Relativized computable categoricity

Workshop on Reverse Mathematics at the Erwin Schrödinger Institute in Vienna, Austria

Java Darleen Villano

August 6th, 2025

University of Toronto

This talk is based on work from two papers: "Computable categoricity relative to a c.e. degree" (arXiv:2401.06641) and "Extensions of categoricity relative to a degree" (arXiv:2505.15706).

Preprints of both are also available on my website.

Outline

1. Background
2. Outside of the c.e. degrees
3. Outside of the class of directed graphs
4. Sketch of construction for the 1-generic result

Background

Computable categoricity

Definition

A computable structure $\mathcal{A}$ is **computably categorical** if for every computable copy $\mathcal{B}$ of $\mathcal{A}$, there exists a computable isomorphism between $\mathcal{A}$ and $\mathcal{B}$.

Computable categoricity

Definition

A computable structure $\mathcal{A}$ is **computably categorical** if for every computable copy $\mathcal{B}$ of $\mathcal{A}$, there exists a computable isomorphism between $\mathcal{A}$ and $\mathcal{B}$.

$(\mathbb{Q}, \leq)$ as a linear order is computably categorical, whereas $(\mathbb{N}, \leq)$ is not computably categorical.

Relativizing categoricity

Definition

A computable structure $\mathcal{A}$ is **relatively computably categorical** if for every copy (not necessarily computable) $\mathcal{B}$ of $\mathcal{A}$, there is a $\mathcal{B}$-computable isomorphism between $\mathcal{A}$ and $\mathcal{B}$.

Relativizing categoricity

Definition

A computable structure $\mathcal{A}$ is **relatively computably categorical** if for every copy (not necessarily computable) $\mathcal{B}$ of $\mathcal{A}$, there is a $\mathcal{B}$-computable isomorphism between $\mathcal{A}$ and $\mathcal{B}$.

Definition

For a Turing degree $\mathbf{d}$, a computable structure $\mathcal{A}$ is **computably categorical relative to d** if for every $\mathbf{d}$-computable copy $\mathcal{B}$ of $\mathcal{A}$, there is a $\mathbf{d}$-computable isomorphism between $\mathcal{A}$ and $\mathcal{B}$.

Relativizing categoricity

Definition

A computable structure $\mathcal{A}$ is **relatively computably categorical** if for every copy (not necessarily computable) $\mathcal{B}$ of $\mathcal{A}$, there is a $\mathcal{B}$-computable isomorphism between $\mathcal{A}$ and $\mathcal{B}$.

Definition

For a Turing degree $\mathbf{d}$, a computable structure $\mathcal{A}$ is **computably categorical relative to d** if for every $\mathbf{d}$-computable copy $\mathcal{B}$ of $\mathcal{A}$, there is a $\mathbf{d}$-computable isomorphism between $\mathcal{A}$ and $\mathcal{B}$.

A structure $\mathcal{A}$ is relatively computably categorical if it is computably categorical relative to all degrees $\mathbf{d}$.

Turing cones

Given a computable structure $\mathcal{A}$, we can consider the following set of degrees.

$$D_{\mathcal{A}} = \{\mathbf{d} : \mathcal{A} \text{ is computably categorical relative to } \mathbf{d}\}.$$

If we assume enough determinacy, then this set of degrees either contains a cone in the Turing degrees or is disjoint from one.

Turing cones

Given a computable structure $\mathcal{A}$, we can consider the following set of degrees.

$$D_{\mathcal{A}} = \{\mathbf{d} : \mathcal{A} \text{ is computably categorical relative to } \mathbf{d}\}.$$

If we assume enough determinacy, then this set of degrees either contains a cone in the Turing degrees or is disjoint from one.

Definition (Csima, Harrison-Trainor [CHT17])

A structure $\mathcal{A}$ is **computably categorical on a cone above d** if for all $\mathbf{c} \geq \mathbf{d}$, whenever $\mathcal{B}$ and $\mathcal{C}$ are $\mathbf{c}$-computable copies of $\mathcal{A}$, there is a $\mathbf{c}$-computable isomorphism between $\mathcal{B}$ and $\mathcal{C}$.

Turing cones

If $\mathcal{A}$ is relatively computably categorical, then we can think of it as being computably categorical on a cone above $\mathbf{0}$.

Turing cones

If $\mathcal{A}$ is relatively computably categorical, then we can think of it as being computably categorical on a cone above $\mathbf{0}$.

$\mathcal{A}$ is computably categorical on a cone above a degree $\mathbf{d}$ if and only if it is computably categorical relative to all $\mathbf{c} \geq \mathbf{d}$.

Turing cones

If $\mathcal{A}$ is relatively computably categorical, then we can think of it as being computably categorical on a cone above $\mathbf{0}$.

$\mathcal{A}$ is computably categorical on a cone above a degree $\mathbf{d}$ if and only if it is computably categorical relative to all $\mathbf{c} \geq \mathbf{d}$.

Fact (by [Ash+89] and [Gon80])

If $\mathcal{A}$ is a computable structure and it is computably categorical relative to some degree $\mathbf{d} \geq \mathbf{0}''$, then $\mathcal{A}$ has a $\mathbf{0}''$-computable Σ^0_1 Scott family. In particular, $\mathcal{A}$ is computably categorical relative to all $\mathbf{d} \geq \mathbf{0}''$.

Turing cones

If $\mathcal{A}$ is relatively computably categorical, then we can think of it as being computably categorical on a cone above $\mathbf{0}$.

$\mathcal{A}$ is computably categorical on a cone above a degree $\mathbf{d}$ if and only if it is computably categorical relative to all $\mathbf{c} \geq \mathbf{d}$.

Fact (by [Ash+89] and [Gon80])

If $\mathcal{A}$ is a computable structure and it is computably categorical relative to some degree $\mathbf{d} \geq \mathbf{0}''$, then $\mathcal{A}$ has a $\mathbf{0}''$-computable Σ^0_1 Scott family. In particular, $\mathcal{A}$ is computably categorical relative to all $\mathbf{d} \geq \mathbf{0}''$.

So the set $D_{\mathcal{A}}$ either contains the cone above $\mathbf{0}''$ or does not contain any cone at all.

In the c.e. degrees, being computably categorical relative to a degree is not monotonic in the following way.

Theorem (Downey, Harrison-Trainor, Melnikov [DHTM21])

There is a computable structure $\mathcal{A}$ and c.e. degrees $\mathbf{0} = \mathbf{d}_0 <_T \mathbf{e}_0 <_T \mathbf{d}_1 <_T \mathbf{e}_1 <_T \dots$ such that

(1) *$\mathcal{A}$ is computably categorical relative to $\mathbf{d}_i$ for each i,*

(2) *$\mathcal{A}$ is not computably categorical relative to $\mathbf{e}_i$ for each i,*

(3) *$\mathcal{A}$ is computably categorical relative to $\mathbf{0}'$.*

We can extend the DHTM result to partial orders of c.e. degrees.

Theorem (V.)

Let $P = (P, \leq)$ be a computable partially ordered set and let $P = P_0 \sqcup P_1$ be a computable partition. Then, there exists a computable directed graph $\mathcal{G}$ and an embedding h of P into the c.e. degrees where

(1) *$\mathcal{G}$ is computably categorical;*
(2) *$\mathcal{G}$ is computably categorical relative to each degree in $h(P_0)$; and*
(3) *$\mathcal{G}$ is not computably categorical relative to each degree in $h(P_1)$.*

We can extend the DHTM result to partial orders of c.e. degrees.

Theorem (V.)

Let $P = (P, \leq)$ be a computable partially ordered set and let $P = P_0 \sqcup P_1$ be a computable partition. Then, there exists a computable directed graph $\mathcal{G}$ and an embedding h of P into the c.e. degrees where

(1) *$\mathcal{G}$ is not computably categorical;*

(2) *$\mathcal{G}$ is computably categorical relative to each degree in $h(P_0)$; and*

(3) *$\mathcal{G}$ is not computably categorical relative to each degree in $h(P_1)$.*

Outside of the c.e. degrees

Definition

A set A is **n-generic** if for all Σ_n^0 set of strings $S \subseteq 2^{<\omega}$, there exists an m such that either $A \upharpoonright m \in S$ or for all $\tau \supseteq A \upharpoonright m$, $\tau \notin S$. A degree **d** is **n-generic** if it contains an n-generic set.

In the generic degrees

Definition

A set A is **n-generic** if for all Σ^0_n set of strings $S \subseteq 2^{<\omega}$, there exists an m such that either $A \upharpoonright m \in S$ or for all $\tau \supseteq A \upharpoonright m$, $\tau \notin S$. A degree **d** is **n-generic** if it contains an n-generic set.

Definition

A degree **d** is **low for isomorphism** if for every pair of computable structures $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A} \cong_{\mathbf{d}} \mathcal{B}$ if and only if $\mathcal{A} \cong_{\Delta^0_1} \mathcal{B}$.

In the generic degrees

Theorem (Franklin, Solomon [FS14])

Every 2-generic degree is low for isomorphism.

In the generic degrees

Theorem (Franklin, Solomon [FS14])

Every 2-generic degree is low for isomorphism.

This means that there *cannot* be a computable structure $\mathcal{A}$ which is not computably categorical but is computably categorical relative to $\mathbf{d}$ for a 2-generic degree $\mathbf{d}$.

In the generic degrees

Theorem (Franklin, Solomon [FS14])

Every 2-generic degree is low for isomorphism.

This means that there *cannot* be a computable structure $\mathcal{A}$ which is not computably categorical but is computably categorical relative to $\mathbf{d}$ for a 2-generic degree $\mathbf{d}$.

Theorem (V.)

There exists a (properly) 1-generic G such that there is a computable directed graph $\mathcal{A}$ where $\mathcal{A}$ is not computably categorical but is computably categorical relative to G.

Outside of the class of directed graphs

Beyond directed graphs

Question

For structures other than directed graphs, can you produce an example which witnesses the pathological behavior in the poset result?

Beyond directed graphs

Question

For structures other than directed graphs, can you produce an example which witnesses the pathological behavior in the poset result?

Corollary (from results in [Hir+02] and [Mil+18])

For the following classes of structures, there exists a computable example in each class which witnesses the behavior in the poset result:

(1) *symmetric, irreflexive graphs; partial orderings; lattices; rings with zero-divisors; integral domains of arbitrary characteristic; commutative semigroups; and 2-step nilpotent groups (Theorem 1.22 of [Hir+02]); and*

(2) *countable fields (Theorem 1.8 of [Mil+18]).*

Linear orders

Theorem (Remmel [Rem81])

A computable linear order is computably categorical if and only if it has finitely many adjacent pairs.

Theorem (Remmel [Rem81])

A computable linear order is computably categorical if and only if it has finitely many adjacent pairs.

In fact, every linear order which is computably categorical also happens to be relatively computably categorical.

Theorem (Remmel [Rem81])

A computable linear order is computably categorical if and only if it has finitely many adjacent pairs.

In fact, every linear order which is computably categorical also happens to be relatively computably categorical.

Question

For a linear order which is not computably categorical, can it be computably categorical relative to some $\mathbf{d} > \mathbf{0}$*?*

Linear orders

Theorem (Remmel [Rem81])

A computable linear order is computably categorical if and only if it has finitely many adjacent pairs.

In fact, every linear order which is computably categorical also happens to be relatively computably categorical.

Question

For a linear order which is not computably categorical, can it be computably categorical relative to some $\mathbf{d} > \mathbf{0}$*?*

No; Remmel's construction in the backwards direction relativizes to any degree $\mathbf{d}$.

In summary

For some classes of structures and degrees, there exists some structure whose categorical behavior relative to a degree can change (e.g., from being computably categorical to not being computably categorical relative to a degree $\mathbf{d} > \mathbf{0}$).

In summary

For some classes of structures and degrees, there exists some structure whose categorical behavior relative to a degree can change (e.g., from being computably categorical to not being computably categorical relative to a degree $\mathbf{d} > \mathbf{0}$).

For other classes of structures (linear orders, Boolean algebras) and degrees (2-generics), there are **no** structures who can change their categorical behavior relative to a degree.

In summary

For some classes of structures and degrees, there exists some structure whose categorical behavior relative to a degree can change (e.g., from being computably categorical to not being computably categorical relative to a degree $\mathbf{d} > \mathbf{0}$).

For other classes of structures (linear orders, Boolean algebras) and degrees (2-generics), there are **no** structures who can change their categorical behavior relative to a degree. In particular, for computable linear orders, their limiting behavior for categoricity relative to a degree already stabilizes on the cone above $\mathbf{0}$.

Sketch of construction for the 1-generic result

Requirements

Theorem (V.)

There exists a (properly) 1-generic G such that there is a computable directed graph $\mathcal{A}$ where $\mathcal{A}$ is not computably categorical but is computably categorical relative to G.

Requirements

Theorem (V.)

There exists a (properly) 1-generic G such that there is a computable directed graph $\mathcal{A}$ where $\mathcal{A}$ is not computably categorical but is computably categorical relative to G.

We have the following requirements:

- R_j : $(\exists \sigma \subseteq G)(\sigma \in W_j \vee (\forall \tau \supseteq \sigma)(\tau \notin W_j))$,
- P_e : $\Phi_e : \mathcal{A} \to \mathcal{B}$ is not an isomorphism, and
- S_i : if $\mathcal{A} \cong \mathcal{M}_i^G$, then there exists a G-computable isomorphism $f_i^G : \mathcal{A} \to \mathcal{M}_i^G$.

Building $\mathcal{A}$ in stages

We build the computable directed graph $\mathcal{A}$ in stages.

Building $\mathcal{A}$ in stages

We build the computable directed graph $\mathcal{A}$ in stages.

At stage $s = 0$, we set the domain of $\mathcal{A}$ to be empty.

Building $\mathcal{A}$ in stages

We build the computable directed graph $\mathcal{A}$ in stages.

At stage $s = 0$, we set the domain of $\mathcal{A}$ to be empty.

At stage $s > 0$, we add two new connected components by adding a_{2s} and a_{2s+1} as root nodes. We attach 2-loop to each node. Then, we attach a $(5s+1)$-loop to a_{2s} and a $(5s+2)$-loop to a_{2s+1}.

Building $\mathcal{A}$ in stages

We build the computable directed graph $\mathcal{A}$ in stages.

At stage $s = 0$, we set the domain of $\mathcal{A}$ to be empty.

At stage $s > 0$, we add two new connected components by adding a_{2s} and a_{2s+1} as root nodes. We attach 2-loop to each node. Then, we attach a $(5s + 1)$-loop to a_{2s} and a $(5s + 2)$-loop to a_{2s+1}.

Definition

The root node a_{2s} in our graph $\mathcal{A}$ with its loops is the **2sth connected component** or just the 2sth component of $\mathcal{A}$.

Basic strategies: P_e

This is our basic strategy to satisfy all P_e requirements.

Let s be the current stage of the construction and let α be a P_e-strategy.

1. If α is first eligible to act at stage s, it defines its witness n_α to be a large unused number. Let $n = n_\alpha$.

Basic strategies: P_e

This is our basic strategy to satisfy all P_e requirements.

Let s be the current stage of the construction and let α be a P_e-strategy.

1. If α is first eligible to act at stage s, it defines its witness n_α to be a large unused number. Let $n = n_\alpha$.
2. Check if Φ_e maps the $2n$th and $(2n+1)$st components of $\mathcal{A}$ to the $2n$th and $(2n+1)$st components of $\mathcal{B}$, respectively.

Basic strategies: P_e

This is our basic strategy to satisfy all P_e requirements.

Let s be the current stage of the construction and let α be a P_e-strategy.

1. If α is first eligible to act at stage s, it defines its witness n_α to be a large unused number. Let $n = n_\alpha$.
2. Check if Φ_e maps the $2n$th and $(2n+1)$st components of $\mathcal{A}$ to the $2n$th and $(2n+1)$st components of $\mathcal{B}$, respectively. If not, α takes no action at stage s.

This is our basic strategy to satisfy all P_e requirements.

Let s be the current stage of the construction and let α be a P_e-strategy.

1. If α is first eligible to act at stage s, it defines its witness n_α to be a large unused number. Let $n = n_\alpha$.
2. Check if Φ_e maps the $2n$th and $(2n+1)$st components of $\mathcal{A}$ to the $2n$th and $(2n+1)$st components of $\mathcal{B}$, respectively. If not, α takes no action at stage s. If so, α takes action by adding new loops to these components in $\mathcal{A}$ and $\mathcal{B}$.

Basic strategies: S_i

The general goal of our basic strategy to satisfy all S_i requirements is the following. Let α be an S_i-strategy.

Basic strategies: S_i

The general goal of our basic strategy to satisfy all S_i requirements is the following. Let α be an S_i-strategy.

For each n, we try to find copies of the $2n$th and $(2n+1)$st components of $\mathcal{A}$ in $\mathcal{M}_i^G$.

Basic strategies: S_i

The general goal of our basic strategy to satisfy all S_i requirements is the following. Let α be an S_i-strategy.

For each n, we try to find copies of the $2n$th and $(2n+1)$st components of $\mathcal{A}$ in $\mathcal{M}_i^G$. Initial segments of G can change throughout the construction, and so loops in $\mathcal{M}_i^G$ or embeddings using certain initial segments of G can disappear or reappear.

Basic strategies: S_i

When α is next eligible to act at stage s, it can check if an initial segment of G has changed up to some previously defined use for an f_i^G-computation at that point in the construction.

Basic strategies: S_i

When α is next eligible to act at stage s, it can check if an initial segment of G has changed up to some previously defined use for an f_i^G-computation at that point in the construction.

This will determine what parameter, n_α, α will work with when trying to match $\mathcal{A}[s]$-components with their copies (if any) in $\mathcal{M}_i^G[s]$.

Interactions between strategies

There are several interactions and conflicts to keep note of in the construction.

Interactions between strategies

There are several interactions and conflicts to keep note of in the construction.

Interaction 1

The P_e requirement wants to diagonalize while the S_i requirements want to build embeddings: this can primarily be resolved by having P_e "wait" for higher priority S_i requirements.

Interactions between strategies

There are several interactions and conflicts to keep note of in the construction.

Interaction 1

The P_e requirement wants to diagonalize while the S_i requirements want to build embeddings: this can primarily be resolved by having P_e "wait" for higher priority S_i requirements.

Interaction 2

Changes in initial segments of G can make computations which disappeared because of a diagonalization reappear again: this can be resolved by making pairs of $\mathcal{A}$-components indistinct.

Thanks

Thank you for your attention!

I'd be happy to answer any questions.

References

[Ash+89] C. Ash et al. **"Generic copies of countable structures".** *Annals of Pure and Applied Logic* 42.3 (1989), pp. 195–205. ISSN: 0168-0072.

[CHT17] B. F. Csima and M. Harrison-Trainor. **"Degrees of Categoricity on a Cone via η-systems".** *The Journal of Symbolic Logic* 82.1 (2017), pp. 325–346. ISSN: 00224812, 19435886.

[DHTM21] R. Downey, M. Harrison-Trainor, and A. Melnikov. **"Relativizing computable categoricity".** *Proc. Amer. Math. Soc.* 149.9 (2021), pp. 3999–4013. ISSN: 0002-9939.

[FS14] J. N. Y. Franklin and R. Solomon. **"Degrees that Are Low for Isomorphism".** *Computability* 3 (2014), pp. 73–89.

[Gon80] S. S. Goncharov. **"The problem of the number of nonautoequivalent constructivizations".** *Algebra i Logika* 19.6 (1980), pp. 621–639, 745.

[Hir+02] D. R. Hirschfeldt et al. **"Degree spectra and computable dimensions in algebraic structures".** *Annals of Pure and Applied Logic* 115.1 (2002), pp. 71–113. ISSN: 0168-0072.

[Mil+18] R. Miller et al. **"A Computable Functor from Graphs to Fields".** *The Journal of Symbolic Logic* 83.1 (2018), 326–348.

[Rem81] J. B. Remmel. **"Recursively Categorical Linear Orderings".** *Proc. Amer. Math. Soc.* 83.2 (1981), pp. 387–391.