Computable Categoricity of Graphs with Finite Components

Barbara F. Csima[1] *, Bakhadyr Khoussainov[2] **, and Jiamou Liu[3] ***

[1] Department of Pure Mathematics
University of Waterloo
Waterloo, ON, Canada N2L 3G1
www.math.uwaterloo.ca/~csima
csima@math.uwaterloo.ca
[2] Department of Computer Science
University of Auckland
Auckland Private Bag 92019 Auckland New Zealand
www.cs.auckland.ac.nz/~bmk
bmk@cs.auckland.ac.nz
[3] Department of Computer Science
University of Auckland
Auckland Private Bag 92019 Auckland New Zealand
www.cs.auckland.ac.nz/~jliu036
jliu036@ec.auckland.ac.nz

Abstract. A computable graph is computably categorical if any two computable presentations of the graph are computably isomorphic. In this paper we investigate the class of computably categorical graphs. We restrict ourselves to strongly locally finite graphs; these are the graphs all of whose components are finite. We present a necessary and sufficient condition for certain classes of strongly locally finite graphs to be computably categorical. We prove that if there exists an infinite Δ^0_2-set of components that can be properly embedded into infinitely many components of the graph then the graph is not computably categorical. We also show that the Δ^0_2-bound found is sharp. This is proved by a construction (that we outline in this paper) that builds a strongly locally finite computably categorical graph with an infinite chain of properly embedded components. There are also several examples.

1 Introduction

In this paper we are interested in computable graphs. A **computable graph** $\mathcal{G}$ is a pair (V, E) where the set V of vertices and the set E of edges are both computable sets. All our graphs are undirected and infinite. If $\mathcal{G}$ is a computable

* Partially supported by Canadian NSERC Discovery Grant 312501.
** B. Khoussainov has partially been supported by Marsden Fund of Royal New Zealand Society
*** J. Liu is supported by NZIDRS of Education New Zealand.

graph isomorphic to a graph $\mathcal{G}'$ then $\mathcal{G}$ is called a **computable presentation of** $\mathcal{G}'$ and $\mathcal{G}'$ is called **computably presentable**. For a computable graph $\mathcal{G}$ we can always assume that the set of vertices of $\mathcal{G}$ is ω, the set of natural numbers.

The study of computable structures goes back to the late 1950s and finds its roots in the work of A. Malcev [15] and M. Rabin [16]. Later the theory has been developed by Yu. Ershov and A. Nerode and their colleagues (e.g. [3]). For the current state of the area see, for example, the book by Ershov and Goncharov [7], the Handbooks on computable models and algebra [5] [6]. See also [11].

One of the central themes in the theory of computable structures is concerned with computable isomorphisms. We say that two computable graphs G_1, G_2 have the same **computable isomorphism type** if G_1 and G_2 are computably isomorphic.

Definition 1. *The number of computable isomorphism types of graph $\mathcal{G}$, denoted by $dim(\mathcal{G})$, is called the* **computable dimension** *of $\mathcal{G}$. If the computable dimension of $\mathcal{G}$ equals 1 then the graph $\mathcal{G}$ is called* **computably categorical**.

For example the graph (ω, E) where $E = \{\{i, i+1\} \mid i \in \omega\}$ is computably categorical. The graph consisting of ω many copies of (ω, E) is not computable categorical; in fact, it has computable dimension ω. In general, providing examples of computably categorical graphs or graphs of computable dimension ω is easy. S. S. Goncharov in [9] was the first to provide examples of graphs of computable dimension n, where $n > 1$. In this paper we will be interested in the study of computably categorical graphs in a specific class of graphs called strongly locally finite graphs.

The study of computably categorical structures constitutes one of the major topics in the study of computable isomorphisms. Here the goal is to provide a characterization of computably categorical structures within specific classes of structures. This has been done for Boolean algebras [4], linearly ordered sets [17], trees [14], Abelian groups [8], ordered Abelian groups [12], etc. Hence, this paper fits the general program devoted to the study of computable isomorphisms.

Let S be a sequence $\mathcal{G}_0, \mathcal{G}_1, \ldots$ of pairwise disjoint finite graphs. Define the new graph $\mathcal{G}_S$ as the disjoint union of these graphs. More formally, the set of vertices of $\mathcal{G}_S$ is $\bigcup_{i \in \omega} V_i$ and the set of edges is $\bigcup_{i \in \omega} E_i$.

Let $\mathcal{G}$ be a graph. We say that vertices v and w are **connected** if there is a path from v to w. In this case we also say that w is **reachable** from v. The **component** of $\mathcal{G}$ is a maximal subset of $\mathcal{G}$ in which any two vertices are connected. The component containing a vertex v is denoted by $C(v)$.

We say that $\mathcal{G}$ is **strongly locally finite** if every component of $\mathcal{G}$ forms a finite graph. It is not hard to see that $\mathcal{G}$ is strongly locally finite if and only if $\mathcal{G}$ is $\mathcal{G}_S$ for some sequence S of pairwise disjoint finite graphs. The following proposition gives a full description of computable dimensions for strongly locally finite graphs:

Proposition 1. *The computable dimension of any strongly locally finite graph is either 1 or ω. In particular, no strongly locally finite graph has a finite computable dimension n, where $n > 1$.*

Proof. We invoke the following well-known result of Goncharov [10]. If any two computable presentations of a structure $\mathcal{A}$ are isomorphic via a Δ^0_2-function then the computable dimension of $\mathcal{A}$ is either 1 or ω. Now, if G is strongly locally finite then any two computable presentations of G are isomorphic via a Δ^0_2-function. □

By this proposition, it makes perfect sense to work towards a characterization of computably categorical strongly locally finite graphs. This is the subject of this paper.

Here is an outline of the rest of the paper. In the next section we provide a necessary and sufficient condition for certain types of strongly locally finite graphs to be computably categorical. In Section 3 we prove that if there is a infinite Δ^0_2-set X of vertices in graph $\mathcal{G}$ such that $C(v)$, the component containing v, embeds into infinitely many components of $\mathcal{G}$ for all $v \in X$, then $\mathcal{G}$ is not computably categorical. In Section 4 we give several examples of computably categorical and non-computably categorical strongly locally finite graphs. Finally, in the last section we outline a construction of a computably categorical strongly locally finite graph that possesses a infinite chain of embedded components. In particular this example shows that the existence of infinitely many components each of which can be embedded into infinitely many components does not guarantee computable categoricity. The example also shows that the Δ^0_2-complexity used in the proof of the main result in Section 3 is sharp.

Finally, *all* our graphs considered in this paper are strongly locally finite.

2 Computable Categoricity and The Size Function

Let $\mathcal{G}$ be a computable graph. Define the **size function** $size_{\mathcal{G}} : V \to \omega$ by $size_{\mathcal{G}}(v) = |C(v)|$, where $C(v)$ is the component of vertex v.

Lemma 1. *Let $\mathcal{G}_1, \mathcal{G}_2$ be computable presentations of $\mathcal{G}$ such that $size_{\mathcal{G}_1}$, $size_{\mathcal{G}_2}$ are computable. Then $\mathcal{G}_1$ and $\mathcal{G}_2$ are computably isomorphic.*

Proof. For $i \in \{1, 2\}$, we can effectively reveal $C(v)$ for any vertex v in $\mathcal{G}_i$ by searching for the $size_{\mathcal{G}_i}(v)$ vertices that are connected to v. To construct a computable isomorphism between $\mathcal{G}_1$ and $\mathcal{G}_2$, map each v to the corresponding vertex v' in $\mathcal{G}_2$ such that $C(v) \cong C(v')$. In the construction, use the back and forth method of building the isomorphism. □

The lemma implies that $\mathcal{G}$ is computably categorical if the size function is computable for all computable presentations of $\mathcal{G}$.

Proposition 2. *Suppose $size_{\mathcal{G}}$ is a computable function. The graph $\mathcal{G}$ is computably categorical if and only if the size function is computable for all computable presentations of $\mathcal{G}$.*

Proof. One direction is proved by Lemma 1. The other direction is straightforward since from $\mathcal{G}$ to any computable presentation $\mathcal{G}'$ of $\mathcal{G}$ there is a computable isomorphism h. Then $size_{\mathcal{G}'}(v) = size_{\mathcal{G}}(h(v))$. □

In the rest of this section we suppose that $size_{\mathcal{G}}$ is computable. For any vertex $v \in V$, one effectively reveals the component of v by using $size_{\mathcal{G}}(v)$. So, we effectively list (without repetition) $C_0, C_1, \ldots$ all components of G.

Given two finite graphs $\mathcal{H}_1 = (V_1, E_1)$ and $\mathcal{H}_2 = (V_2, E_2)$, we say $\mathcal{H}_1$ **properly embeds** into $\mathcal{H}_2$ if V_1 can be mapped injectively to a *proper* subset of V_2 that preserves the edge relation. We denote it by $\mathcal{H}_1 \prec \mathcal{H}_2$.

Lemma 2. *If there are infinitely many i such that $\{j \mid C_i \prec C_j\}$ is an infinite set, then $\mathcal{G}$ is not computably categorical.*

Proof. Our goal is to build a graph $\mathcal{G}' = (\omega, E')$ such that $\mathcal{G}' \cong \mathcal{G}$ but $\mathcal{G}'$ is not computably isomorphic to $\mathcal{G}$. Let $\Phi_0, \Phi_1, \ldots$ be a standard enumeration of all partial computable functions from ω to ω. We construct a graph $\mathcal{G}'$ that satisfies the following requirements:

$$P_e : \Phi_e \text{ is not an isomorphism from } G \text{ to } G'$$

The requirement P_e has a higher **priority** than P_t if $t > e$. We construct $\mathcal{G}'$ by stages. At stage s we construct a finite graph $\mathcal{G}'_s$ so that $\mathcal{G}'_s$ is isomorphic to $\mathcal{G}$ restricted to $C_0 \cup \ldots \cup C_{s-1}$, $\mathcal{G}'_s \subset \mathcal{G}'_{s+1}$ for all s, and f_s is the isomorphism constructed at stage s. Our desired graph will be $\mathcal{G}' = \cup_s \mathcal{G}'_s$. Set $\mathcal{G}'_0$ to be the empty graph. Set f_0 to be undefined.

At stage $s+1$, consider $\mathcal{G}_s$ obtained by adding C_s to $\mathcal{G}_{s-1}$. Let $C'_0, \ldots, C'_{s-1}$ be all components in $\mathcal{G}'_{s-1}$ such that each C'_i is isomorphic to C_i via the partial function f_s for $i < s$. Find minimal $e \leq s+1$ such that for some $i < s$ we have:

1. Φ_e has not been processed and $\Phi_{e,s+1}$ is defined on C_i.
2. $\Phi_{e,s+1}$ is a partial isomorphism.
3. The component $C'_j = \Phi_e(C_i)$ is free for Φ_e, and $C_i \prec C_s$.

If such e does not exist then go on to the next stage. Otherwise, act as follows: (1) Extend C'_j to a component, denoted by C'_s, such that $C'_s \cong C_s$; (2) Build a new copy C'_j isomorphic to C_j; (3) Redefine f_s by mapping C_j to C'_j and C_s to C'_s. Declare C'_s not free for all Φ_t with $t > e$, and declare Φ_e processed. This completes the construction for $\mathcal{G}'_{s+1}$.

The correctness of the construction is now a standard proof. The proof is based on the following two observations. First of all, one inductively shows that each requirement P_e is satisfied. Secondly, one proves that the function $f(v) = \lim_s f_s(v)$ establishes an isomorphism (which is necessarily a Δ^0_2-set). □

For a computable graph $\mathcal{G}$ with a computable size function, let $C_0, C_1, \ldots$ be an effective list of all components of $\mathcal{G}$. Define the **proper extension function** $ext_{\mathcal{G}} : \omega \to \omega$ by $ext_{\mathcal{G}}(i) = |\{j \mid C_i \prec C_j\}|$.

Lemma 3. *Suppose there are finitely many i such that the set $\{j \mid C_i \prec C_j\}$ is infinite. If $ext_{\mathcal{G}}$ is not computable then $\mathcal{G}$ is not computable categorical.*

Proof. The construction of $\mathcal{G}'$ that is isomorphic but not computably isomorphic to $\mathcal{G}$ is very similar to the construction for the previous lemma. The only difference is that we start with $\mathcal{G}_0$ as consisting of all (finitely many) components in $\mathcal{G}$ that embed into infinitely many components. Therefore in this construction let $C_0, C_1, \ldots$ list all *other* components in $\mathcal{G}$. The construction of the previous lemma is then repeated.

Suppose P_e is the requirement with the highest priority that is not satisfied. Let s be the stage when all requirements with higher priorities are satisfied. Since Φ_e is an isomorphism, we can compute the the function $ext_{\mathcal{G}}$ as follows. Consider C_i for which $\Phi_e(C_i)$ is free for Φ_e. Note that there are only finitely many C_i that are not free for Φ_e. Let t be the stage $> s$ such that $\Phi_{e,t}$ is defined on C_i. From this stage on C_i can not be properly embedded into C_k for all $k > t$. Hence the number of proper extensions of C_i in $\mathcal{G}_t$ can be computed effectively. □

We can now prove the following characterization theorem:

Theorem 1. *Let $\mathcal{G}$ be a graph such that $size_{\mathcal{G}}$ is a computable function. Then the following are equivalent:*

1. *$\mathcal{G}$ is computably categorical.*
2. *The size function is computable in all computable presentations of $\mathcal{G}$.*
3. *There are finitely many i such that the set $\{j \mid C_i \prec C_j\}$ is infinite and the function $ext_{\mathcal{G}}$ is computable.*

Proof. The equivalence of (1) and (2) follows from Proposition 2. The direction (1) to (3) follows from the lemmas above. We prove the implication (3) $\rightarrow$ (1). So, let $\mathcal{G}'$ be a computable presentation of $\mathcal{G}$. Take all components C_i such that $\{j \mid C_i \prec C_j\}$ is infinite. There are only finitely many such C_i; non-uniformly map them to isomorphic components in $\mathcal{G}'$.

Take C_i such that $\{j \mid C_i \prec C_j\}$ is finite. Since $ext_{\mathcal{G}}$ is computable, we can list all components $X_1, \ldots, X_p$ in $\mathcal{G}$ that properly extend C_i. In $\mathcal{G}'$ find components $Y, Y_1, \ldots, Y_p$ such that Y is isomorphic to C_i and each Y_i is isomorphic to X_i. Map C_i isomorphically to Y. It is not hard to show, using the definition of the function $ext_{\mathcal{G}}$ and induction on the number of proper extensions of C_i, that Y is a component of $\mathcal{G}'$ isomorphic to C_i. □

3 A Sufficient Condition for Not Computably Categorical

In this section we do *not* assume computability of the size function $size_{\mathcal{G}}$. The theorem below gives us a version of Lemma 2 in this case.

Theorem 2. *Let $\mathcal{G}$ be a strongly locally finite graph on which the reachability relation is computable. If there exists an infinite Δ^0_2 set of vertices X such that $(\forall x \in X)(\exists^\infty v)[C(x) \prec C(v)]$, then $\mathcal{G}$ is not computably categorical.*

Proof. For each $s \in \omega$, let $\mathcal{G}_s$ be the restriction of the graph of $\mathcal{G}$ to vertices among $\{0, ..., s\}$. Since $\mathcal{G}$ is computable, we can uniformly compute $\mathcal{G}_s$. For each $v \in \{0, ..., s\}$, let $C_s(v)$ denote the connected component of v in $\mathcal{G}_s$. Since the reachability relation on $\mathcal{G}$ is computable, we may assume without loss of generality that if $C_{\max(v,w)}(v) \neq C_{\max(v,w)}(w)$, then $C_s(v) \neq C_s(w)$ for all s. That is, when a new vertex is added to the graph of $\mathcal{G}$ it is immediately decided whether it is in the same component as any previously present vertices.

We will build a computable graph $\mathcal{H} \cong \mathcal{G}$ such that we meet for each $e \in \omega$ the requirement:

$$R_e : \Phi_e \text{ is not an isomorphism from } \mathcal{H} \text{ to } \mathcal{G}$$

We will construct $\mathcal{H}$ by stages. At each stage s we will have a function $h_s : \mathcal{G}_s \cong \mathcal{H}_s$ and we will ensure that $h = \lim_s h_s$ exists.

If we declare that $h_s(v) = w$, then we will define h_s such that $h_s : C_s(v) \cong C_s(w)$. If at a later stage t the component of v in $\mathcal{G}$ grows ($C_s(v) \subsetneq C_t(v)$), and we still have $h_t(v) = h_s(v)$, then we will add a new vertex to $\mathcal{H}_t$ and define h_t to extend h_s so that $h_t : C_t(v) \cong C_t(w)$.

To meet requirement R_e we will find a vertex v_e such that either $\Phi_e(v_e) \uparrow$ or $C(v_e) \prec C(\Phi_e(v_e))$.

Let $\{X_s\}_{s \in \omega}$ be a Δ^0_2 approximation of X. For $n, s \in \omega$, let $x_{n,s} = \mu x[x \in X_s \wedge (\forall m < n)[x \notin C_s(x_{m,s})]]$. Note that since X is Δ^0_2 and since each component of $\mathcal{G}$ is finite, $x_n = \lim_s x_{n,s}$ exists for all n.

At each stage s of the construction, we will have $v_{e,s} = x_{n,s}$ for some $n \geq e$. We will ensure that for each $e \in \omega$, $v_e = \lim_s v_{e,s}$ exists and provides the witness for meeting requirement R_e.

The basic idea for meeting a single requirement R_0 is as follows. We let $v_{0,s} = x_{0,s}$ at every stage s. If we ever see that $\Phi_{0,s}(v_{0,s}) \downarrow$, and if Φ_0 appears to be an isomorphism in the sense that the component of $v_{0,s}$ in $\mathcal{G}_s$ is isomorphic to the component of $\Phi_0(v_{0,s})$ in $\mathcal{H}_s$, then we begin to search for a new component to appear in $\mathcal{G}$ that properly extends the component of $v_{0,s}$. If $v_{0,s} \in X$, then we will find such a component. So, at the same time as searching for the component, we also run the approximation of X to see if $v_{0,t} \neq v_{0,s}$ at some later stage t. If we first find out that v_0 changes, then we continue to wait for Φ_0 to converge on this new v_0. If we are provided with a new component extending that of $v_{0,s}$ then we re-define our map h and extend the graph $\mathcal{H}$ so that the component of $\Phi_0(v_{0,s})$ in $\mathcal{H}$ is now isomorphic to the new large component, and we include a new component in $\mathcal{H}$ that is isomorphic to the component of $v_{0,s}$ in $\mathcal{G}$. Thus at the end of stage $s+1$, we will have $C_s(v_{0,s}) \prec C_s(\Phi_e(v_{0,s}))$. This will have us meet requirement R_0 unless the component of $v_{0,s}$ in $\mathcal{G}$ grows at some later stage. If this happens, we again search for a proper extension of the component of v_0 in $\mathcal{G}$ to complete the diagonalization. Note that after a certain stage, $v_{0,s}$ will never change, and will always be a member of X. Since the component of v_0 in $\mathcal{G}$ is finite, it can grow only finitely often. If after the component of v_0 in $\mathcal{G}$ has fully appeared we see that $\Phi_0(v_0) \downarrow$, then we will at that point succeed in meeting requirement R_0.

The only extra complication for multiple requirements is that we want to ensure that $h : \mathcal{H} \cong \mathcal{G}$, so we must make sure that if some $w \in \text{range}(h_s)$, then $h^{-1}(w)$ exists. That is, we only re-define $h_s^{-1}(w)$ finitely often. This is where we will use the v_e instead of just x_e as witnesses. If we find that $\Phi_e(v_{e,s}) \downarrow$, but is mapped to some component where we have already redefined h for the sake of higher priority requirements, then instead of proceeding with the diagonalization, we will change v_e to be the next member of X (i.e., if $v_{e,s} = x_{n,s}$, we would let $v_{e,s+1} = x_{n+1,s+1}$). Since each requirement only causes h to be re-defined finitely often, v_e will only be re-defined finitely often for this reason. If we notice that we were wrong about our guess for x_n (i.e., $x_{n,s} \neq x_{n,s+1}$), then we will drop back down all the $v_{e,s} \geq x_{n,s}$ to be as small as possible.

We now give the formal construction.

We may assume without loss of generality that if $C_s(v) \neq C_s(v')$, and if $\Phi_e(v) \downarrow$ and $\Phi_e(v') \downarrow$ then $C_s(\Phi_e(v)) \neq C_s(\Phi_e(v'))$. This is because since $\mathcal{G}$ has the computable reachability relation, $C_s(v) \neq C_s(v') \Rightarrow C(v) \neq C(v')$, so if Φ_e maps v and v' to the same component in $\mathcal{H}$ then we immediately have R_e satisfied. We also assume that $\Phi_{e,s}(x) \downarrow \Rightarrow (\forall y < x)[\Phi_{e,s}(y) \downarrow]$.

Stage 0*:* Let $v_{e,0} = x_{e,0}$ for all $e \in \omega$. Let $h_0(0) = 0$. Let $\mathcal{H}_0$ have the single vertex 0 and no edges.

Stage $s+1$*:*

Step 1: Choose the least e such that $\Phi_{e,s+1}(v_{e,s+1}) \downarrow$ and $C_{s+1}(v_{e,s+1}) \cong C_{s+1}(\Phi_{e,s+1}(v_{e,s+1}))$, and such that $v_{e,s+1} = v_{e,s}$. If no such e exists, move to Step 2. If h^{-1} or h have already been re-defined at earlier stages by higher priority requirements on $\Phi_{e,s+1}(v_{e,s+1})$ or $h^{-1}(\Phi_{e,s+1}(v_{e,s+1}))$, respectively, then set $v_{e,s+1} = x_{n+1,s+1}$, where n is such that $x_{n,s} = v_{e,s}$. For $i > e$, let $v_{i,s+1} = x_{n+1+i-e,s+1}$. For $i < e$, let $v_{i,s+1} = x_{m,s+1}$, where m is such that $x_{m,s} = v_{i,s}$. Move to stage $s+2$.

Otherwise, speed up the enumeration of $\mathcal{G}$ and the approximation of X until we either find some $t > s$ such that $v_{e,t} \neq v_{e,s}$ (more precisely, $x_{n,t} \neq x_{n,s}$, where $v_{e,s} = x_{n,s}$), or we find some $t > s$ such that there exists $v \in \mathcal{G}_t$, $v \notin \text{dom}(h_s)$, and $\mathcal{C}_t(v_{e,s+1}) \prec C_t(v)$. In the first case, move to step 2. In the second case, re-define $\mathcal{H}$ setting $h_{s+1}(v) = \Phi_{e,s+1}(v_{e,s+1})$ and expand the component of $\Phi_{e,s+1}(v_{e,s+1})$ to be isomorphic to $C_t(v)$. Also introduce a new component isomorphic to $C_t(h_s^{-1}(\Phi_e(v_{e,s+1})))$ into $\mathcal{H}_{s+1}$, and define h_{s+1} on $C_t(h_s^{-1}(\Phi_e(v_{e,s+1})))$ accordingly.

Step 2: Let n be least such that $x_{n,s+1} \neq x_{n,s}$. For e such that $v_{e,s} = x_{m,s}$ with $m < n$, let $v_{e,s+1} = v_{e,s}$. Let e be least such that $v_{e,s} = x_{m,s}$ with $m \geq n$. For $i \geq e$, let $v_{i,s+1} = x_{n+i-e,s+1}$.

Step 3: For all new vertices v introduced into $\mathcal{G}_{s+1}$ (there may be more than 1 since we sped up the enumeration in step 1), if not already done so in step 1, introduce corresponding new vertices into $\mathcal{H}_{s+1}$. Extend h_{s+1} accordingly.

This completes the construction.

The correctness of the construction is based on the following observations. Firstly, for each e, $v_e = \lim_s v_{e,s}$ exists; this tells us that each requirement R_e is met and is eventually satisfied. Secondly, for each $v \in \mathcal{G}$, $h(v) = \lim_s h_s(v)$ exists, and that for each $w \in \mathcal{H}$, $h^{-1}(w) = \lim_s h_s^{-1}(w)$ exists. These together with the fact that at each stage s, $h_s : \mathcal{G}_s \cong \mathcal{H}_s$ show that h establishes an isomorphism between $\mathcal{G}$ and $\mathcal{H}$. Thus $\mathcal{G} \cong \mathcal{H}$, but $\mathcal{G}$ is not computable isomorphic to $\mathcal{H}$, and hence $\mathcal{G}$ is not computably categorical. □

We note that with essentially the same proof Theorem 2 can be strengthened by removing the assumption that the reachability relation is computable.

4 Examples

In this section, we provide some examples of strongly locally finite graphs on which the reachability relation is computable with various properties that are either computably categorical or not computably categorical. In our examples all the graphs have components of the following types.

Definition 2. *1. A **cycle** of length $n > 2$ is a graph isomorphic to $\{\{1, ..., n\}, E\}$, where $E = \{\{1, 2\}, \{2, 3\}, ..., \{n - 1, n\}, \{n, 1\}\}$. Denote this graph by $\mathcal{C}_n$.*
*2. A **sun** of size $n > 2$ is obtained by attaching a new edge to every vertex of a cycle of length n. Denote this graph by $\mathcal{S}_n$.*
*3. A **line** of length $n > 1$ is a graph isomorphic to $\{\{0, ..., n\}, E\}$, where $E = \{\{0, 1\}, \{1, 2\}, ..., \{n - 1, n\}\}$. Denote this graph by $\mathcal{L}_n$.*
4. $\mathcal{C}'_n$ is obtained by attaching exactly 1 edge to only one vertex of $\mathcal{C}_n$.
5. $\mathcal{C}''_n$ is obtained by attaching exactly 2 edges to only one vertex of $\mathcal{C}_n$.

Example 1. Let $\mathcal{G}_1$ be the graph that for each $n \geq 1$ contains a copy of $\mathcal{L}_n$. This graph is not computably categorical. Indeed, $\mathcal{G}_1$ has a presentation in which the size function is computable. Each component of this graph is embedded into ω many components. The rest follows from Lemma 2.

Example 2. There is a computably categorical graph such that in all computable presentations of the graph the size function is not computable. The desired graph is obtained as follows. Let $\mathcal{G}_2$ be the graph has a copy of $\mathcal{C}_n$ if $n \notin K$ and a copy of S_n if $n \in K$. The verification is left to the reader.

Example 3. There is a computably categorical graph such that in all computable presentations of the graph the size function is not computable. Indeed, let $\mathcal{G}_3$ be the disjoint union of the graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ described above. Then $\mathcal{G}_3$ is not computably categorical for the same reason that $\mathcal{G}_1$ is not, and the size function on $\mathcal{G}_3$ is intrinsically non-computable for the same reason as on $\mathcal{G}_2$.

In Theorem 1 we saw that for graphs on which the size function is computable, if the proper extension function is computable in all computable presentations, then the graph is computably categorical. We now generalize the definition of the proper extension function to graphs on which the size function need not be computable.

Definition 3. *For a graph $\mathcal{G}$ and a vertex v of $\mathcal{G}$, let $\rho(v) = |\{x \mid C(v) \prec C(x)\}|$. That is, $\rho(v)$ is the number of components of $\mathcal{G}$ into which the component of v can be properly embedded.*

Example 4. There exists a graph $\mathcal{G}_4$ that is not computably categorical, but that has a presentation on which the size function is computable.

We will simultaneously construct two computable presentations $\mathcal{G}_4 \cong \mathcal{H}_4$ that are not computably isomorphic, as follows.

Stage s: Introduce copies of $\mathcal{C}_s$ and $\mathcal{C}'_s$ into both $\mathcal{G}_{4,s}$ and $\mathcal{H}_{4,s}$. If $\Phi_{e,s}(v) \downarrow \in \mathcal{C}_e^{\mathcal{H}_{4,s}}$ for some $v \in \mathcal{C}_e^{\mathcal{G}_{4,s}}$, then extend $\mathcal{C}_e^{\mathcal{G}_{4,s}}$ to a copy of $\mathcal{C}'_e$ and extend $\mathcal{C}'^{\mathcal{G}_{4,s}}_e$ to a copy of $\mathcal{C}''_e$. In the other copy, extend $\mathcal{C}_e^{\mathcal{H}_{4,s}}$ to a copy of $\mathcal{C}''_e$. This ensures that Φ_e is not an isomorphism, but maintains $\mathcal{G}_{4,s+1} \cong \mathcal{H}_{4,s+1}$. It is not hard to show that the construction is correct.

Our final example is of a structure that is computably categorical and yet who's proper extension function is not computable. This, together with the previous example, shows that the condition on the size function was necessary for both parts of the equivalence between (1) and (3) in Theorem 1.

Example 5. There is a computably categorical graph on which the proper extension function is non-computable. Indeed, let $\mathcal{G}_5$ be the graph that has one copy of $\mathcal{C}_n$ and one copy of $\mathcal{S}_n$ if $n \notin K$, and two copies of $\mathcal{S}_n$ if $n \in K$. We leave it to the reader to show that the graph constructed has the desired properties.

5 Infinite Chains of Embedded Components

From the two theorems above, one may suggest that the existence of an infinite chain of properly embedded components in a graph may imply that the graph is not computably categorical. One may also suggest that the Δ^0_2-bound in Theorem 2 could be replaced with a Σ^0_2-bound. The main result of this section is to refute these two suggestions and outline of a proof for the following result:

Theorem 3. *There is strongly locally finite computably categorical graph that possesses an infinite chain of properly embedded components. In fact, the set* $\{v \mid C(v)$ *is properly embedded into* ω *many components*$\}$ *is computable in* $0''$.

Proof. Let $\Phi_0, \Phi_1, \ldots$ be a standard enumeration of all partial computable functions from ω^2 to $\{0,1\}$. Based on this, one builds an effective enumeration of all computable graphs $\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2, \ldots$ uniformly. On i at stage t we have: (1) $V_{i,t} \subseteq V_{i,t+1}$, $E_{i,t} \subseteq E_{i,t+1}$ for $t \in \omega$; (2) $\bigcup_t \mathcal{G}_{i,t} = G_i$, where $\mathcal{G}_{i,t} = (V_{i,t}, E_{i,t})$; (3) $V_{i,t} = \{0, \ldots, k_{i,t}\}$ where $k_{i,t}$ is the maximal $j \leq t$ such that for all $n, m \leq j$ the values $\Phi_{i,t}(n,m)$ are defined.

Our goal is to construct a graph $\mathcal{G} = (\omega, E)$ such that $\mathcal{G}$ has an infinite sequence $C_0 \prec C_1 \prec C_2 \prec \ldots$ of properly embedded components, and the construction of $\mathcal{G}$ meets the following requirements:

$$R_e : \text{If } \mathcal{G}_e \cong \mathcal{G} \text{ then } \mathcal{G}_e \text{ is computably isomorphic to } \mathcal{G}$$

Here we show how to satisfy just one requirement R_e. The general construction (that we omit in this paper due to space limitations) is based on putting all strategies for R_e on a priority tree. The general construction produces a true path through the tree, the true path is computable in $0''$, and all the requirements R_e are satisfied along the path. The general construction is somewhat similar to and simpler than the constructions in [1], [2], and [13] .

The rest of the proof will handle one requirement R_e. We need some notation and definitions. We use cycles as defined in the previous section.

Let H be a graph and v be its vertex. To **attach** a cycle $\mathcal{C}_n$ to v means to extend the graph H by adjoining to H the graph $\mathcal{C}_n$ and adding the edge $\{v, 1\}$.

The graph $\mathcal{G}$ that we construct will be strongly locally finite such that each component of $\mathcal{G}$ will consist of a vertex v together with finitely many cycles attached to v. We call such components **special-cyclic components**.

We approximate $\mathcal{G}_e$ as $\mathcal{G}_{e,0} \subseteq \mathcal{G}_{e,1} \subseteq \mathcal{G}_{e,2} \subseteq \ldots$ such that every component of $\mathcal{G}_{e,t}$ is special-cyclic. During the construction we guarantee the following. If $\mathcal{G}_{e,t}$ provides a component C that can not be embedded into $\mathcal{G}_t$ then C will never be embedded into $\mathcal{G}$. In this case R_e is satisfied, and we ensure that $\mathcal{G}$ has an infinite sequence of properly embedded components. Thus, we can always assume that $\mathcal{G}_{e,t}$ is embedded into $\mathcal{G}$ currently built. During the construction we also guarantee that no two components of $\mathcal{G}$ are isomorphic.

The graph $\mathcal{G}_t$ denotes approximation to $\mathcal{G}$ at stage t. Components of $\mathcal{G}_t$ are denoted by $H_{j,t}$, and we assume a natural order between the components (e.g. $H_1 < H_2$ if the minimum vertex in H_1 is less than the minimum vertex in H_2).

At stage t, the function f_t denotes a partial isomorphism from $\mathcal{G}_{e,t}$ into $\mathcal{G}_t$ that we build. We will also have finitely many selected components in $\mathcal{G}_t$. We say $H_{j,t}$ (a component of $\mathcal{G}_t$) is **covered** if there is a component $H_{e,j,t}$ of $\mathcal{G}_e$ such that f_t maps $H_{e,j,t}$ into $H_{j,t}$. We say that R_e is in the **waiting state** (at stage t) if there are selected and uncovered components of $\mathcal{G}_t$. We say that R_e **recovers** (at stage t) if for every selected component H in $\mathcal{G}_t$ there is a (necessarily) unique component H' in $\mathcal{G}_{e,t}$ such that H' embeds into H and H' can not be embedded into any other component of $\mathcal{G}_t$ and f has not been defined on H'. Now we describe our stagewise construction of $\mathcal{G}$ against one R_e.

Stage 0. Set $\mathcal{G}_0$ to contain two special-cyclic components such that one component has a cycle of length 3 attached and the other has a cycle of length 4 attached. Select both components. *Mark* the first component. R_e is now in the waiting state. The function f_0 is empty.

Stage $t+1$. Compute $\mathcal{G}_{e,t+1}$. Assume R_e is in the waiting state. Build a new component H in $\mathcal{G}_t$ such that the component built in the previous stage is properly embedded into H. This builds $\mathcal{G}_{t+1}$.

Assume R_e has recovered. Let $\mathcal{H}_1$ be the marked component. Let $\mathcal{H}_2$ be the first selected component such that $\mathcal{H}_2$ is not marked. For every selected component H, consider H' in $\mathcal{G}_{e,t+1}$ such that H' embeds into H, f_t is not defined on H', and H' does not embed into any other component of $\mathcal{G}_t$. Extend f_t to f_{t+1} by mapping all such H' into H. Extend $\mathcal{G}_t$ to $\mathcal{G}_{t+1}$ as follows:

1. Let t' be the last recovery stage before stage $t+1$. To all components built between stages $t'+1$ and $t+1$ attach new and distinct cycles of distinct

unused lengths. This makes these components non-embeddable into each other. Declare these components newly selected.
2. Declare all the components selected at stage t' unselected.
3. Consider H_1 and $\mathcal{H}_2$. To H_2 attach a cycle of length n if H_1 has a cycle of length n attached to it and H_2 has no cycle of length n attached. Remove the mark from H_1 and mark the newly extended H_2. Declare H_2 selected.
4. Construct a new component with a new cycle of unused length in $\mathcal{G}_{t+1}$.

This finishes the description of stage $t+1$. Set $\mathcal{G} = \bigcup_t \mathcal{G}_t$. Now we show that $\mathcal{G}$ is a desired graph.

Lemma 4. *Suppose there is a stage t after which R_e never recovers. Then $\mathcal{G}$ has an infinite chain of properly embedded components and R_e is satisfied.*

Proof. After stage t, the construction builds an infinite chain of properly embedded components. Also, $\mathcal{G}_e \not\cong \mathcal{G}'$ and hence R_e is satisfied. □

Lemma 5. *If $\mathcal{G}_e \cong \mathcal{G}$ then $\bigcup_t f_t$ effectively extends to an isomorphism.*

Proof. It must be the case that Φ_e is total. Let H_1 be a component of $\mathcal{G}$.

Case 1. The component H_1 is never marked. In this case, by construction, H_1 must contain a cycle of length n such that no other component of $\mathcal{G}$ has a cycle attached of the same length. Assume H_1 is selected at stage t'. In the next recovery stage $t+1$, f_{t+1} maps $H'_{1,t+1}$ into $H_{1,t+1}$. Since H'_1 is the only component that contains a cycle of length n, we will have $H_1 \cong H'_1$.

Case 2. Assume H_1 is marked at stage t' and let $t+1$ be the next recovery stage after t'. We can assume that $f_{t'}$ maps H'_1 to H_1.

At stage $t+1$ we have H_2 (see stage $t+1$). H_2 contains a cycle of length m such that no other component has a cycle of length m. At stage $t+1$ we also have a mapping f_{t+1} such that f_{t+1} maps $H'_{1,t+1}$ to $H_{1,t+1}$ and $H'_{2,t+1}$ to $H_{2,t+1}$ and $f_{t'} \subseteq f_{t+1}$. Let t_1 be the next recovery stage after $t+1$. Again it must be the case that $f_{t+1} \subseteq f_{t_1}$ as otherwise $\mathcal{G}_e$ contains two components containing cycles of length m (after which the construction guarantees that $\mathcal{G}$ contains no two components with cycles of length m). □

Lemma 6. *Assume $\mathcal{G}_e \cong \mathcal{G}$. Then $\mathcal{G}$ contains an infinite chain of properly embedded components.*

Proof. The components marked at recovery stages form the desired chain. □

These lemmas prove that the construction is correct to satisfy one R_e. In the general construction our priority T will be the binary tree over the alphabet r, w with the order $r < w$, where r represents recovery and w represents the waiting state. The nodes of length e in T will be devoted to satisfy R_e. □

References

1. Peter Cholak, Sergey Goncharov, Bakhadyr Khoussainov, and Richard A. Shore. Computably categorical structures and expansions by constants. *J. Symbolic Logic*, 64(1):13–37, 1999.
2. Peter Cholak, Richard A. Shore, and Reed Solomon. A computably stable structure with no Scott family of finitary formulas. *Arch. Math. Logic*, 45(5):519–538, 2006.
3. John N. Crossley, editor. *Aspects of effective algebra*, Vic., 1981. Upside Down A Book Co. Yarra Glen.
4. V. D. Dzgoev and S. S. Gončarov. Autostability of models. *Algebra i Logika*, 19:45–58, 1980.
5. Yu. L. Ershov, S. S. Goncharov, A. Nerode, J. B. Remmel, and V. W. Marek, editors. *Handbook of recursive mathematics. Vol. 1*, volume 138 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1998. Recursive model theory.
6. Yu. L. Ershov, S. S. Goncharov, A. Nerode, J. B. Remmel, and V. W. Marek, editors. *Handbook of recursive mathematics. Vol. 2*, volume 139 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1998. Recursive algebra, analysis and combinatorics.
7. Yuri L. Ershov and Sergei S. Goncharov. *Constructive models*. Siberian School of Algebra and Logic. Consultants Bureau, New York, 2000.
8. S. S. Gončarov. Autostability of models and abelian groups. *Algebra i Logika*, 19(1):23–44, 132, 1980.
9. S. S. Gončarov. The problem of the number of nonautoequivalent constructivizations. *Algebra i Logika*, 19(6):621–639, 745, 1980.
10. S. S. Goncharov. Limit equivalent constructivizations. In *Mathematical logic and the theory of algorithms*, volume 2 of *Trudy Inst. Mat.*, pages 4–12. "Nauka" Sibirsk. Otdel., Novosibirsk, 1982.
11. Sergei S. Goncharov. Computability and computable models. In *Mathematical problems from applied logic. II*, volume 5 of *Int. Math. Ser. (N. Y.)*, pages 99–216. Springer, New York, 2007.
12. Sergey S. Goncharov, Steffen Lempp, and Reed Solomon. The computable dimension of ordered abelian groups. *Adv. Math.*, 175(1):102–143, 2003.
13. Denis R. Hirschfeldt. Degree spectra of relations on structures of finite computable dimension. *Ann. Pure Appl. Logic*, 115(1-3):233–277, 2002.
14. Steffen Lempp, Charles McCoy, Russell Miller, and Reed Solomon. Computable categoricity of trees of finite height. *J. Symbolic Logic*, 70(1):151–215, 2005.
15. A. I. Malćev. Constructive algebras. I. *Uspehi Mat. Nauk*, 16(3 (99)):3–60, 1961.
16. Michael O. Rabin. Computable algebra, general theory and theory of computable fields. *Trans. Amer. Math. Soc.*, 95:341–360, 1960.
17. Jeffrey B. Remmel. Recursively categorical linear orderings. *Proc. Amer. Math. Soc.*, 83:387–391, 1981.