When does a relation code an isomorphism?

Barbara F. Csima[*], Michael Deveau[†], and Jonathan Stephenson[‡]

Department of Pure Mathematics
University of Waterloo
Waterloo, ON N2L 3G1, Canada

November 20, 2017

Abstract

For $\mathcal{A}$ and $\mathcal{B}$ computable copies of the same structure, if U is a computable subset of the domain of $\mathcal{A}$ and $f : \mathcal{A} \cong \mathcal{B}$, then certainly $f(U) \leq_T f$. When we know something about the structure, we can often find a computable U such that $f(U) \equiv_T f$. Indeed, the method of the artfully chosen U is how Turing degrees of isomorphisms between particular copies of a structure are normally computed. This paper examines the question, to what extent is this possible? We focus on the linear orders $(\omega, <)$ and $(\omega^2, <)$ as our examples.

[*]csima@uwaterloo.ca
[†]m2deveau@uwaterloo.ca
[‡]jonny.stephenson@valpo.edu

1 Introduction

In computable structure theory, we are interested in studying mathematical structures from a computable point of view, so it is natural to regard two computable copies of the same structure are being equivalent if they are isomorphic via some computable isomorphism. However, there are many examples of very standard structures for which there are computable copies which are not computably isomorphic. For example, consider $(\omega, <)$, the linear order with order type ω. We let $\mathcal{N}$ denote the usual decidable copy. One can readily construct another computable copy $\mathcal{A}$ of $(\omega, <)$ such that the unique isomorphism between $\mathcal{N}$ and $\mathcal{A}$ is of Turing degree $\mathbf{0}'$. In fact, in the case of $(\omega, <)$, it is easy to show that $\mathbf{0}'$ can compute the isomorphism between any two computable copies. So there is a sense in which $\mathbf{0}'$ is the degree of difficulty of computing isomorphisms between copies of $(\omega, <)$. This is made precise in the following definition.

Definition 1.1.

Let $\mathcal{A}$ be a computable structure, and suppose that $\mathbf{d}$ is a Turing degree which can compute an isomorphism between any two computable copies of $\mathcal{A}$. Then we say that $\mathcal{A}$ is $\mathbf{d}$-*computably categorical*.

If $\{\mathbf{c} \mid \mathcal{A} \text{ is } \mathbf{c}\text{-computably categorical }\} = \{\mathbf{c} \mid \mathbf{c} \geq \mathbf{d}\}$, then we say that $\mathbf{d}$ is *the degree of categoricity* of $\mathcal{A}$.

If $\mathcal{A}$ has degree of categoricity $\mathbf{d}$ and there exist computable copies $\mathcal{A}_1$ and $\mathcal{A}_2$ of $\mathcal{A}$ such that every isomorphism $f : \mathcal{A}_1 \cong \mathcal{A}_2$ computes $\mathbf{d}$, we say $\mathcal{A}$ has *strong* degree of categoricity $\mathbf{d}$.

Finally, we say $\mathbf{d}$ is a (strong) degree of categoricty if there exists a computable structure with (strong) degree of categoricity $\mathbf{d}$.

Degrees of categoricity were introduced by Fokina, Kalimullin and Miller [FKM10], who showed that every degree $\mathbf{d}$ which is 2-c.e. in and above $\mathbf{0}^{(n)}$

for some $n \in \omega$ is a strong degree of categoricity. This was extended by Csima, Franklin and Shore [CFS13] to $\mathbf{0}^{(\alpha)}$ for any computable ordinal α, and degrees 2-c.e. in and above $\mathbf{0}^{(\alpha)}$ for computable successor ordinals α.

Bazhenov, Kalimullin, and Yamaleev [BKY] as well as Csima and Stephenson [CS] constructed structures that have a degree of categoricity, but no strong degree of categoricity.

All of the above papers use the same approach for computing the degree of categoricity of the structures constructed. The structures are built in such a way that there is a computable unary relation U on one of the copies of the structure, so that if f is the isomorphism between the copies, then the isomorphism has degree $f(U)$.

This raises the natural question: To what extent is this possible?

The reader may note that we are restricting attention to computable subsets of the domain A of a structure $\mathcal{A}$, rather than, for instance, working within A^n for arbitrary n, or even in $A^{<\omega}$. Apart from our original motivating observations, there is a very straightforward reason for this restriction: if we work with tuples instead, then we can trivially recover the degree of an isomorphism f from $f(R)$, where R is chosen entirely independently of the structure we are working with:

Observation 1.2.

If $\mathcal{A}$ is a computable structure and $f : \mathcal{A} \to \mathcal{B}$ is an isomorphism to another computable copy $\mathcal{B}$, let $R := \{(n, n+1)\}_{n \in \omega}$. Then $f(R) = \{(f(n), f(n+1))\}$ computes f.

Let us first examine the situation for unary relations on the linear order $(\omega, <)$. We begin with the following proposition.

Proposition 1.3.

Let $\mathcal{A}$ be any computable copy of $(\omega, <)$, and let $\mathcal{N}$ denote the standard decidable copy. Let $f : \mathcal{A} \to \mathcal{N}$ be the isomorphism between the two copies. Let $U \coloneqq \{m \mid (\exists n)[n <^{\mathcal{N}} m \wedge m <^{\mathcal{A}} n]\}$. Then $f(U) \equiv_T f$.

Proof Sketch:

Suppose we are given $f(U)$. We show how to build the isomorphism f. First, find the least member of $\overline{f(U)}$, say n_0. Then we know there are exactly n_0-many numbers that are $<^{\mathcal{A}}$-below 0. We reveal the order $<^{\mathcal{A}}$ until we find $a_0 <^{\mathcal{A}} a_1 <^{\mathcal{A}} \cdots <^{\mathcal{A}} a_{n_0-1} <^{\mathcal{A}} a_{n_0} = 0$ and define $f(a_i) = i$. We have now defined f on an initial segment, and proceed inductively. $\square$

However, there is an asymmetry with $(\omega, <)$, as we will see that there exists a computable copy $\mathcal{A}$ of $(\omega, <)$ such that for $f : \mathcal{N} \cong \mathcal{A}$ there is no computable U with $f(U) \equiv_T f$. Indeed, this will follow from an easy modification to the proof of the following Theorem.

Theorem 1.4.

There are two computable copies $\mathcal{A}$ and $\mathcal{B}$ of $(\omega, <)$ such that if $f : \mathcal{A} \to \mathcal{B}$ is the isomorphism between them, then f is of Turing degree $\mathbf{0}'$, and there is no computable set U such that $f(U) \equiv_T f$ or $f^{-1}(U) \equiv_T f$.

Note that Proposition 1.3 implies that for any computable copy $\mathcal{A}$ of ω, if $f : \mathcal{A} \to \mathcal{N}$ is the isomorphism between $\mathcal{A}$ and the usual decidable copy of the order, there is a computable set U such that $f(U) \equiv_T f$; this exposes a fundamental distinction between effectiveness of isomorphisms mapping *into* and *out of* the standard copy $\mathcal{N}$ of ω.

One might wonder whether this phenomenon is somehow more about mapping into and out of decidable structures, rather than about the particular choice of the structure $(\omega, <)$. This raises the question:

Question 1.5.

Suppose $\mathcal{A}$ is a computable structure and that $\mathcal{B}$ is a decidable copy isomorphic to $\mathcal{A}$. Suppose that the structures are rigid and that $f : \mathcal{A} \to$

$\mathcal{B}$ is the isomorphism between them. Must there be a computable U such that $f(U) \equiv_T f$?

This conjecture turns out to be rather easy to dismiss; indeed, all we need to do is to look at $(\omega^2, <)$ rather than $(\omega, <)$.

Theorem 1.6.
Let $\mathcal{N}^2$ be a decidable copy of $(\omega^2, <)$. There is a computable copy $\mathcal{A}$ of $(\omega^2, <)$ such that if $f : \mathcal{A} \to \mathcal{N}^2$, then for no computable unary relation U on $\mathcal{A}$ do we have $f(U) \equiv_T f$.

We follow standard notation for computability theory, as found in Soare [Soa16]. Section 2 is devoted to the proof of Theorem 1.4, Section 3 to proof of Theorem 1.6, and we close with further thoughts on future directions in Section 4.

2 Isomorphisms on copies of $(\omega, <)$

This section is devoted to the proof of Theorem 1.4, which we restate here.

Theorem 2.1.
There are two computable copies $\mathcal{A}$ and $\mathcal{B}$ of $(\omega, <)$ such that if $f : \mathcal{A} \to \mathcal{B}$ is the isomorphism between them, then f is of Turing degree $\mathbf{0}'$, and there is no computable set U such that $f(U) \equiv_T f$ or $f^{-1}(U) \equiv_T f$.

Proof:
We aim to meet, for all $e, j \in \omega$, the following requirements:

$\mathbf{R}_{\langle e,j \rangle}$: If $\varphi_e = \chi_U$ for some set U, then $(\exists x)[\Phi_j^{f(U)}(x) \neq f(x)]$, and

$\mathbf{S}_{\langle e,j \rangle}$: If $\varphi_e = \chi_U$ for some set U, then $(\exists x)[\Phi_j^{f^{-1}(U)}(x) \neq f^{-1}(x)]$.

To do this, we will build $\mathcal{A}$ and $\mathcal{B}$ by stages, enumerating the least unused element into the domains of $\mathcal{A}$ and $\mathcal{B}$, and perhaps more, at each stage. We will enforce that there are only finitely many enumerations at any given position, so that $\mathcal{A}$ and $\mathcal{B}$ are isomorphic to $(\omega, <)$. At each stage s, we let $\mathcal{A}_s$ and $\mathcal{B}_s$ denote the partially constructed portions of their respective structures at that stage, and f_s the partial isomorphism between $\mathcal{A}_s$ and $\mathcal{B}_s$. Note that by enumerating into $\mathcal{A}$ and $\mathcal{B}$ at different positions, we can force that $f_s(x) \neq f_{s+1}(x)$ or $f_s^{-1}(y) \neq f_{s+1}^{-1}(y)$, so $f := \lim_{s\to\infty} f_s$ need not extend any f_s. Nevertheless, since any given position will change at most finitely often, f will be computably approximable by this sequence of partial isomorphisms. That is, f will extend longer and longer initial segments of the partial isomorphisms, so that for a fixed initial segment of f, there is eventually some stage after which all partial isomorphisms extend that initial segment.

An important note: although we aim to explicitly meet all requirements through the construction, we will only implicitly meet some of them. If φ_e is the characteristic function of some set U which is finite or cofinite, then we do not need to meet $\mathbf{R}_{\langle e,j\rangle}$ or $\mathbf{S}_{\langle e,j\rangle}$ explicitly for any j — that is, declaring a witness x or y during the construction that eventually shows the requirement is met. Instead, since $f(U)$ and $f^{-1}(U)$ will be finite or cofinite (hence computable), we will automatically have that $f(U) \not\equiv_T f$ or $f^{-1}(U) \not\equiv_T f$, provided we make f non-computable. We will make $f \equiv_T \emptyset'$, so in particular, f will be non-computable. However, we cannot know which indices e correspond to the finite or cofinite sets, so we must still ensure that they do not stall the construction, even if the actions that they take do not explicitly ever meet their requirements.

Because we are working with partial approximations to computable sets, we set the following notation. At stage s for index e, we let $\sigma_{e,s}$ be the string defined by the longest segment of $\varphi_{e,s}$ that looks like a characteristic function. Thus $|\sigma_{e,s}| \leq s$ by conventions on convergence. Hence if $\varphi_e = \chi_U$, then $\lim_{s\to\infty} \sigma_{e,s} = \chi_U$.

The plan for meeting a single $\mathbf{R}_{\langle e,j\rangle}$ is to choose a witness x and wait for $\Phi_{j,s}^{f(\sigma_{e,s})}(x)\downarrow= f(x)$. If this happens, we place an immediate $<^{\mathcal{B}}$-predecessor to $f(x)$. However, this may have the unfortunate side-effect of also causing the use of the computation $\Phi_{j,s}^{f(\sigma_{e,s})}(x)$ to be damaged, since it may be the case that $f_s(\sigma_{e,s})$ no longer agrees with $f_{s+1}(\sigma_{e,s+1})$. This is because the enumeration of this predecessor value will change the alignment of all values above it in $\mathcal{B}_{s+1}$. Thus, after such an enumeration – which attempts to diagonalize against some computation – we may wish to restore that computation if it was damaged by aligning the values used in the computation so that they are again either in or out of $f(U)$ in the limit.

Notice that we do not have to re-align these values with the exact same values as the ones they were aligned with when the computation first existed. For example, if at stage s the use of a computation contained 5 with $f_s(3) = 5$ and $\varphi_{e,s}(3)\downarrow= 1$, (that is, $5 \in f_s(\sigma_{e,s})$) then if we wish to restore this computation, we only need to find a value n such that $n \in U$ and arrange for $f(n) = 5$, since then $5 \in f(U)$. So, as long as U is both infinite and co-infinite, we can always wait for a sequence of values in the correct order to appear and then arrange for them to be aligned with the use that we wish to restore. We cannot stall the computation waiting for these values, however, since we do not know if φ_e determines an infinite, co-infinite set, so we shall instead wait for such a configuration to appear. If one does not, then we shall show that φ_e does not determine an infinite, co-infinite set and therefore we do not have to satisfy the requirement directly, as noted above.

Additionally, we shall have markers $\{\gamma_i\}_{i\in\omega}$ that will code $\emptyset'$. We shall arrange these markers so that if γ_i eventually comes to rest on x, then $i \in \emptyset'$ if and only if $i \in K_{f(x)}$, where here $\{K_s\}_{s\in\omega}$ is a standard enumeration of $\emptyset'$. To show that $f \geq_T \emptyset'$, we will show that f allows us to additionally compute these final resting places of each γ_i. Since any isomorphism between two copies of $(\omega, <)$ is $\emptyset'$-computable, this is all we need to show that $f \equiv_T \emptyset'$. For each marker, we have the following requirement, which we aim to meet:

$\mathbf{\Gamma}_i$: Each γ_i eventually comes to rest on some value z_i such that $i \in \emptyset'$ if and only if $i \in K_{f(z_i)}$.

To enable f to compute the final resting places of each marker, we shall take certain actions to leave a trace of when each $\mathbf{\Gamma}_i$ is injured. Since a given $\mathbf{\Gamma}_i$ can only be injured by higher priority requirements, we will have these higher priority requirements leave the trace when they act, which allows us to use f to determine a stage where all requirements of priority higher than a given $\mathbf{\Gamma}_i$ have finished acting. By simulating the construction to this stage and then waiting – if we need to – for a stage where $\mathbf{\Gamma}_i$ is (re-)initialized, we can then determine the final position of γ_i, since it will never be injured after this stage.

We arrange the requirements according to the priority order $\mathbf{R}_0 < \mathbf{S}_0 < \mathbf{\Gamma}_0 < \mathbf{R}_1 < \mathbf{S}_1 < \mathbf{\Gamma}_1 < \cdots$ and proceed via a finite injury construction.

At any stage $s+1$, each requirement of index less than s will have a witness value. For the requirement $\mathbf{R}_{\langle e,j\rangle}$, we denote this witness by $x_{\langle e,j\rangle}$, and similar for $\mathbf{S}_{\langle e,j\rangle}$, we write $y_{\langle e,j\rangle}$.

Additionally, during the course of the construction, we might associate (the graph of) a finite characteristic partial function, say g, with a requirement. When we do this, we say that the requirement has *assigned function g*. At certain stages, we may also declare a requirement to be *satisfied*. This means that at such stages the requirement believes that it has no more action to take and will be met. This is reset by injury, however, since the requirement may be forced to abandon its witness by higher priority requirements.

We will enumerate the least value not in the domain of $\mathcal{A}_s$ at the end of stage s, to ensure that $\mathcal{A}$ eventually has domain ω. This also serves a second purpose: any value enumerated during stage s must be at least s. This value will be enumerated at the end of $\mathcal{A}_s$, i.e. we will enumerate it and declare it to be $<^{\mathcal{A}_s}$-larger than any other value in the domain of $\mathcal{A}_s$. This entire procedure is repeated for $\mathcal{B}$.

At each stage s, each requirement may require attention and then possibly receive it. We say that a requirement that is not satisfied requires attention under the following conditions. We only list the $\mathbf{R}_i$ requirements and the $\mathbf{\Gamma}_i$ requirements, as the conditions and actions for the $\mathbf{S}_i$ requirements are similar to the $\mathbf{R}_i$ requirements once the obvious changes have been made.

- $\mathbf{R}_{\langle e,j\rangle}$ *requires attention for diagonalization* if $\Phi_{j,s}^{f_s(\sigma_{e,s})}(x_{\langle e,j\rangle})\downarrow = f_s(x_{\langle e,j\rangle})$ and it has no assigned function.

  If this requirement receives attention for this reason, then enumerate the least value not in the domain of $\mathcal{B}_s$ so that it is just below $f_s(x_{\langle e,j\rangle})$, so that we have that $f_{s+1}(x_{\langle e,j\rangle}) \neq f_s(x_{\langle e,j\rangle})$. Let g denote the characteristic function of the use segment of the computation $\Phi_{j,s}^{f_s(\sigma_{e,s})}(x_{\langle e,j\rangle})$. That is, the domain of g is the set of values in the domain of $\mathcal{B}_s$ used in this computation. Assign g to this requirement.

- $\mathbf{R}_{\langle e,j\rangle}$ *requires attention for restoration* if it has assigned function g and for the elements of the domain of g exceeding $f_s(x_{\langle e,j\rangle})$, say $f_s(x_{\langle e,j\rangle}) <^{\mathcal{B}_s} d_1 <^{\mathcal{B}_s} d_2 <^{\mathcal{B}_s} \cdots <^{\mathcal{B}_s} d_k$, there are elements in the domain of $\mathcal{A}_s$ say, $a_1 <^{\mathcal{A}_s} a_2 <^{\mathcal{A}_s} \cdots <^{\mathcal{A}_s} a_k$ such that $\sigma_{e,s}(a_i)\downarrow = g(d_i)$ and such that $d_1 <^{\mathcal{B}_s} f_s(a_1)$. (This last condition ensures that the d_is and a_is are not already aligned, which is important since we must make an enumeration no matter what for coding a trace of injury, and this enumeration would mis-align them if they were already so.)

If this requirement receives attention for this reason, then enumerate the least value not in the domain of $\mathcal{B}_s$ so that it is just below $f_s(x_{\langle e,j\rangle})$. Next enumerate unused values under each d_i in $<^{\mathcal{B}_s}$-increasing order so that $f_{s+1}(a_i) = d_i$. We may assume that this is always possible by noting that the values $d_1, \ldots, d_k$ were originally $<^{\mathcal{B}_s}$-consecutive, as they were the tail of the use segment for a computation, and we will enforce that enumerations between these values (by a higher priority requirement) would re-initialize this requirement, so the values must still be $<^{\mathcal{B}_s}$-consecutive. Hence with the correct pattern of enumerations, we can align each d_i with its corresponding a_i. Furthermore, the first enumeration just below $f_s(x_{\langle e,j\rangle})$ will not inhibit our ability to do this, because $d_1 <^{\mathcal{B}_s} f_s(a_1)$. Declare that $\mathbf{R}_{\langle e,j\rangle}$ is satisfied.

- $\mathbf{\Gamma}_i$ *requires attention* if $i \in K_s$.

  If this requirement receives attention for this reason, then enumerate the least value not in the domain of $\mathcal{B}_s$ so that it is just below $f_s(z_i)$, where z_i is the value currently marked by γ_i. Declare that $\mathbf{\Gamma}_i$ is satisfied.

Note that if $\mathbf{R}_i$ receives attention at stage s and has witness x, then $f_{s+1}(x) \geq s$ since we enumerate a fresh value in this spot in both cases. Similarly, if $\mathbf{S}_i$ receives attention at stage s and has witness y, then $f_{s+1}^{-1}(y) \geq s$, and if $\mathbf{\Gamma}_i$ receives attention at stage s and γ_i marking z_i, then $f_{s+1}(z_i) \geq s$. In this way, the witness / marked values of each requirement codes the stage where it has caused injury most recently (if at all).

Construction:

Stage s: For the highest priority active requirement that requires attention at stage s, perform the action indicated above and injure all lower priority requirements, de-activating them. For the highest priority requirement that is not active, (re-)initialize it, assigning it a fresh large witness / marked value as needed. These values are chosen larger than the use of any computation seen so far and larger than any values ever used as witnesses or marked values.

Enumerate the least value not in the domain of $\mathcal{A}_s$ as the final, largest element. Proceed similarly for $\mathcal{B}_s$.

Let $\mathcal{A} \coloneqq \bigcup_s \mathcal{A}_s$, $\mathcal{B} \coloneqq \bigcup_s \mathcal{B}_s$ and $f \coloneqq \lim_{s\to\infty} f_s$. This completes the construction.

Note that since witnesses are always chosen larger than any existing witness, and enumerations always occur (for that witness) at most just below it, any given position can only be enumerated into finitely often, provided we show that each requirement receives attention at most finitely often.

<u>Claim:</u> Each requirement receives attention at most finitely often.

<u>Proof of Claim:</u> We proceed by induction on the priority order. Suppose that we have requirement $\mathbf{R}_{\langle e,j\rangle}$ (the case for $\mathbf{S}_{\langle e,j\rangle}$ is similar) and that all higher priority requirements receive attention at most finitely often. So there exists a stage s after which all higher priority requirements never again receive attention, and hence after stage s, $\mathbf{R}_{\langle e,j\rangle}$ will permanently choose a witness, x. We may assume that this stage is also s. If $\mathbf{R}_{\langle e,j\rangle}$ never receives attention after stage s, then we are done. Since $\mathbf{R}_{\langle e,j\rangle}$ has no assigned function when it is re-initialized, the first stage $t \geq s$ where $\mathbf{R}_{\langle e,j\rangle}$ requires attention (and receives it, since it is of highest priority by choice of s) must be for diagonalization, and $\mathbf{R}_{\langle e,j\rangle}$ will be assigned some function g.

As $\mathbf{R}_{\langle e,j\rangle}$ will never be re-initialized by choice of s, if $\mathbf{R}_{\langle e,j\rangle}$ requires attention after stage t, it must be for restoration. Receiving attention for restoration causes $\mathbf{R}_{\langle e,j\rangle}$ to be declared satisfied, and this will never change. So $\mathbf{R}_{\langle e,j\rangle}$ cannot receive attention again.

Now suppose that we have requirement $\mathbf{\Gamma}_i$, and that again all higher priority requirements receive attention at most finitely often, finishing by stage s. If $\mathbf{\Gamma}_i$ never receives attention after stage s, then we are done, and if it does require attention, it must receive it (as it is of highest priority) and will be declared satisfied, and this will never change. So $\mathbf{\Gamma}_i$ cannot receive attention again. ■

<u>Claim:</u> For each infinite, coinfinite, computable set U and index j, there is some x such that $\Phi_j^{f(U)}(x) \neq f(x)$. That is, if φ_e is the characteristic function for U, then requirement $\mathbf{R}_{\langle e,j \rangle}$ is met.

<u>Proof of Claim:</u> Fix a computable set U with $\varphi_e = \chi_U$ and index j. Suppose otherwise, so that $\Phi_j^{f(U)} = f$. By the previous claim, there is some stage s after which $\mathbf{R}_{\langle e,j \rangle}$ never receives attention and has permanent witness x. Choose $t \geq s$ large enough so that $\Phi_{j,t}^{f(U)}(x) = f(x)$, and also large enough so that after stage t, any partial isomorphism $f_{t'}$ extends the initial segment of f given by the use of this computation. That is, if n is in the use of this computation, and $f^{-1}(n) = m$, then m is in the domain of $\sigma_{e,t}$ and $f_{t'}(m) = f(m)$ for all $t' \geq t$. Such a stage exists eventually since φ_e really is a characteristic function so $\sigma_{e,t}$ can be chosen to be arbitrarily long, and f is the limit of the sequence $\{f_s\}_s$.

Since $\mathbf{R}_{\langle e,j \rangle}$ is not re-initialized after stage s, all requirements that are not of lower priority must have stopped enumerating values, and all lower priority requirements only enumerate values above the witness x of $\mathbf{R}_{\langle e,j \rangle}$. Hence at stage t, we have that $f_t(x) = f(x)$, and also by choice of t we have that $\Phi_{j,t}^{f_t(U)}(x)\downarrow = \Phi_j^{f(U)}(x)\downarrow$.

So, at stage t, we have that $\Phi_{j,t}^{f_t(\sigma_{e,t})}(x) = \Phi_j^{f(U)}(x)\downarrow = f(x) = f_t(x)$. Then $\mathbf{R}_{\langle e,j \rangle}$ should receive attention for diagonalization, but this is impossible since we are beyond stage s. So it must be the case that $\mathbf{R}_{\langle e,j \rangle}$ has an assigned function g.

We claim that $\mathbf{R}_{\langle e,j\rangle}$ cannot permanently have a function assigned without eventually requiring attention for restoration. Let the elements of the domain of g exceeding $f(x) = f_t(x)$ be $f(x) <^{\mathcal{B}} d_1 <^{\mathcal{B}} d_2 <^{\mathcal{B}} \cdots <^{\mathcal{B}} d_k$. Since U is infinite and coinfinite, there must be values $a_1 < a_2 < \cdots < a_k$ such that $\chi_U(a_i) = g(d_i)$ for all $i \leq k$. So wait for a stage $t' \geq t$ where $\sigma_{e,t'}(a_i)\downarrow$. Then at stage t', $\mathbf{R}_{\langle e,j\rangle}$ would receive attention for restoration at stage t'. But this is impossible, since $t' \geq t \geq s$.

So it must have been that at stage t, $\mathbf{R}_{\langle e,j\rangle}$ is already declared satisfied. This implies that there was a stage t_1 where $\mathbf{R}_{\langle e,j\rangle}$ received attention for diagonalization and was assigned some function g, then at some later stage t_2 received attention for restoration, and was marked as satisfied, and then was never re-initialized.

At stage t_1, it must be that $\Phi_{j,t_1}^{f_{t_1}(\sigma_{e,t_1})}(x)\downarrow = f_{t_1}(x)$. A value was enumerated into $\mathcal{B}$ so that $f_{t_1+1}(x) \neq f_{t_1}(x)$, and $\mathbf{R}_{\langle e,j\rangle}$ was assigned the function g whose domain is the set of values in $\mathcal{B}_{t_1}$ used in this computation, with $g(a_i) = \sigma_{e,t_1}(a_i) = \varphi_e(a_i) = \chi_U(a_i)$. Since no higher priority requirements receive attention and enumerate values, all the values in $\mathcal{B}_{t_1}$ that are $<^{\mathcal{B}}$-below $f_{t_1}(x)$ are never enumerated below, and all the values in $\mathcal{A}_{t_1}$ that are $<^{\mathcal{A}}$-below x are never enumerated below. So at all future stages $\hat{t} > t_1$, we have that these values are in $\sigma_{e,\hat{t}}$ if and only if they are in σ_{e,t_1}. Hence for these d_i, we have that $d_i \in f_{\hat{t}}(\sigma_{e,\hat{t}})$ if and only if $d_i \in f_{t_1}(\sigma_{e,t_1})$.

At stage t_2, values are enumerated into $\mathcal{B}$ so that $f_{t_2+1}(a_i) = d_i$ where d_i are the values in the domain of g not considered above, and a_i is chosen so that $\chi_U(a_i) = \sigma_{e,t_2}(a_i) = g(d_i)$. As no values are then ever enumerated into $\mathcal{A}$ or $\mathcal{B}$ to destroy this, we have that for any stage $\hat{t} > t_2$, these values d_i have that $d_i \in f_{\hat{t}}(\sigma_{e,\hat{t}})$ if and only if $\sigma_{e,\hat{t}}(a_i) = g(d_i) = 1$, and $g(d_i) = 1$ exactly when $d_i \in f_{t_1}(\sigma_{e,t_1})$.

Hence we must have that for all such stages $\hat{t} > t_2 > t_1$ we have that $\Phi_{j,\hat{t}}^{f_{\hat{t}}(\sigma_{e,\hat{t}})}(x)\downarrow = \Phi_{j,t_1}^{f_{t_1}(\sigma_{e,t_1})}(x)\downarrow = f_{t_1}(x) \neq f_{\hat{t}}(x)$. Therefore this holds in the limit, so $\Phi_j^{f(U)}(x)\downarrow \neq f(x)$, and so the requirement is met. ■

An extremely similar argument shows that the following claim holds:

<u>Claim:</u> For each infinite, coinfinite, computable set U and index j, there is some y such that $\Phi_j^{f^{-1}(U)}(y) \neq f^{-1}(y)$. That is, if φ_e is the characteristic function for U, then requirement $\mathbf{S}_{\langle e,j\rangle}$ is met.

Therefore, as noted above, *all* $\mathbf{R}_i$ requirements and $\mathbf{S}_i$ requirements are met, since the remaining requirements are automatically met once f is non-computable, and this must be the case, for the previous claims could not be true if f were computable.

<u>Claim:</u> Given f, we can compute the final resting position of each γ_i. Furthermore, f can compute $\emptyset'$.

<u>Proof of Claim:</u> To determine the final marked value of some γ_i, notice that it suffices to determine a stage after which no requirement of higher priority than $\mathbf{\Gamma}_i$ ever receives attention. Once we know such a stage, we can run the construction to that stage and then wait for $\mathbf{\Gamma}_i$ to be initialized and mark some value z. Since no higher priority requirement will ever receive attention after this initialization, $\mathbf{\Gamma}_i$ cannot be injured, and must mark z forever after. That is, z is the final resting position of γ_i.

This then allows us to decide if $i \in \emptyset'$: Note that if i ever enters $\emptyset'$, $\mathbf{\Gamma}_i$ will require attention and receive it at some point once it has marked z, since no higher priority requirement ever receives attention after z has been marked. Say this occurs at stage t. In this case, we will enumerate a fresh large value into the domain of $\mathcal{B}$ just below $f_t(z)$, so that $f_t(z) \geq t$ – since values enumerated at stage t must be at least t. Since no enumeration can take place below this point, we must have $f(z) \geq t$. So if i enters $\emptyset'$

by stage t, then $f(z) \geq t$. Hence to decide if $i \in \emptyset'$, compute $s \coloneqq f(z)$ and then determine if $i \in K_s$.

It remains to show that, given f, we can determine a stage after which no higher priority requirement ever receives attention. Proceed by induction on the priority order. Note: We need to include *all* types of requirements in this induction, not just $\mathbf{\Gamma}_j$s. Suppose we have a requirement $\mathbf{Q}$ of some type and using f we can determine a stage s after which no requirement of priority higher than $\mathbf{Q}$ receives attention. Run the construction to stage s and then wait for $\mathbf{Q}$ to be (re-)initialized. We need to determine a stage t after which $\mathbf{Q}$ never receives attention.

If $\mathbf{Q} = \mathbf{R}_j$ for some j, then notice that if such a requirement recevies attention at stage t, then it enumerates a value into the domain of $\mathcal{B}$ such that $f_t(x_j) \geq t$. Since no higher priority requirement can disrupt this, we would have $f(x_j) \geq t$. Hence, we use f to compute $t \coloneqq f(x_j)$. We need to wait for $\mathbf{R}_j$ to be initialized above so that we can determine what its witness x_j is.

Similarly, if $\mathbf{Q} = \mathbf{S}_j$ for some j, then we can compute $t \coloneqq f^{-1}(y_j)$, where y_j is the witness chosen for $\mathbf{S}_j$ when it is (re-)initialized for the final time after stage s.

Finally, if $\mathbf{Q} = \mathbf{\Gamma}_j$ for some j, then again notice that if such a requirement receives attention at stage t, it enumerates a value into the domain of $\mathcal{B}$ such that $f_t(z_j) \geq t$, where z_j is the value marked by $\mathbf{\Gamma}_j$. So again, we use f to compute $t \coloneqq f(z_j)$.

This concludes the induction. We can use f to determine up to what stage to run the construction for the highest priority requirement to stop receiving attention, run the construction until the next requirement is (re-)initialized and then repeat this for each successive requirement under the priority ordering until we can determine when a given $\mathbf{\Gamma}_i$ marks its final value, which, as noted above, allows us to decide if $i \in \emptyset'$ using f once again.

So $f \geq_T \emptyset'$, and hence $f \equiv_T \emptyset'$. $\blacksquare$

Since $\mathcal{A}$, $\mathcal{B}$ and f are as claimed, this completes the proof. $\square$

Note that if we remove the $\mathbf{S}_i$ requirements, then no enumerations occur into $\mathcal{A}$ except for the ones that occur at the end of each stage, which always occur at the end of the current segment $\mathcal{A}_s$, and so $\mathcal{A}$ will be the standard copy of $(\omega, <)$. Hence we also have the following:

Corollary 2.2.

There is a computable copy $\mathcal{B}$ of $(\omega, <)$ such that if $f : \mathcal{N} \to \mathcal{B}$ is the isomorphism between the standard copy of $(\omega, <)$ and $\mathcal{B}$, then $f \equiv_T \emptyset'$ and no computable set U exists such that $f(U) \equiv_T f$.

3 Isomorphisms on copies of $(\omega^2, <)$

This section is devoted to the proof of Theorem 1.4, which we restate here.

Theorem 3.1.

There is a computable copy $\mathcal{A}$ of ω^2 such that if $f : \mathcal{A} \to \mathcal{N}^2$, then for no computable unary relation U on $\mathcal{A}$ do we have $f(U) \equiv_T f$.

Proof:

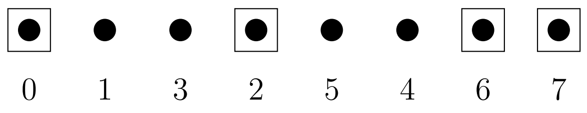
To build $\mathcal{A}$, we will meet the following requirements:

$\mathbf{R}_{\langle e,j \rangle}$: If $\varphi_e = \chi_U$ for some set U, then $(\exists x)[\Phi_j^{f(U)}(x) \neq f(x)]$.

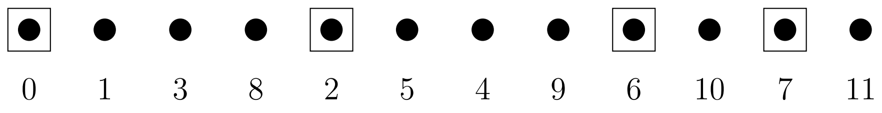
We build $\mathcal{A}$ by stages, enumerating finitely many values into the domain of $\mathcal{A}$. Since at any stage s, $\mathcal{A}_s$ will be isomorphic to some $n < \omega$, we shall guarantee $\mathcal{A} := \cup_s \mathcal{A}_s$ is isomorphic to ω^2 by having an infinite sequence of markers that are promised to be the limit points. Although a marker may occasionally change its value, we will ensure that this occurs at most finitely often (in fact, at most twice) so that eventually each marker settles. We will also ensure that all limit points in $\mathcal{A}$ arise in this way, by ensuring that only marked values have new values enumerated into the domain of $\mathcal{A}$ directly below them infinitely often. This allows us to make strong claims about uses of computations that occur in the limit.

At the end of every stage s, we enumerate new values into the domain of $\mathcal{A}_s$, A_s in a way to eventually forced the marked values to become limit points, provided the markers do not change. We refer to this action as "upkeep", since it maintains the guarantee that the marked values become limit points. For a marked value x, we define the *tail* of x to be the set $\{y \geq^{\mathcal{A}} x \mid (\forall z)[x <^{\mathcal{A}} z \leq^{\mathcal{A}} y \Rightarrow z \text{ is not marked}]\}$. That is, the tail of x is the smallest set containing x and closed under unmarked successors. The upkeep action consists of enumerating a single new value into A_s at the end of every marker's tail.

For example, if at the end of stage s we have $\mathcal{A}_s$ as



where the marked values are boxed, then when we perform this step, we enumerate four new values into the domain of $\mathcal{A}$ – 8, 9, 10 and 11 – so that $\mathcal{A}_s$ becomes



It is clear that if the markers eventually settle, then $\mathcal{A}$ will be isomorphic to ω^2 through this procedure, provided that there are infinitely many markers, and provided we do not disrupt this as mentioned above, by building non-marked limit points, or by a more obviously destructive action, such as building an ω^* somewhere, for example.

In light of this, we will think of each marker as corresponding to a potential ω-chain in $\mathcal{A}$ consisting of the marked value and its eventual infinite tail.

To meet a single requirement $\mathbf{R}_{\langle e,j\rangle}$ in isolation, we employ the following strategy. Choose some witness value, say x_0, and mark it, so that it is associated with some limit point, and choose some other value ℓ to the left of x_0 and mark it as well. For simplicity, we shall suppose for this single requirement that ℓ is associated with the limit point 0 and x_0 is associated with the limit point ω. (We also mark infinitely many values to the right of x_0 so that we build a structure isomorphic to ω^2, but those values are unimportant for now.) If no other action is taken, then the upkeep action detailed above will build $\mathcal{A}$ isomorphic to ω^2 via f, such that $f(\ell) = 0$ and $f(x_0) = \omega$. If the requirement is not met, then $\varphi_e = \chi_U$ and $(\forall x)[\Phi_j^{f(U)}(x) = f(x)]$, and so in particular, we would have $\Phi_j^{f(U)}(x_0) = \omega = f(x_0)$. So, if we see a computation of this form at some stage s, we seek to diagonalize against it, by introducing a new value x_1 to the immediate right of x_1, and moving the marker from x_0 to x_1. This has the effect of making x_1 the value associated with ω, and pushes x_0 into the tail of ℓ, so that x_0 is now associated with some $m \in \omega$.

Unfortunately, this action may destroy the use of the original computation that we were diagonalizing against. Notice, however, that the only affected value is ω. So, if $\varphi_e(x_0) = \varphi_e(x_1)$, then $x_0 \in U$ if and only if $x_1 \in U$. Hence the computation will be restored, as then $\omega \in f(U) \Leftrightarrow x_1 \in U \Leftrightarrow x_0 \in U \Leftrightarrow \omega \in f(U)[s]$. Then we will have $\Phi_j^{f(U)}(x_0) = \omega$, but $f(x_0) = m \neq \omega$, a win.

In the case where $\varphi_e(x_0) \neq \varphi_e(x_1)$, we perform the same trick, but this time using x_1 in place of x_0. So, we wait for a stage where $\Phi_j^{f(U)}(x_1) = \omega = f(x_1)$ and then introduce a new value x_2 to the immediate right of x_1 and move the marker from x_1 to x_2. This pushes x_1 also into the tail of ℓ, and we now win automatically: If $\varphi_e(x_1) = \varphi_e(x_2)$, then the argument above works with x_0 and x_1 replaced by x_1 and x_2, respectively. On the other hand, if $\varphi_e(x_1) \neq \varphi_e(x_2)$, then $\varphi_e(x_0) = \varphi_e(x_2)$, since φ_e would be $\{0,1\}$-valued, and so this restores the original computation we diagonalized against when introducing x_1, so $\Phi_j^{f(U)}(x_0) = \omega$, but $f(x_0) = m \neq \omega$.

In this way, through at most two actions, we can always diagonalize against a computation, or wait forever for such a computation, which is also a win. Notice that in meeting this single requirement, we only needed to consider two marked values, ℓ and one of x_0, x_1 or x_2. So, we can satisfy a single requirement in a $\omega + \omega$ inside ω^2.

So we group the limit points of ω^2 into consecutive pairs, as $(0, \omega), (\omega \cdot 2, \omega \cdot 3), \ldots$ and use these pairs as locations to satisfy each requirement. When a requirement is injured, it abandons its associated pair and is assigned a fresh large pair. Of course, this may require enumerating new values at the end of $\mathcal{A}_s$ and marking them, so as to "create" a new pair of potential limit points. Since each pair of marked values and their tails will eventually correspond to an $\omega \cdot 2$ inside $\mathcal{A}$, we refer to the pair of marked values itself as an $\omega \cdot 2$, and so we may speak of "creating a fresh large $\omega \cdot 2$ at stage s", for instance, even though this only truly refers to enumerating two values at the end of $\mathcal{A}_s$ and marking them.

As we can see above, each requirement will also have a state: it is either waiting for a computation involving x_0, waiting for a computation involving x_1 or has won. When computations are (re-)initialized, they will always be waiting for x_0.

For a given requirement $\mathbf{R}_{\langle e,j \rangle}$, we will say that $\mathbf{R}_{\langle e,j \rangle}$ *requires attention at stage s* under the following conditions:

- If $\mathbf{R}_{\langle e,j \rangle}$ is waiting for x_0, then its $\omega \cdot 2$ has the form

$$\begin{array}{cccc} \boxed{\bullet} & \cdots & \boxed{\bullet} & \cdots \\ \ell & & x_0 & . \end{array}$$

The requirement requires attention if $\Phi_j^{f(U)}(x_0)[s]\downarrow = f_s(x_0)$, and if it receives attention, then enumerate a new value, x_1, directly to the right of x_0, and move the marker from x_0 to x_1. The $\omega \cdot 2$ for the requirement will then have the form

$$\begin{array}{ccccc} \boxed{\bullet} & \bullet\bullet\bullet & \bullet & \boxed{\bullet} & \bullet\bullet\bullet \\ \ell & & x_0 & x_1 & \end{array},$$

and it will be waiting for x_1.

- If $\mathbf{R}_{\langle e,j\rangle}$ is waiting for x_1, then its $\omega \cdot 2$ has the form

  $$\begin{array}{cccccc} \boxed{\bullet} & \bullet\bullet\bullet & \bullet & \bullet\bullet & \boxed{\bullet} & \bullet\bullet\bullet\bullet\bullet \\ \ell & & x_0 & & x_1 & \end{array}.$$

  The requirement requires attention if either $\varphi_{e,s}(x_0)\downarrow = \varphi_{e,s}(x_1)\downarrow$, or if $\varphi_{e,s}(x_0)\downarrow \neq \varphi_{e,s}(x_1)\downarrow$ and $\Phi_j^{f(U)}(x_1)[s]\downarrow = f_s(x_1)$. In the former case, no further action is needed, and the requirement is met (unless it is later injured).

  In the latter case, if the requirement receives attention then enumerate a new value, x_2, directly to the right of x_1, and move the marker from x_1 to x_2. The $\omega \cdot 2$ for the requirement will then have the form

  $$\begin{array}{cccccccc} \boxed{\bullet} & \bullet\bullet\bullet & \bullet & \bullet\bullet & \bullet & \boxed{\bullet} & \bullet\bullet\bullet\bullet\bullet \\ \ell & & x_0 & & x_1 & x_2 & \end{array}$$

  and the requirement will be met (again, unless it is later injured).

Here, when we write something like $\Phi_j^{f(U)}(x_0)[s]$, we think of this computation as converging if each value y in the use is verifiably in $f_s(U)$ or verifiably not in $f_s(U)$. That is, if x is such that $f_s(x) = y$, then $\varphi_{e,s}(x)\downarrow \in \{0, 1\}$.

We arrange the requirements according to the priority ordering $\mathbf{R}_0 > \mathbf{R}_1 > \cdots$, and proceed by a finite injury argument.

<u>Construction</u>:

Stage s: For the highest priority active $\mathbf{R}_i$ that requires attention at stage s, perform the action indicated above and injure all lower priority requirements, de-activating them. Perform the upkeep as mentioned above, where new values are enumerated into A_s at the end of the tail associated to each marker. Finally, for the highest priority requirement that is not active, assign to it a fresh large $\omega \cdot 2$, beyond the use of any computation seen so far in the construction.

Verification:

Claim: Every requirement receives attention at most finitely often and is met.

Proof of Claim: It is clear from the construction that each requirement can only be injured by higher priority requirements, and each requirement can receive attention at most twice if it is never injured. Because of this, it is clear that each requirement receives attention at most finitely often. Indeed, $\mathbf{R}_i$ receives attention at most $2^{i+2} - 2$ times.

Consider the requirement $\mathbf{R}_{\langle e,j \rangle}$. By induction on the priority ordering, we may assume that there is a stage after which no requirement of higher priority than $\mathbf{R}_{\langle e,j \rangle}$ receives attention, and a least stage s after that where $\mathbf{R}_{\langle e,j \rangle}$ was (re-)initialized for the final time, and permanently associated with an $\omega \cdot 2$, say with limit points $\omega \cdot n$ and $\omega \cdot (n+1)$.

Suppose for a contradiction that $\varphi_e = \chi_U$ for some set U (i.e. φ_e is total and $\{0,1\}$-valued) and for all x we have $\Phi_j^{f(U)}(x)\downarrow = f(x)$.

First, $\mathbf{R}_{\langle e,j \rangle}$ cannot permanently be waiting for x_0. To see why, note that if no action takes place, then $f_t(x_0) = \omega \cdot (n+1)$ for all $t > s$, and so $f(x_0) = \omega \cdot (n+1)$. Since $\Phi_j^{f(U)}(x_0)\downarrow = f(x_0)$ by assumption, there must be some stage $s' > s$ where $\Phi_j^{f(U)}(x_0)[s']\downarrow = \omega \cdot (n+1) = f_{s'}(x_0)$. But then at stage s', $\mathbf{R}_{\langle e,j \rangle}$ would require attention and receive it, since it is of highest priority. So at some stage s', $\mathbf{R}_{\langle e,j \rangle}$ must receive attention and wait for x_1.

Recall that while waiting for x_1, requirements can require attention for two reasons, which we refer to as condition (1) and condition (2). We will first show that if a requirement that is waiting for x_1 never requires attention via condition (2), then it must require attention via condition (1). Since $\mathbf{R}_{\langle e,j\rangle}$ will be of highest priority, if it requires attention for either of these reasons, it will receive it. We will therefore show that in either case, the action taken contradicts the supposition above, which will complete the proof of the claim.

So, suppose the requirement never requires attention via condition (2). By similar reasoning as in the previous case, we can find some stage $s'' > s' > s$ where $\Phi_j^{f(U)}(x_1)[s'']\downarrow = \omega \cdot (n+1) = f_{s''}(x_1)$. So at no stage $t \geq s''$ can we have $\varphi_{e,t}(x_0)\downarrow \neq \varphi_{e,t}(x_1)\downarrow$. Since φ_e is total by assumption, there must be some stage $t' \geq s''$ where $\varphi_{e,t'}(x_0)\downarrow = \varphi_{e,t'}(x_1)\downarrow$. But this is exactly condition (1), so $\mathbf{R}_{\langle e,j\rangle}$ would require attention via condition (1) at stage t'.

Suppose first that $\mathbf{R}_{\langle e,j\rangle}$ receives attention via condition (1). Note that at stage s', it must have been the case that $\Phi_j^{f(U)}(x_0)[s']\downarrow = \omega \cdot (n+1)$. Since all higher priority requirements do not act after $s < s''$ and lower priority requirements were injured at stage s' and then later re-initialized beyond the use of this computation, we know that $f(U)[s'] = f(U)[s'']$ on the use of this computation, except for possibly at x_0 since $f_{s'}(x_0) = \omega \cdot (n+1)$ and $f_{s'+1}(x_0) = \omega \cdot n + m$ for some $m \in \omega$.

Hence, at stage t', we have $f_{t'}(x_1) = \omega \cdot (n+1)$ and $\varphi_{e,t'}(x_0)\downarrow = \varphi_{e,t'}(x_1)\downarrow$, so $x_0 \in f(U)[s']$ if and only if $x_1 \in f(U)[t']$, and so $\Phi_j^{f(U)}(x_0)[t']\downarrow = \omega \cdot (n+1)$. Now note that x_0 is never again marked after we stop waiting for x_0, so $f(x_0) = \omega \cdot n + m \neq \omega \cdot (n+1)$, and so $\Phi_j^{f(U)}(x_0) \neq f(x_0)$, a contradiction. Hence $\mathbf{R}_{\langle e,j\rangle}$ is met in this case.

Suppose second that $\mathbf{R}_{\langle e,j\rangle}$ receives attention via condition (2). Hence $\varphi_e(x_0)\downarrow \neq \varphi_e(x_1)\downarrow$, and so $x_0 \in U$ if and only if $x_1 \notin U$. Wait for some stage where $\varphi_e(x_2)$, which exists since φ_e is total. If $\varphi_e(x_2)\downarrow = \varphi_e(x_1)\downarrow$, then similar reasoning as before gives that $\Phi_j^{f(U)}(x_1)\downarrow = \omega \cdot (n+1) \neq \omega \cdot n + m' = f(x_1)$, where $m' \in \omega$. If $\varphi_e(x_2)\downarrow \neq \varphi_e(x_1)\downarrow$, then since φ_e is $\{0,1\}$-valued by assumption, it must be the case that $\varphi_e(x_2)\downarrow = \varphi_e(x_0)\downarrow$, and so $\Phi_j^{f(U)}(x_0)\downarrow = \omega \cdot (n+1) \neq \omega \cdot n + m = f(x_0)$. So again, we have a contradiction and $\mathbf{R}_{\langle e,j\rangle}$ is met.

Thus $\mathbf{R}_{\langle e,j\rangle}$ is met in all cases, as desired. ■

Note that in any given $\omega \cdot 2$, enumerations only take place twice when the associated requirement (if there is any) receives attention and also during the upkeep at the end of each stage. The latter clearly preserves the ordinal structure, since it occurs at the end of the two tails, and the former occurs only twice. The marker moves at most twice. Hence it is clear that each pair of marked values really does build an $\omega \cdot 2$ and so $\mathcal{A} \cong \omega^2$, since infinitely many $\omega \cdot 2$ will be created during the construction, as a new $\omega \cdot 2$ is created each time a requirement is newly initialized. □

Note that in Theorem 1.4, the isomorphism constructed has degree $\mathbf{0}'$, which is the degree of categoricity of the structure in question, $(\omega, <)$. However, the isomorphism f we construct above is clearly limit computable, so it does not have degree $\mathbf{0}''$, the degree of categoricity of $(\omega^2, <)$. So in this sense, Theorem 1.6 is a weak analogue of Theorem 1.4. However, it is possible to modify the construction above to combine it with the method of the theorem for the $(\omega, <)$ case to code $\emptyset''$ into the isomorphism. We proceed similarly to the previous theorem, with $\mathbf{R}$-type requirements that serve to force the isomorphism to be of higher degree than $f(U)$ for any computable U. In addition, we also code $\emptyset''$ into f by using the marking strategy involving $\mathbf{\Gamma}$-type requirements. However, these requirements are now more complicated than requiring a single marked value, since they must code $\emptyset''$ rather than $\emptyset'$. To do so, we assign each $\mathbf{\Gamma}$ an entire $\omega \cdot 2$ block and arrange for it to code whether, say i, is in Fin, by modifying the elements that correspond to limit points in this block. However, since this possibly causes infinite action to be taken – if i is not in Fin – we must arrange the requirements on a tree and use an infinite injury priority argument. The exact details are technical, and deferred to Deveau's thesis.

4 Conclusion

We close by mentioning related questions of interest.

Our analysis in this paper shows that between computable copies of a rigid structure that realize the strong degree of categoricity of the structure, there need not be a computable set whose image codes the degree of the isomorphism, even if we assume one of the copies is decidable. However, we were working with the structures $(\omega, <)$ and $(\omega^2, <)$, which each *do* have *some* copies between which a unary relation coding the degree of the isomorphism exists.

Question 4.1.
Let $\mathcal{A}$ be rigid and computable, with strong degree of categoricity $\mathbf{d}$. Are there computable copies $\mathcal{B}, \mathcal{C}$ of $\mathcal{A}$ with isomorphism $f : \mathcal{B} \to \mathcal{C}$ and a computable U such that $f(U)$ is of Turing degree $\mathbf{d}$?

If the answer to Question 4.1 is positive, can the assumption about rigidity be dropped?

Is there a rigid structure such that two non-computably isomorphic computable copies exist and are isomorphic via f so that for any computable set U, $f(U)$ is not only strictly below f in degree, but is in fact always computable? Does there exist a structure where this is true for all pairs of computable copies of that structure?

References

[BKY] Nikolay A. Bazhenov, Iskander Sh. Kalimullin, and Mars M. Yamaleev. Degrees of categoricity and spectral dimension.

[CFS13] Barbara F. Csima, Johanna N. Y. Franklin, and Richard A. Shore. Degrees of categoricity and the hyperarithmetic hierarchy. *Notre Dame J. Form. Log.*, 54(2):215–231, 2013.

[CS] Barbara Csima and Jonathan Stephenson. Finite computable dimension and degrees of categoricity.

[FKM10] Ekaterina B. Fokina, Iskander Kalimullin, and Russell Miller. Degrees of categoricity of computable structures. *Arch. Math. Logic*, 49(1):51–67, 2010.

[Soa16] Robert I. Soare. *Turing computability.* Theory and Applications of Computability. Springer-Verlag, Berlin, 2016. Theory and applications.

B. Csima was partially supported by Canadian NSERC Discovery Grant 312501. M. Deveau was partially supported by Canadian NSERC Postgraduate Scholarship PGSD1-234567-2017. J. Stephenson is now a Visiting Assistant Professor at Valparaiso University, Indiana.