or not) will be constant unless explicitly changed during the construction. For all  $n \leq s$  there is exactly one active module  $\alpha$  with  $|\alpha| = n$ . No modules with  $|\alpha| > s$  will be active at stage  $s$ , and if module  $\alpha$  is active and  $\beta \subseteq \alpha$ , then module  $\beta$  is also active. We determine below exactly which modules are active at stage  $s$ .

*Basic Module:*

To ensure that  $T$  is infinite and of the required form (that is, its components all have components of type A, B, C or D satisfying (3) and (4)), during the construction we keep extending  $T$  as follows:

I. At each stage we add one component to  $T$  all of whose components are of type D. Furthermore we ensure that this new component has more type D components than any other existing component. Components are indexed in the order they are added so that  $T$  satisfies (3) and (4);

II. At stages in the construction when we extend  $T$ , we immediately add the minimum number of new components of type D to the components of  $T$  to ensure that (3) and (4) continue to hold.

We enumerate the tree  $\phi_e$  in stages and require that at all stages it can be extended to a tree of the required form. If at any stage this is not true,  $R_e$  is automatically satisfied. Similarly, if  $f_e$  ever ceases to be extendible to an order-preserving self-embedding of  $\phi_e$ , then  $R_e$  is immediately satisfied.

*Step 0.* A parameter  $\alpha \in \{0, 1, 2, 3\}^\omega$ , the index of the module, is used in the module.  $s$  is the stage of the overall construction.  $k = k(\alpha, s)$  is input to the module and its value is computed by the overall construction.

Values  $i = i(\alpha, s)$ ,  $j = j(\alpha, s)$  and  $l = l(\alpha, s)$  are calculated during the operation of the module.  $i$  and  $j$  are initially set to be -1 and  $l$  to be  $\infty$ . These values may change during the operation of the module.

*Step 1.* As the tree  $\phi_e$  (where  $e = |\alpha|$ ) is being enumerated, we search for a component  $S_\alpha$  of  $\phi_e$  such that

1.  $\|S_\alpha^s\| > k$ ,  
(where as before,  $S_\alpha^s$  is the enumeration of  $S_\alpha$  by stage  $s$ ),
2.  $\|S_\alpha^s\| < l(\alpha, s)$  where  $l(\alpha, s)$  is defined recursively by:  $l(\emptyset, s) = \infty$ , and if  $\alpha \neq \emptyset$ ,

$$l(\alpha, s) = \min_{\beta \subsetneq \alpha} (\{\|T_{i(\beta, s)}^s\| : \beta * \langle 0 \rangle \subseteq \alpha\} \cup \{\|T_{j(\beta, s)}^s\| : \beta * 1 \subseteq \alpha\}).$$

If  $i(\beta, s)$  (or  $j(\beta, s)$ ) is still in its initial value of -1, then we take  $\|T_{i(\beta, s)}^s\|$  ( $\|T_{j(\beta, s)}^s\|$ ) to be  $\infty$ . We also adopt the convention that the minimum of the empty set is  $\infty$ .

The definition of  $l(\alpha, t)$  is designed to help manage the conflicts between requirements. The idea (which will be made exact later) behind the definition of  $l(\alpha, s)$  is if  $\alpha$  is working under the assumption

that  $\lim_s \|T_{i(\beta,s)}^s\| = \infty$  (or  $\lim_s \|T_{j(\beta,s)}^s\| = \infty$ ), then it waits for a stage that  $\|S_\alpha^s\| < \|T_{i(\beta,s)}^s\|$  ( $\|T_{j(\beta,s)}^s\|$ ).

3.  $f_e$  is defined on all of  $S_\alpha^s$  (and respects the tree ordering on  $\phi_e$ ), and  $f_e[S_\alpha^s] \cap S_\alpha^s = \emptyset$ .

Each  $T_n$  has at least two nodes of height 2 and we wait for a stage when this is also true of  $S_\alpha$ . We can then identify the root node of  $S_\alpha$  and of  $\phi_e$ . Let  $V_\alpha$  be the component of  $\phi_e$  that contains  $f_e[S_\alpha]$ . If at any stage  $t$  in the computation  $f_e$  ceases to be an order-preserving embedding from  $S_\alpha^t$  to  $V_\alpha^t$ , the module goes into a waiting state until this is (possibly) remedied by the growth of  $V_\alpha$ . If it is never remedied, the requirement will be satisfied.

*Step 2.* We wait for a stage  $s$  in the enumeration of  $T$  such that  $T$  has a component  $T_i^s$  such that  $S_\alpha^s \simeq T_i^s$ . We set  $i(\alpha, s) = i$ . If no such stage exists,  $R_e$  is automatically satisfied.

If at some later stage  $t$   $S_\alpha^t$  ceases to be isomorphic to  $T_{i(\alpha,t)}^t$  (because one or both of them has grown), then we return to step 2 to find a new  $T_i$  for  $S_\alpha$ .

*Step 3.* If  $T_i^s$  does not have exactly one type A component, extend  $T_i^s$  so that it does and return to step 2. Suppose now that  $T_i^s$  does have exactly one type A component. This type A component is the *designated component* of  $T_i^s$ . The root of the designated component is the *designated node* of  $T_i^s$ . The corresponding node and component of  $S_\alpha^s$  are its designated node and component. If  $\nu$  is the designated node of  $S_\alpha^s$ , then  $f_e(\nu)$  and  $V_\alpha^s(f_e(\nu))$  are the designated node and component of  $V_\alpha^s$ .

*Step 4.* Wait until there is a stage  $t$  and a component  $T_j^t$  such that  $V_\alpha^t \simeq T_j^t$  and  $\|T_j^t\| + 3 < l(\alpha, t)$ ,

We then set  $j(\alpha, t) = j$ . Notice that I and II above ensure that  $i < j$  for all non-initial values of  $i$  and  $j$ . If at a later stage  $t'$   $V_\alpha^{t'}$  ceases to be isomorphic to  $T_{j(\alpha,t')}^{t'}$  (while  $S_\alpha^{t'}$  remains isomorphic to  $T_i^{t'}$ ), then we return to Step 4 to search for a new  $T_j$ .

*Step 5.* If  $T_j^t$  has no type A components, go to Step 6. Otherwise minimally extend  $T_j^t$  so that it has no type A components, and go to step 4.

*Step 6.* (The diagonalisation step.) The designated node of  $V_\alpha$  will be a root node of a type B or C component. If type B (type C) enlarge  $T_i$  so that its designated component is type C (type B). As described before, if now  $S_\alpha \not\simeq T_i$ , return to step 2.  $\square$

We use a tree of strategies to coordinate the execution of the basic modules and to ensure that all requirements are satisfied. We define  $\delta_s$  that will control which modules are active. For all  $s$   $|\delta_s| = s$  and is the longest active node.  $\alpha$  is active at stage  $s$  if and only if  $\alpha \subseteq \delta_s$ . Define

$$\delta_0 = \lambda.$$

If  $n \leq |\delta_s|$ , and  $\alpha \subseteq \delta_s$  is of length  $n$ , then

$$\delta_{s+1}(n) = \begin{cases} 0 & \text{if } T_{i(\alpha,s)}^s \neq T_{i(\alpha,s+1)}^{s+1} \text{ and } i(\alpha,s) \neq -1 \\ 1 & \text{if } T_{i(\alpha,s)}^s = T_{i(\alpha,s+1)}^{s+1}, T_{j(\alpha,s)}^s \neq T_{j(\alpha,s+1)}^{s+1} \text{ and } i(\alpha,s) \neq -1 \\ 2 & \text{if } T_{i(\alpha,s)}^s = T_{i(\alpha,s+1)}^{s+1}, T_{j(\alpha,s)}^s = T_{j(\alpha,s+1)}^{s+1} \text{ and } i(\alpha,s) \neq -1 \\ 3 & \text{if } i(\alpha,s) = -1. \end{cases}$$

The *true path*,  $f \in \{0, 1, 2, 3\}^\omega$ , is defined by  $f(n) = \liminf_{t > n} \delta_t(n)$ .

If  $\gamma' \supseteq \gamma * 0$ , then  $\gamma'$  is working under the assumption that  $\lim_t \|T_{i(\gamma,t)}^t\| = \infty$  and  $\lim_t \|T_{j(\gamma,t)}^t\| = \infty$ . If  $\gamma' \supseteq \gamma * 1$ , then module  $\gamma'$  is working under the assumption that  $\lim_t \|T_{i(\gamma,t)}^t\| < \infty$  and  $\lim_t \|T_{j(\gamma,t)}^t\| = \infty$ . If  $\gamma' \supseteq \gamma * 2$ , then  $\gamma'$  is working under the assumption that  $\lim_t \|T_{i(\gamma,t)}^t\| < \infty$  and  $\lim_t \|T_{j(\gamma,t)}^t\| < \infty$ . If  $\gamma' \supseteq \gamma * 3$ , then  $\gamma$  is working under the assumption that  $S_\gamma$  does not exist or is not isomorphic to any  $T_i$ . It is now straightforward to see that the true path is the union of all true assumptions.

Let  $\kappa_s$  be the greatest number used in the construction by stage  $s$ . At the first stage  $s$  that a module becomes active we set  $k(\alpha, s)$  to be  $\kappa_s + 5 \cdot 2^s$ . At each stage  $s$ , and for every index  $\alpha \not\supseteq \delta_s$  that is lexicographically to the right of  $\delta_s$ , we *initialise* module  $\alpha$ . That is, we set  $k(\alpha, s)$  to be  $\kappa_s + 5 \cdot 2^s$ ,  $l(\alpha, s)$  to be  $\infty$ , and  $i(\alpha, s)$  and  $j(\alpha, s)$  to be  $-1$ . If at any stage in the future module  $\alpha$  becomes active, we start it at step 1.

Notice that if  $\gamma$  is to the left of the true path, then it is active only finitely often; that if  $\gamma$  is on the true path then it is active infinitely often; and that  $\gamma$  is to the right of the true path if and only if it is initialised infinitely often.

*Analysis of the algorithm:*

We first prove that  $T$  has the form described above. The only nontrivial part is to show that  $T$  has no infinite components.

**Lemma 2.32.** *T has no infinite components.*

*Proof.* Suppose for a contradiction that  $n$  is least such that  $\|T_n\|$  is infinite. As  $T_n$  is of finite height, it must have infinitely many components. In the construction, components are added to  $T_n$  only by II above and in step 3. But as every  $T_m$  with  $m < n$  is finite, II will add only finitely many components to  $T_n$  during the construction. So step 3 adds infinitely many (type A) components to  $T_n$ .

Let  $t$  be a stage by when all  $T_m$  with  $m < n$  have stopped growing. The basic module ensures that any type A component of  $T_n$  gets extended to a type B or C component before another component is added to  $T_n$ . This can only happen in step 5 or step 6.

We divide into two cases:

Case 1. There are only finitely many stages such that  $T_n$  is extended by a module in step 5.

Let  $s \geq t$  be a stage such that  $T_n$  is not extended by any module in step 5 at any stage after  $s$ . Let  $\alpha$  be any module that extends a type A component of  $T_n$  at stage  $u \geq s$ . The claim is that after stage  $u$   $\alpha$  can never again act to extend  $T_n$ . To see that this is enough to establish case 1, consider all  $\gamma$  of length less than or equal to  $u$ . These are the only modules that potentially have been active by stage  $u$ . By the claim, each such  $\gamma$  can at most once act to extend  $T_n$  in step 6. But at most four nodes are added to  $T_n$  for each module that extends  $T_n$  in step 6 - three in step 5 and one in step 6. Therefore, all the modules of length less than or equal to  $u$  can add in total at most  $4 \cdot 2^u$  nodes to  $T_n$ .

Modules of length strictly greater than  $u$  can become active only after stage  $u$ . Let  $\beta$  be the first such module to act to extend  $T_n$  at stage  $u' > u$ . Then

$$k(\beta, u') = \kappa_{u'} + 5 \cdot 2^{u'} > \|T_n^u\| + 4 \cdot 2^u \geq \|T_n^{u'}\|.$$

And so  $\beta$  cannot affect  $T_n$ .

Thus only finitely many modules (those with indices of length less than or equal to  $u$ ) can affect  $T_n$  and these only finitely often each. Hence  $T_n$  is finite, contradiction.

We just now need to establish the claim. Once  $\alpha$  acts to extend  $T_n$  at stage  $u$ , the basic module returns to step 2 and searches for a  $T_i$  to be isomorphic to  $S_\alpha$ . If it never finds such a  $T_i$ , then  $\alpha$  can never

act to increase  $T_n$  again. To find a  $T_i$ ,  $S_\alpha$  must grow because by this stage all the components with index less than  $n$  are fixed.

But  $S_\alpha$  cannot increase to become isomorphic to  $T_n^{u+1}$  as, if it did, then the designated component of  $V_\alpha$  would embed both a type B and type C module.  $\phi_e$  would then not be of the required form, and the module would never exit step 1.

Hence  $S_\alpha$  must increase to become isomorphic to some  $T_p$  with  $p > n$ . Suppose this happens at stage  $v > u$ . Thus  $S_\alpha^v \simeq T_p^v$ . Thus at stage  $v$   $S_\alpha^v$  must have more type D components than  $T_n^u$ . But the number of type D components of  $T_n^t$  cannot change at any later stage as  $T_m$  is unchanging by stage  $t$  for all  $m < n$  (the number of type D components of a component can change only if a smaller component grows). Hence  $S_\alpha$  must always have more type D components than  $T_n$  after stage  $v$ , and  $\alpha$  can never again affect  $T_n$ .

Case 2. There are infinitely many stages such that  $T_n$  is extended by a module in step 5.

First we make the observation, useful later, that for any  $\gamma$ ,  $\|T_{i(\gamma,s)}^s\|$  and  $\|T_{j(\gamma,s)}^s\|$  are both increasing in  $s$  (because they are equal to  $\|S_\gamma^s\|$  and  $\|V_\gamma^s\|$ , which are both increasing with respect to  $s$ ).

As before, let  $t$  be a stage by when all  $T_m$  with  $m < n$  have stopped growing. Let  $\gamma$  and  $t'$  be any module and stage such that  $\gamma$  extends a type A component of  $T_n$  at stage  $t' > t$ . Thus  $j(\gamma, t') = n$ , and  $i(\gamma, t') = m$  for some  $m < n$ . As  $T_m$  is finite, there are only finitely many such modules  $\gamma$  (any module  $\gamma$  that becomes active for the first time after stage  $t'$  must choose  $\|S_\gamma\| > k > \|T_m\|$ ). Thus there is a module  $\gamma$  such that  $i(\gamma, u) = m$  for infinitely many  $u$  and consequently  $i(\gamma, u) = m$  for all sufficiently large  $u$ .

Let  $\beta$  be such a module and  $s > t$  a stage where  $\beta$  extends  $T_n$  in step 5. Let  $\alpha$  be the module that next extends  $T_n$  by adding a type A component at stage  $s' > s$ . Notice that  $T_n$  does not grow between stages  $s$  and  $s'$ .

Module  $\beta$  is active infinitely often, so it cannot be to the left of the true path. But if  $\beta$  were to the right of the true path, it would get initialised infinitely often and so eventually,  $i(\beta, t') > m$ , for all  $t'$  sufficiently large, contradicting the assumption on  $\beta$ . So  $\beta$  must be on the true path. Take  $\beta$  to be the longest such index.

As  $\beta$  is on the true path, we can assume that  $s$  is large enough so that after stage  $s$  no module strictly to the left of  $\beta$  is active. Hence  $\alpha$  is not to the left of  $\beta$ . If  $\alpha \not\geq \beta$  and were strictly to the right of  $\beta$ , then

$\alpha$  would have been initialised as soon as  $\beta$  acted and could no longer affect  $T_n$ . It is clear that  $\alpha \neq \beta$  as  $\lim_t i(\beta, t) < n = \liminf_t i(\alpha, t)$ . So  $\alpha \supsetneq \beta$  or  $\beta \supsetneq \alpha$ .

Suppose  $\beta \supsetneq \alpha$ . So  $\alpha$  is also on the true path. We can take  $s$  to be large enough so that  $i(\alpha, u) \neq -1$  for all  $u \geq s$ . We can also assume that  $s$  is large enough so that, for all  $\gamma \subseteq \beta$ , if  $\gamma$  extends  $T_n$  only finitely often, then  $\gamma$  never extends  $T_n$  after stage  $s$ . So we can assume  $\alpha$  extends  $T_n$  infinitely often. Therefore, as  $\alpha$  is on the true path,  $\beta \supseteq \alpha * 0$ .  $\beta$  extends  $T_n$  at stage  $s$ , and this can only happen if  $\|T_{j(\beta, s)}^s\| + 1 < \|T_{i(\alpha, s)}^s\|$  (as  $\|T_{j(\beta, s)}^s\| + 3 < l(\beta, s) \leq \min\{\|T_{i(\gamma, s)}^s\| : \gamma * 0 \subseteq \beta\}$ ). As  $\alpha$  is the first module to extend  $T_n$  after stage  $s$ , and  $\beta$  added exactly one node to  $T_n$  at stage  $s$ , we must have  $\|T_{i(\alpha, s')}^{s'}\| = \|T_{j(\beta, s)}^s\| + 1$ . But then, using the observation at the beginning of the proof,

$$\|T_{i(\alpha, s')}^{s'}\| = \|T_{j(\beta, s)}^s\| + 1 < \|T_{i(\alpha, s)}^s\| \leq \|T_{i(\alpha, s')}^{s'}\|,$$

contradiction.

Suppose  $\beta \subsetneq \alpha$ . There are now four cases:

1.  $\alpha \supseteq \beta * 0$ .  $\beta * 0$  is to the left of the true path (as  $\lim_t \|T_{i(\beta, t)}^t\| < \infty$ ) and so we can assume that  $s$  is so large that  $\beta * 0$  is never active at any stage  $t \geq s$ . Hence  $\alpha$  is never active at any stage  $t \geq s$ , contradiction.

2.  $\alpha \supseteq \beta * 1$ . Let

$$B = \{\gamma \subseteq \alpha : \text{infinitely often } \gamma \text{ extends a type A component of } T_n\}.$$

$\beta$  is then the longest element of  $B$ . As above,  $\alpha$  adds a component to  $T_n$  at stage  $s'$ . Suppose  $\beta'$  is the next element of  $B$  to extend  $T_n$ , say at stage  $s'' > s'$ . As  $\beta$  is on the true path, so are all  $\gamma \in B$ , and hence  $\alpha \supsetneq \gamma * 1$  for all  $\gamma \in B$ . So

$$\begin{aligned} \|T_{j(\beta', s'')}^{s''}\| &\geq \|T_{j(\beta', s')}^{s'}\| \\ &\quad \text{by the initial observation in the proof} \\ &\geq l(\alpha, s') \\ &> \|T_{i(\alpha, s')}^{s'}\| + 3 \\ &= \|T_n^{s'+1}\| \\ &\quad \text{as three nodes are added by } \alpha \text{ to } T_n \text{ at stage } s' \\ &= \|T_n^{s''}\| \\ &\quad \text{as } T_n \text{ is unchanged between stages } s' + 1 \text{ and } s'' \\ &= \|T_{j(\beta', s'')}^{s''}\|, \end{aligned}$$

contradiction.

3.  $\alpha \supseteq \beta * 2$ . At stage  $s$ ,  $\beta * 1$  is active, so  $\alpha$  is initialised. Hence  $\|T_{i(\alpha, s')}^{s'}\| \geq k(\alpha, s+1) > \|T_n^{s+1}\| = \|T_n^{s'}\| = \|T_{i(\alpha, s')}^{s'}\|$ , contradiction.

4.  $\alpha \supseteq \beta * 3$ . Finally,  $i(\beta, t) \neq -1$  for any  $t \geq s$ , as  $\beta$  acts to increase  $T_n$  at stage  $s$ . So no nodes extending  $\beta * 3$  are active after this time, contradiction □

**Lemma 2.33.** *All requirements are satisfied.*

*Proof.* Suppose not and let  $R_e$  be the least requirement that is not satisfied. Thus  $f_e : \phi_e \rightarrow \phi_e$  is a nontrivial self-embedding and  $\phi \simeq T$ . Let  $\alpha$  be the module on the true path of length  $e$ . So  $\alpha$  is active infinitely often and initialised only finitely often. Let  $s$  be a stage such that  $\alpha$  is never initialised after stage  $s$ . Thus  $k(\alpha, t) = k(\alpha, s)$  for all  $t \geq s$ . Let  $k = k(\alpha, s)$

Recall that if  $\alpha = \emptyset$  or  $\alpha(m) \notin \{0, 1\}$  for all  $m < |\alpha|$ , then  $l(\alpha, s) = \infty$ . Suppose now that  $\beta \subsetneq \alpha$  is such that  $\beta * 0 \subseteq \alpha$ . As  $\alpha$  is on the true path, it is active infinitely often and so there are infinitely many stages  $t$  such that  $T_{i(\beta, t+1)}^{t+1} \neq T_{i(\beta, t)}^t$ . Thus  $\lim_t \|T_{i(\beta, t)}^t\| = \infty$  for all  $\beta$  such that  $\beta * 0 \subseteq \alpha$ . By a similar argument,  $\lim_t \|T_{j(\beta, t)}^t\| = \infty$  for all  $\beta$  such that  $\beta * 1 \subseteq \alpha$ . Therefore  $\lim_t l(\alpha, t) = \infty$ . So we can assume that  $s$  is large enough so that there are distinct components of  $\phi_{e, s}$ , namely  $S_\alpha^s$  and  $V_\alpha^s$ , such that  $k < \|S_\alpha^s\| < l(\alpha, s)$  and  $\|V_\alpha^s\| + 3 < l(\alpha, s)$  and such that  $f_e$  takes  $S_\alpha^t$  into  $V_\alpha^t$  for all  $t \geq s$ . As  $f_e$  really is a self-embedding of  $\phi_e$ ,  $f_e$  must send  $\lim_t S_\alpha^t$  into  $\lim_t V_\alpha^t$ , and both of these components of  $\phi_e$  are finite. Let  $S_\alpha = \lim_t S_\alpha^t$  and  $V_\alpha = \lim_t V_\alpha^t$ , and suppose that  $s$  is large enough that  $S_\alpha^s = S_\alpha$  and  $V_\alpha^s = V_\alpha$ .

The above shows that there is at least one component of  $\phi_e$  that is available to be chosen as  $S_\alpha$  in step 1 of the module. This choice will never be changed as  $\alpha$  is never initialised after stage  $s$ . Hence the first choice  $\alpha$  makes after it has been initialised for the last time, will be its final choice for  $S_\alpha$ .  $\alpha$  will then move into step 2 of the module and never return to step 1.

As  $\phi_e \simeq T$ , there must be an  $i$  and a  $j > i$  such that  $T_i \simeq S_\alpha$  and  $T_j \simeq V_\alpha$ . During the operation of the module, if at any stage  $t \geq s$ ,  $T_{i(\alpha, t)}^t \not\cong S_\alpha^t = S_\alpha$ , the module will return to step 1 and search for a new  $T_i$ . As no two distinct components of  $T$  are isomorphic, this process will continue until a stage  $t \geq s$  is arrived at such that for all  $t' \geq t$ ,  $i(\alpha, t') = i$  and  $S_\alpha = S_\alpha^{t'} \simeq T_{i(\alpha, t')}^{t'} = T_i$ .

We assume that  $s$  is large enough so that the above is true for  $s$ . Thus  $\alpha$  never returns to step 2 after stage  $s$ . So  $\alpha$  eventually leaves step 2 for the last time and moves to step 3, say at stage  $s' \geq s$ .  $T_i = T_i^{s'}$  and so  $T_i$  must have a unique type A component (or else one is added and  $T_i^{s'} \neq T_i$ ). Thus the module moves into step 4.

By an argument similar to above, there is a stage  $s'' \geq s'$  such that  $j(\alpha, s'') = j$ , and so the module eventually moves into step 5. As  $T_j$  has ceased growing by stage  $s''$ , it must have no type A components and the module moves into step 6.

At step 6 the module adds a node to  $T_i$ , contradicting the fact that  $T_i$  has stopped growing. Thus  $R_e$  is satisfied.  $\square$

$\square$

The proof of Theorem 2.31 can be simply adapted to give a proof of the following.

**Theorem 2.34.** *There is a computable, binary branching tree  $S$  with exactly one infinite path such that no computable tree classically isomorphic to  $S$  has a computable nontrivial self-embedding.*

*Proof.* Let  $\langle T, \preceq_T \rangle$  be the tree constructed in Theorem 2.31,  $\langle T_i \rangle$  its sequence of components, and  $\lambda_T$  its root. Also let  $\lambda_i$  denote the root of component  $T_i$ . Let  $A = \{a_i : i \in \mathbb{N}\}$  be a set of distinct elements disjoint from  $T$ . We define the tree  $\langle S, \preceq_S \rangle$  as follows:

1.  $S = (T \setminus \{\lambda_T\}) \cup A$
2.  $\forall x, y \in S \quad x \preceq_T y \implies x \preceq_S y$
3.  $\forall i \in \mathbb{N} \quad a_{i+1}$  and  $\lambda_i$  are immediate successors of  $a_i$ .

It is straightforward to see that there is a unique tree  $S$  satisfying 1, 2 and 3 and that  $S$  has exactly the one infinite path  $a_0 a_1 a_2 \dots$ .

Also notice that any nontrivial self-embedding  $\delta$  of  $S$  computes a nontrivial self embedding of  $T$ , and hence by Theorem 2.31,  $\delta$  must be noncomputable.

To see this, as  $a_0 a_1 a_2 \dots$  is the only infinite path in  $S$ ,  $\delta$  must take this path strictly into itself. Let  $k$  be least such that  $\delta(a_k) \neq a_k$ . Then for all  $i \geq k$  there exists a unique  $j$  such that  $\delta(a_i) = a_j \succ a_i$  and  $\delta$  embeds  $T_i$  into  $T_j$ .

But then  $\delta$  induces a nontrivial self-embedding  $\delta'$  of  $T$  given by

1.  $\delta'(\lambda_T) = \lambda_T$
2. for all  $x \in T \setminus \{\lambda_T\} \quad \delta'(x) = \delta(x)$ .

To prove the theorem, let  $\langle U, \preceq_U \rangle$  be any computable tree classically isomorphic to  $S$ . We show that any nontrivial self-embedding  $\delta$  of  $U$  computes a nontrivial self-embedding of a computable tree classically isomorphic to  $T$ . Hence by Theorem 2.31,  $\delta >_T 0$ .

As  $U$  is isomorphic to  $S$ , it too has exactly one infinite path. Let  $b_0b_1b_2\dots$  be that path and let  $B = \{b_i : i \in \mathbb{N}\}$ . We first claim that  $B$  is computable. It is c.e. because  $x \in B$  if and only if there exists a chain of height 4 above  $x$  (recall all the components of  $T$  have heights at most 3). And it is co-c.e. because  $x \notin B$  if and only if there exists a  $b \in B$  such that  $b$  and  $x$  are incomparable.

So we can build a computable tree  $\langle V, \preceq_V \rangle$  isomorphic to  $T$  as follows. Let  $V = (U \setminus B) \cup \{\rho\}$ , where  $\rho$  is a new element which will serve as the root of  $V$ . For all  $x, y \in V$  define  $x \preceq_V y$  if and only if  $x = \rho$  or  $x \preceq_U y$ . As  $B$  is computable, so is  $\preceq_V$ , and it is easy to see that  $V$  is isomorphic to  $T$ .

Now from  $\delta$  we can now compute a nontrivial self-embedding  $\delta'$  of  $V$ . Let  $\delta'(\rho) = \rho$  and for all  $x \in U \setminus B$  let  $\delta'(x) = \delta(x)$ .  $\delta'$  is clearly computable from  $\delta$  and hence  $\delta >_T 0$ .  $\square$

**Theorem 2.35.** *There is a computable finitely branching tree  $T$  with no isolated path or maximal infinite node such that for any computable tree  $S$  classically isomorphic to  $T$ , if  $\varphi$  is a nontrivial self-embedding of  $S$ , then  $0' \not\leq_T \varphi$ .*

*Proof.* This follows easily from Theorem ???. Take  $T$  to be the tree in that theorem. If  $S$  were a computable tree classically isomorphic to  $T$  and  $\phi$  a nontrivial self embedding. Then  $\phi$  computes a function that dominates  $\text{brl}_S$  using the argument from Theorem ??. As  $T$  is isomorphic to  $S$ ,  $\phi$  also computes a function that dominates  $\text{brl}_T$ . Therefore  $\phi$  computes a function that computes  $0''$ .  $\square$

These results leave open some questions. Is it possible to code  $0'$  into the construction of Theorem 2.31 to build a tree with the property that any nontrivial self-embedding of an isomorphic computable tree compute  $0'$ . A similar question could be asked about the construction used to prove Theorem 2.30: is there a way to code  $0''$  into the isomorphism type of a type 3 tree?

One interesting difference between theorems 2.31 and 2.30 is the latter proves result about all trees isomorphic to a given computable tree while the former only considers *computable* trees isomorphic to a given computable tree. This is a consequence of the different types of

construction used. Theorem 2.30 encodes  $0''$  into the branching levels of a tree - and so information is preserved by *any* isomorphic tree - computable or not (relative to its successor relation). Theorem 2.31 uses a standard diagonalisation method against computable trees to eliminate the possibility that our tree is isomorphic to a computable tree with a computable nontrivial self-embedding. It has nothing to say about non-computable trees

### 3 Chains and antichains in trees

One simple application of Ramsey's Theorem for pairs with two colors is that any infinite partial order must have either an infinite chain or an infinite antichain. Herrmann [1] examined the effective content of this result and proved the following theorem.

**Theorem 3.1 (Herrmann [1]).** *If  $P$  is an infinite computable partial order, then  $P$  has either an infinite  $\Delta_2^0$  chain or an infinite  $\Pi_1^0$  antichain. In addition, there is an infinite computable partial order which has no infinite  $\Sigma_1^0$  chains or antichains.*

In this section, we consider this result in the context of trees rather than general partial orders and we show that for trees these results can be improved by exactly one quantifier.

**Theorem 3.2.** *Let  $T$  be an infinite computable tree.  $T$  has either an infinite computable chain or an infinite  $\Pi_1^0$  antichain.*

*Proof.* If  $T$  has infinitely many leaves, then the set of leaves is an infinite  $\Pi_1^0$  antichain. Otherwise,  $T$  must have a node  $x$  such that  $T(x)$  is infinite and contains no leaves. In this case, let  $x_0 = x$  and  $x_{i+1}$  be the  $\leq_{\mathbb{N}}$  least element of  $T$  which satisfies  $x_i \prec x_{i+1}$ . The sequence  $x_0, x_1, \dots$  gives an infinite computable chain.  $\square$

**Theorem 3.3.** *There is an infinite binary branching computable tree such that  $T$  has no infinite c.e. chains or antichains.*

*Proof.* We build  $(T, \preceq)$  to meet the following requirements.

$$\begin{aligned} R_{2e} : W_e \text{ is not an infinite chain} \\ R_{2e+1} : W_e \text{ is not an infinite antichain} \end{aligned}$$

We build  $T$  in stages beginning with  $T_0 = \{\lambda\}$ . Throughout the construction, we maintain the property that each node  $x$  is either

currently a leaf or else has exactly two immediate successors. Each requirement  $R_i$  keeps a parameter  $r_i$  such that any node  $x$  added to  $T$  by a lower priority requirement after  $r_i$  is defined satisfies  $r_i \preceq x$ . For uniformity of notation, we set  $r_{-1} = \lambda$ . If a strategy is initialized, then all of its parameters become undefined. Any parameter not explicitly redefined or undefined by initialization retains its current value. If a requirement ends the current stage, then it initializes all lower priority requirements. The action for  $R_{2e}$  at stage  $s$  is as follows.

1. If  $s$  is the first stage at which  $R_{2e}$  is eligible to act or if  $R_{2e}$  has been initialized since it was last eligible to act, let  $a$  be such that  $r_{2e-1} \preceq a$  and  $a$  is a leaf in  $T_s$ . Add new nodes  $b$  and  $c$  to  $T_s$  as immediate successors of  $a$ . Set  $r_{2e} = b$  and end the stage.
2. If  $r_{2e}$  is defined but  $R_{2e}$  has not succeeded yet, then check whether there is a node  $x \in T_s$  such that  $r_{2e} \preceq x$  and  $x \in W_{e,s}$ . If not, then let  $R_{2e+1}$  act. If so, then let  $z$  denote the immediate predecessor of  $x$ . Since  $z$  is not a leaf, it has two immediate successors. Let  $y$  denote the successor of  $z$  which is not equal to  $x$ . Redefine  $r_{2e} = y$  and end the stage. We say that  $R_{2e}$  has succeeded.
3. If  $R_{2e}$  has succeeded, then let  $R_{2e+1}$  act.

The action for  $R_{2e+1}$  at stage  $s$  is as follows.

1. If  $s$  is the first stage at which  $R_{2e+1}$  is eligible to act or if  $R_{2e+1}$  has been initialized since it was last eligible to act, define  $r_{2e+1} = r_{2e}$ . End the stage.
2. If  $r_{2e+1}$  is defined and  $R_{2e+1}$  has not succeeded yet, check whether there is a node  $x \in T_s$  such that  $r_{2e+1} \preceq x$  and  $x \in W_e$ . If not, then let  $R_{2e+2}$  act. If so, then redefine  $r_{2e+1} = x$  and end the stage. We say  $R_{2e+1}$  succeeds.
3. If  $R_{2e+1}$  has succeeded, then let  $R_{2e+2}$  act.

This argument is finite injury so each parameter reaches a limit. Because nodes are added to  $T$  only in Step 1 of the  $R_{2e}$  action,  $T$  has the property that at each stage, each node is either currently a leaf or has exactly two successors.

To see that  $R_{2e}$  is met, let  $s$  be the least stage such that  $R_{2e}$  is never initialized after stage  $s$ . The parameter  $r_{2e}$  is defined at stage  $s$  and can only change values after stage  $s$  if  $R_{2e}$  changes the value in Step 2 of its action.

There are two cases to consider. First, if there is a stage  $t > s$  and a node  $x \in T_t$  such that  $r_{2e} \preceq x$  and  $x \in W_{e,s}$ . In this case,

$r_{2e}$  is redefined so that  $r_{2e}$  is incompatible with  $x$ . Because  $r_{2e}$  is not changed again and because no strategy of higher priority than  $R_{2e}$  adds elements to  $T$  after stage  $t$ , there are only finitely many elements in  $T$  which are not above this final value of  $r_{2e}$ . Therefore,  $x$  cannot be part of an infinite chain and  $R_{2e}$  is met.

The second case is when there is no such stage  $t$  and node  $x$ . In this case,  $r_{2e}$  has reached its limit at stage  $s$  and every node added to  $T$  after stage  $s$  is added above  $r_{2e}$ . Because there are only finitely many nodes in  $T$  which are not above  $r_{2e}$ , there cannot be an infinite chain which is disjoint from  $T(r_{2e})$ . Therefore,  $R_{2e}$  is met.

The argument that  $R_{2e+1}$  is met is quite similar. Let  $s$  be the least stage such that  $R_{2e+1}$  is never initialized after  $s$  and let  $r_{2e+1}$  denote the value of the parameter at stage  $s$ . If there is no node  $x \in W_e$  such that  $r_{2e+1} \preceq x$ , then  $R_{2e+1}$  never changes the value of  $r_{2e+1}$  and there are only finitely many nodes of  $T$  which are not above  $r_{2e+1}$ . Any infinite antichain must intersect  $T(r_{2e+1})$ , so  $R_{2e+1}$  is met.

If there is a node  $x \in W_e$  such that  $r_{2e+1} \preceq x$ , then let  $x$  be the first such node seen by  $R_{2e+1}$  after stage  $s$ . At this point,  $r_{2e+1}$  is redefined to be equal to  $x$ , so there are only finitely many nodes in  $T$  which are not comparable to  $x$ . Hence, no set containing  $x$  can be an infinite antichain. Therefore,  $R_{2e+1}$  is won.  $\square$

**Theorem 3.4.** *Let  $L$  be any low set. There is an infinite binary branching computable tree  $T$  such that  $T$  has no infinite chains or antichains computable from  $L$ .*

*Proof.* We need to meet the following requirements.

$$\begin{aligned} R_{2e} &: \varphi_e^L \text{ is not an infinite chain} \\ R_{2e+1} &: \varphi_e^L \text{ is not an infinite antichain} \end{aligned}$$

As in the proof of Theorem 3.3, we build  $T$  in stages and maintain the property that each node  $x$  is either currently a leaf or else has exactly two immediate successors. Each requirement  $R_i$  keeps a parameter  $r_i$  as before. The main change in this construction is that the value of  $r_i$  can change more than once (but still only finitely often) after the last time  $R_i$  is initialized. As before, whenever a requirement ends a stage, it initializes all lower priority requirements.

$R_{2e}$  keeps three parameters:  $r_{2e}$ ,  $\hat{r}_{2e}$  and  $x_{2e}$ . The  $r_{2e}$  parameter is used as before to force lower priority requirements to work above  $r_{2e}$ . The  $\hat{r}_{2e}$  parameter is used to store an “old value” of  $r_{2e}$  in case

our approximations to computations from  $L$  change and we need to revert back to an earlier situation and wait for reconvergence. The  $x_{2e}$  parameter will be explained when it appears in the construction below.

When  $R_{2e}$  first acts or if  $R_{2e}$  has been initialized since its last action, it lets  $a$  be a node such that  $r_{2e-1} \preceq a$  and  $a$  is currently a leaf. It adds two new nodes  $b$  and  $c$  as immediate successors of  $a$  in  $T$ , defines  $\hat{r}_{2e} = r_{2e} = b$  and ends the stage. At future stages  $s$ ,  $R_{2e}$  requires lower priority strategies to work above  $r_{2e}$  and it tries to decide whether  $\exists x \exists t (r_{2e} \preceq x \wedge \varphi_{e,t}^L(x) = 1)$ . This predicate is  $\Sigma_1^L$ , so it is computable from  $L'$  and hence from  $0'$  (since  $L$  is low). Fix the  $\Delta_2^0$  predicate  $P(e, k)$  defined by

$$P(e, k) \Leftrightarrow \exists x \exists t (k \preceq x \wedge \varphi_{e,t}^L(x) = 1).$$

Let  $P(e, k, s)$  be a computable approximation such that  $P(e, k) = \lim_s P(e, k, s)$ .

At stage  $s$ ,  $R_{2e}$  checks whether  $P(e, \hat{r}_{2e}, s) = 1$ . (Notice that  $\hat{r}_{2e} = r_{2e}$  at this point, so  $R_{2e}$  is really checking whether  $P(e, r_{2e}, s) = 1$ .) If not, then  $R_{2e}$  has no need to diagonalize and it lets  $R_{2e+1}$  act. If so, then  $R_{2e}$  wants to find the  $\leq_{\mathbb{N}}$  least witness  $x$  which appears to satisfy this existential statement. We define a second  $\Delta_2^0$  predicate  $Q(x, u)$  by

$$Q(x, u) \Leftrightarrow \forall t (t \geq u \rightarrow \varphi_{e,t}^L(x) = 1)$$

and fix a computable approximation  $Q(x, u, s)$  such that  $Q(x, u) = \lim_s Q(x, u, s)$ . (The predicate  $Q(x, u)$  is computable from  $L'$ , so it is  $\Delta_2^0$  because  $L$  is low.)  $R_{2e}$  looks for the  $\leq_{\mathbb{N}}$  least  $x$  such that  $\hat{r}_{2e} = r_{2e} \preceq x$  and  $Q(x, s, s)$ . If there is no such  $x$  then  $R_{2e}$  lets  $R_{2e+1}$  act and waits to check again at the next stage. Eventually, it must find an  $x$  and  $s$  for which  $Q(x, s, s)$ . (Of course, if  $P(e, \hat{r}_{2e}, s)$  changes from value 1 to value 0 while  $R_{2e}$  is waiting for such an  $x$ , it ends the stage and returns to waiting for  $P(e, \hat{r}_{2e}, s)$  to have value 1.)

When  $R_{2e}$  finds such an  $x$ , it defines its third parameter  $x_{2e} = x$ . Let  $z$  be the immediate predecessor of  $x_{2e}$  and let  $y$  be the successor  $z$  which is not equal to  $x_{2e}$ .  $R_{2e}$  sets  $r_{2e} = y$  but leaves the value of  $\hat{r}_{2e}$  unchanged. That is,  $\hat{r}_{2e}$  retains the “old value” of  $r_{2e}$ .  $R_{2e}$  ends the stage (and hence initializes the lower priority strategies so that they will work above the new value of  $r_{2e}$  in the future).

If  $P(e, \hat{r}_{2e})$  really holds and  $x_{2e}$  really is a correct witness for this existential statement, then we have successfully diagonalized. However, it is possible that either  $P(e, \hat{r}_{2e})$  does not hold or that  $x_{2e}$  is not

a correct witness. Therefore, at each future stage  $s$ ,  $R_{2e}$  continues to check whether  $P(e, \hat{r}_{2e}, s) = 1$ . If this value ever changes to 0, then  $R_{2e}$  redefines  $r_{2e}$  to have value  $r_{2e} = \hat{r}_{2e}$ , cancels its parameter  $x_{2e}$ , ends the stage and returns to waiting for  $P(e, \hat{r}_{2e}, s) = 1$ . If  $P(e, \hat{r}_{2e}, s)$  retains its value of 1, then  $R_{2e}$  checks whether  $Q(x_{2e}, s, s)$  still gives the value 1. If so, then  $R_{2e}$  continues to believe it has correctly diagonalized and lets  $R_{2e+1}$  act. If  $Q(x_{2e}, s, s) = 0$  at some future stage  $s$  (while  $P(e, \hat{r}_{2e}, s) = 1$ ), then  $R_{2e}$  cancels the parameter  $x_{2e}$ , redefines  $r_{2e}$  to have value  $r_{2e} = \hat{r}_{2e}$ , ends the stage and returns to looking for the  $\leq_{\mathbb{N}}$  least  $x$  such that  $Q(x, s, s) = 1$ .

To understand why this strategy eventually succeeds, let  $u$  be a stage such that  $R_{2e}$  is never initialized after  $u$ . At stage  $u$ ,  $R_{2e}$  defines  $r_{2e}$  and  $\hat{r}_{2e}$  and it will never change the value of  $\hat{r}_{2e}$  again. (The entire construction is finite injury so there is such a stage.) Because  $P(e, \hat{r}_{2e})$  is a  $\Delta_2^0$  predicate, there is a  $t \geq u$  such that for all  $s \geq t$ ,  $P(e, \hat{r}_{2e}, s)$  is either constantly 0 or constantly 1.

If  $P(e, \hat{r}_{2e}, s)$  is eventually constantly 0, then  $r_{2e}$  will eventually be set permanently equal to  $\hat{r}_{2e}$ . From this stage on, all nodes added to  $T$  are above  $\hat{r}_{2e}$ . Because  $P(e, \hat{r}_{2e})$  does not hold,  $\varphi_e^L$  does not place any elements from  $T(\hat{r}_{2e})$  into its chain. Because there are only finitely many elements of  $T$  outside of  $T(\hat{r}_{2e})$ ,  $\varphi_e^L$  cannot compute an infinite chain and  $R_{2e}$  is met.

If  $P(e, \hat{r}_{2e}, s)$  is eventually constantly 1, then there is an  $x$  such that  $\hat{r}_{2e} \preceq x$  and  $x$  is a witness to the existential statement  $P(e, \hat{r}_{2e})$ . Because  $Q(x, u, s)$  is a  $\Delta_2^0$  predicate and we look for the  $\leq_{\mathbb{N}}$  least witness  $x$ ,  $R_{2e}$  eventually defines  $x_{2e}$  such that  $Q(x_{2e}, s, s)$  has reached its limit of 1. Both  $r_{2e}$  and  $x_{2e}$  have reached their limits at this stage. After this stage, all elements added to  $T$  are above  $r_{2e}$  and hence are incomparable with  $x_{2e}$ . Because  $\varphi_e^L(x_{2e}) = 1$ ,  $\varphi_e^L$  cannot compute an infinite chain in  $T$  so  $R_{2e}$  is met.

In either case, notice that  $\hat{r}_{2e}$  and  $r_{2e}$  reach limits and that  $x_{2e}$  either reaches a limit or there is a stage after which it is never defined. Therefore,  $R_{2e}$  only initializes lower priority strategies finitely often.

The strategy to meet requirement  $R_{2e+1}$  is similar.  $R_{2e+1}$  also keeps three parameters  $r_{2e+1}$ ,  $\hat{r}_{2e+1}$  and  $x_{2e+1}$ . When it first acts (or after it has been initialized),  $R_{2e+1}$  sets  $\hat{r}_{2e+1} = r_{2e+1} = r_{2e}$  and ends the stage.

At future stages,  $R_{2e+1}$  checks whether  $P(e, \hat{r}_{2e+1}, s) = 1$ . If not, it lets  $R_{2e+2}$  act. If so, it looks for the  $\leq_{\mathbb{N}}$  least  $x$  such that  $Q(x, s, s) = 1$ . If there is no such  $x$ , it lets  $R_{2e+2}$  act next. If there is such an  $x$ , it sets

$x_{2e+1} = x$ , redefines  $r_{2e+1}$  so that  $r_{2e+1} = x_{2e+1}$  and ends the stage. (As above, it leaves  $\hat{r}_{2e+1}$  unchanged to mark the “old value” of  $r_{2e+1}$ . If  $P(e, \hat{r}_{2e+1}, s)$  changes values from 1 to 0 while  $R_{2e+1}$  is waiting for such an  $x$ , it ends the stage and returns to waiting for  $P(e, \hat{r}_{2e+1}, s)$  to equal 1.)

Once  $x_{2e+1}$  is defined,  $R_{2e+1}$  continues to check whether  $P(e, \hat{r}_{2e+1}, s) = 1$ . If this value ever changes to 0, it cancels  $x_{2e+1}$ , redefines  $r_{2e+1}$  so that  $r_{2e+1} = \hat{r}_{2e+1}$ , ends the stage and returns to waiting for  $P(e, \hat{r}_{2e+1}, s)$  to equal 1. As long as  $P(e, \hat{r}_{2e+1}, s)$  remains equal to 1,  $R_{2e+1}$  checks whether  $Q(x_{2e+1}, s, s)$  continues to equal 1. As long as it does,  $R_{2e+1}$  lets  $R_{2e+2}$  act. If  $Q(x_{2e+1}, s, s)$  changes values to 0, then  $R_{2e+1}$  cancels  $x_{2e+1}$ , redefines  $r_{2e+1}$  to have value  $r_{2e+1} = \hat{r}_{2e+1}$ , ends the stage and returns to looking for the  $\leq_{\mathbb{N}}$  least  $x$  such that  $Q(x, s, s) = 1$ .

The analysis that  $R_{2e+1}$  eventually succeeds and that it initializes the lower priority requirements only finitely often is similar to the analysis given for  $R_{2e}$ . The details are left to the reader.  $\square$

There is no need to restrict ourselves to a single low set  $L$  in the proof of Theorem 3.4. That is, essentially the same proof (with a little extra bookkeeping in the indices) shows that if  $L_i$  (for  $i \in \mathbb{N}$ ) is a sequence of uniformly low, uniformly  $\Delta_2^0$  sets, then there is an infinite binary branching computable tree  $T$  such that  $T$  has no infinite chains and no infinite antichains computable from any of the  $L_i$  sets. By Jockusch and Soare [2] and Simpson [3], there is an  $\omega$ -model  $\mathcal{M}$  of  $WKL_0$  such that the second order part of  $\mathcal{M}$  consists of all the sets in the Turing ideal generated by a sequence  $L_0 \leq_T L_1 \leq_T \dots$  of uniformly low, uniformly  $\Delta_2^0$  sets. Thus, we obtain the following corollary.

**Corollary 3.5.**  *$WKL_0$  is not strong enough to prove that every infinite binary branching tree has either an infinite chain or an infinite antichain.*

One potential point of confusion here is the definition of “binary branching.” A binary branching tree  $T$  in our sense is a tree  $T$  for which each node has at most two successors. Although classically this definition is the same as saying  $T$  is a subset of  $2^{<\omega}$  which is closed under initial segments, these definitions are not the same in a weak subsystem such as  $WKL_0$ . For further discussion of this point, see Section III.7 of Simpson [3].

## References

- [1] E. Herrmann, Infinite chains and antichains in computable partial orderings, *Journal of Symbolic Logic* **66** (2001), 923-934.
- [2] C.G. Jockusch Jr. and R.I. Soare,  $\Pi_1^0$  classes and degrees of theories, *Transactions of the American Mathematical Society*, **173** (1972), 33-56.
- [3] S.G. Simpson, *Subsystems of second order arithmetic*, Springer-Verlag, Berlin, 1999.