Computable categoricity relative to a c.e. degree

CT Logic Seminar, Fall 2023

Java Villano

October 3rd, 2023

University of Connecticut

Outline

1. Overview on computable categoricity

    Historical overview

2. Computable categoricity relative to a degree

    Focusing on the c.e. degrees

    Main result

    Outline of strategies

Overview on computable categoricity

Computable categoricity

Definition

Let $\mathcal{A}$ be 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

Let $\mathcal{A}$ be 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 $\mathcal{B}$.

Throughout the talk, I will abbreviate "being computably categorical" to **being c.c.**

Computable categoricity

Definition

Let $\mathcal{A}$ be 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 $\mathcal{B}$.

Throughout the talk, I will abbreviate "being computably categorical" to **being c.c.**

Example

Let $L = (A, <_L)$ be a computable linear ordering. Two elements $a, b \in A$ are said to be **adjacent** if $a <_L b$ and there is no $c \in A$ such that $a <_L c <_L b$.

L is c.c. if and only if it has only finitely many pairs of adjacent elements (Remmel [7]).

Relativizing c.c.-ness

The following relativization of c.c.-ness has been the studied extensively in the past.

Relativizing c.c.-ness

The following relativization of c.c.-ness has been the studied extensively in the past.

Definition

Let $\mathcal{A}$ be 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 c.c.-ness

The following relativization of c.c.-ness has been the studied extensively in the past.

Definition

Let $\mathcal{A}$ be 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 $\mathcal{B}$.

Remark

If a structure is relatively computably categorical (abbreviated as **relatively c.c.***), then it is c.c. already.*

Relativizing c.c.-ness

The following relativization of c.c.-ness has been the studied extensively in the past.

Definition

Let $\mathcal{A}$ be 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 $\mathcal{B}$.

Remark

If a structure is relatively computably categorical (abbreviated as **relatively c.c.***), then it is c.c. already.*

Historically, there have been two approaches in exploring the connection between c.c.-ness and relatively c.c.-ness:

Relativizing c.c.-ness

The following relativization of c.c.-ness has been the studied extensively in the past.

Definition

Let $\mathcal{A}$ be 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 $\mathcal{B}$.

Remark

If a structure is relatively computably categorical (abbreviated as **relatively c.c.***), then it is c.c. already.*

Historically, there have been two approaches in exploring the connection between c.c.-ness and relatively c.c.-ness: an **algebraic** perspective and a **model theoretic** perspective.

The algebraic perspective

Question: Can we give algebraic characterizations of c.c.-ness in natural classes of structures?

The algebraic perspective

Question: Can we give algebraic characterizations of c.c.-ness in natural classes of structures?

Example

- Remmel [7] showed that a computable linear ordering L is c.c. if and only if L has only finitely many pairs of adjacent elements.

The algebraic perspective

Question: Can we give algebraic characterizations of c.c.-ness in natural classes of structures?

Example

- Remmel [7] showed that a computable linear ordering L is c.c. if and only if L has only finitely many pairs of adjacent elements.
- Ershov [4] showed that an algebraically closed field is c.c. if and only if it has a finite transcendence degree over its prime subfield.

The algebraic perspective

Question: Can we give algebraic characterizations of c.c.-ness in natural classes of structures?

Example

- Remmel [7] showed that a computable linear ordering L is c.c. if and only if L has only finitely many pairs of adjacent elements.
- Ershov [4] showed that an algebraically closed field is c.c. if and only if it has a finite transcendence degree over its prime subfield.
- Goncharov, Lempp, and Solomon [5] showed that an ordered abelian group is c.c. if and only if it has finite rank.

Algebraic characterizations under relativization

Typically, if there is an algebraic characterization for being c.c. in a class of structures, then being relatively c.c. is equivalent to being c.c. for those structures.

Algebraic characterizations under relativization

Typically, if there is an algebraic characterization for being c.c. in a class of structures, then being relatively c.c. is equivalent to being c.c. for those structures.

For example, suppose a computable linear order L has finitely many adjacent pairs, and L' is any copy of L.

Algebraic characterizations under relativization

Typically, if there is an algebraic characterization for being c.c. in a class of structures, then being relatively c.c. is equivalent to being c.c. for those structures.

For example, suppose a computable linear order L has finitely many adjacent pairs, and L' is any copy of L.

We can build an isomorphism by nonuniformly matching the finitely many adjacent pairs correctly, and then extending the map via a back-and-forth construction on the leftover dense intervals.

Typically, if there is an algebraic characterization for being c.c. in a class of structures, then being relatively c.c. is equivalent to being c.c. for those structures.

For example, suppose a computable linear order L has finitely many adjacent pairs, and L' is any copy of L.

We can build an isomorphism by nonuniformly matching the finitely many adjacent pairs correctly, and then extending the map via a back-and-forth construction on the leftover dense intervals.

To do the back-and-forth construction, we only need to be able to compute $\leq_L$ and $\leq_{L'}$, and so the isomorphism will be computable in L'.

Additional conditions for equivalence

For structures where there is no algebraic characterization of being c.c., what other conditions do you need to be relatively c.c.?

Additional conditions for equivalence

For structures where there is no algebraic characterization of being c.c., what other conditions do you need to be relatively c.c.?

Model theory can help answer that question.

The model theoretic perspective

A structure being relatively c.c. coincides with the following syntactic condition.

The model theoretic perspective

A structure being relatively c.c. coincides with the following syntactic condition.

Theorem (Ash, Knight, Manasse, and Slaman [1])

A structure is relatively c.c. if and only if it has a formally Σ_1 *Scott family.*

The model theoretic perspective

A structure being relatively c.c. coincides with the following syntactic condition.

Theorem (Ash, Knight, Manasse, and Slaman [1])

A structure is relatively c.c. if and only if it has a formally Σ_1 Scott family.

Definition

A **formally Σ_1 Scott family** for $\mathcal{A}$ is a c.e. set W of $\exists$-formulas with a fixed finite set of parameters such that

(1) for every $\overline{a} \in \mathcal{A}$, there is a $\varphi(\overline{x}) \in W$ where $\mathcal{A} \models \varphi(\overline{a})$, and
(2) for every $\overline{a}, \overline{b} \in \mathcal{A}$, if $\mathcal{A} \models \varphi(\overline{a})$ and $\mathcal{A} \models \varphi(\overline{b})$, then $(\mathcal{A}, \overline{a}) \cong (\mathcal{A}, \overline{b})$.

The model theoretic perspective

However, there is no syntactic characterization for *just* computable categoricity, since the index set of all c.c. structures is Π^1_1 complete (Downey et al [3]).

The model theoretic perspective

However, there is no syntactic characterization for *just* computable categoricity, since the index set of all c.c. structures is Π^1_1 complete (Downey et al [3]).

Additionally, we have the following extra criteria for when a c.c. structure is also relatively c.c.

The model theoretic perspective

However, there is no syntactic characterization for *just* computable categoricity, since the index set of all c.c. structures is Π^1_1 complete (Downey et al [3]).

Additionally, we have the following extra criteria for when a c.c. structure is also relatively c.c.

Theorem (Goncharov [6])

If a structure is c.c. and its $\forall\exists$ theory is decidable, then it is relatively c.c.

The model theoretic perspective

However, there is no syntactic characterization for *just* computable categoricity, since the index set of all c.c. structures is Π^1_1 complete (Downey et al [3]).

Additionally, we have the following extra criteria for when a c.c. structure is also relatively c.c.

Theorem (Goncharov [6])

If a structure is c.c. and its $\forall\exists$ theory is decidable, then it is relatively c.c.

Corollary

If a structure is c.c. and its for $\forall\exists$ theory is decidable, then it has a formally Σ_1 Scott family.

Computable categoricity relative to a degree

A new relativization

We have the following newer relativization of computable categoricity.

A new relativization

We have the following newer relativization of computable categoricity.

Definition

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

A new relativization

We have the following newer relativization of computable categoricity.

Definition

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

Fact

A computable structure $\mathcal{A}$ is **relatively computably categorical** *if for all $X \in 2^{\mathbb{N}}$, $\mathcal{A}$ is c.c. relative to X.*

A new relativization

We have the following newer relativization of computable categoricity.

Definition

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

Fact

A computable structure $\mathcal{A}$ is **relatively computably categorical** *if for all $X \in 2^{\mathbb{N}}$, $\mathcal{A}$ is c.c. relative to X.*

Question: Are there structures where they are only c.c. relative to certain degrees **d**?

The cone above $0''$

We have the following fact.

The cone above $\mathbf{0}''$

We have the following fact.

Fact (Downey, Harrison-Trainor, Melnikov [2])

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.

The cone above $\mathbf{0}''$

We have the following fact.

Fact (Downey, Harrison-Trainor, Melnikov [2])

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_1^0 Scott family.

This implies that $\mathcal{A}$, as in the statement above, must be c.c. relative to all degrees above $\mathbf{0}''$.

The cone above $\mathbf{0}''$

If $\mathcal{A}$ is c.c. relative to some $\mathbf{d} \geq \mathbf{0}''$, then it is c.c. relative to *all* degrees above $\mathbf{0}''$.

The cone above $\mathbf{0}''$

If $\mathcal{A}$ is c.c. relative to some $\mathbf{d} \geq \mathbf{0}''$, then it is c.c. relative to *all* degrees above $\mathbf{0}''$.

The contrapositive also gives us that if $\mathcal{A}$ does not have a $\mathbf{0}''$-computable Σ^0_1 Scott family, then it is not c.c. relative to *any* $\mathbf{d} \geq \mathbf{0}''$.

If $\mathcal{A}$ is c.c. relative to some $\mathbf{d} \geq \mathbf{0}''$, then it is c.c. relative to *all* degrees above $\mathbf{0}''$.

The contrapositive also gives us that if $\mathcal{A}$ does not have a $\mathbf{0}''$-computable Σ^0_1 Scott family, then it is not c.c. relative to *any* $\mathbf{d} \geq \mathbf{0}''$.

So at $\mathbf{0}''$ and above, any computable structure $\mathcal{A}$ will settle on whether it is c.c. relative to all degrees or to none of them.

The cone above $\mathbf{0}''$

If $\mathcal{A}$ is c.c. relative to some $\mathbf{d} \geq \mathbf{0}''$, then it is c.c. relative to *all* degrees above $\mathbf{0}''$.

The contrapositive also gives us that if $\mathcal{A}$ does not have a $\mathbf{0}''$-computable Σ^0_1 Scott family, then it is not c.c. relative to *any* $\mathbf{d} \geq \mathbf{0}''$.

So at $\mathbf{0}''$ and above, any computable structure $\mathcal{A}$ will settle on whether it is c.c. relative to all degrees or to none of them.

Question: What happens between $\mathbf{0}$ and $\mathbf{0}''$?

In the c.e. degrees

The following is known regarding categoricity in the c.e. degrees.

The following is known regarding categoricity in the c.e. degrees.

Theorem (Downey, Harrison-Trainor, Melnikov [2])

There is a computable structure $\mathcal{A}$ and c.e. degrees
$\mathbf{0} = Y_0 <_T X_0 <_T Y_1 <_T X_1 <_T \ldots$ *such that*

In the c.e. degrees

The following is known regarding categoricity in the c.e. degrees.

Theorem (Downey, Harrison-Trainor, Melnikov [2])

There is a computable structure $\mathcal{A}$ and c.e. degrees
$\mathbf{0} = Y_0 <_T X_0 <_T Y_1 <_T X_1 <_T \dots$ *such that*

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

In the c.e. degrees

The following is known regarding categoricity in the c.e. degrees.

Theorem (Downey, Harrison-Trainor, Melnikov [2])

There is a computable structure $\mathcal{A}$ and c.e. degrees
$\mathbf{0} = Y_0 <_T X_0 <_T Y_1 <_T X_1 <_T \dots$ *such that*

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

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

In the c.e. degrees

The following is known regarding categoricity in the c.e. degrees.

Theorem (Downey, Harrison-Trainor, Melnikov [2])

There is a computable structure $\mathcal{A}$ and c.e. degrees $\mathbf{0} = Y_0 <_T X_0 <_T Y_1 <_T X_1 <_T \ldots$ such that

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

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

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

Partial orders of c.e. degrees

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

Partial orders of c.e. degrees

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

Theorem (V.)

Let $P = (P, \leq)$ be a countably infinite partially ordered set and suppose we partition P as $P = P_0 \sqcup P_1$. There exists a computable c.c. directed graph $\mathcal{G}$ and an embedding h of P into the c.e. degrees where $\mathcal{G}$ is c.c. relative to each degree in $h(P_0)$ and is not c.c. relative to each degree in $h(P_1)$.

Partial orders of c.e. degrees

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

Theorem (V.)

Let $P = (P, \leq)$ be a countably infinite partially ordered set and suppose we partition P as $P = P_0 \sqcup P_1$. There exists a computable c.c. directed graph $\mathcal{G}$ and an embedding h of P into the c.e. degrees where $\mathcal{G}$ is c.c. relative to each degree in $h(P_0)$ and is not c.c. relative to each degree in $h(P_1)$.

The construction for this result has four main goals:

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

Theorem (V.)

Let $P = (P, \leq)$ be a countably infinite partially ordered set and suppose we partition P as $P = P_0 \sqcup P_1$. There exists a computable c.c. directed graph $\mathcal{G}$ and an embedding h of P into the c.e. degrees where $\mathcal{G}$ is c.c. relative to each degree in $h(P_0)$ and is not c.c. relative to each degree in $h(P_1)$.

The construction for this result has four main goals: embedding P into the c.e. degrees via a map h,

Partial orders of c.e. degrees

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

Theorem (V.)

Let $P = (P, \leq)$ be a countably infinite partially ordered set and suppose we partition P as $P = P_0 \sqcup P_1$. There exists a computable c.c. directed graph $\mathcal{G}$ and an embedding h of P into the c.e. degrees where $\mathcal{G}$ is c.c. relative to each degree in $h(P_0)$ and is not c.c. relative to each degree in $h(P_1)$.

The construction for this result has four main goals: embedding P into the c.e. degrees via a map h, making the graph $\mathcal{G}$ c.c.,

Partial orders of c.e. degrees

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

Theorem (V.)

Let $P = (P, \leq)$ be a countably infinite partially ordered set and suppose we partition P as $P = P_0 \sqcup P_1$. There exists a computable c.c. directed graph $\mathcal{G}$ and an embedding h of P into the c.e. degrees where $\mathcal{G}$ is c.c. relative to each degree in $h(P_0)$ and is not c.c. relative to each degree in $h(P_1)$.

The construction for this result has four main goals: embedding P into the c.e. degrees via a map h, making the graph $\mathcal{G}$ c.c., making $\mathcal{G}$ c.c. relative to all degrees in $h(P_0)$,

Partial orders of c.e. degrees

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

Theorem (V.)

Let $P = (P, \leq)$ be a countably infinite partially ordered set and suppose we partition P as $P = P_0 \sqcup P_1$. There exists a computable c.c. directed graph $\mathcal{G}$ and an embedding h of P into the c.e. degrees where $\mathcal{G}$ is c.c. relative to each degree in $h(P_0)$ and is not c.c. relative to each degree in $h(P_1)$.

The construction for this result has four main goals: embedding P into the c.e. degrees via a map h, making the graph $\mathcal{G}$ c.c., making $\mathcal{G}$ c.c. relative to all degrees in $h(P_0)$, and finally, making $\mathcal{G}$ not c.c. relative to any degree in $h(P_1)$.

Notation

For $p \in P$, we build uniformly c.e. sets A_p.

Notation

For $p \in P$, we build uniformly c.e. sets A_p.

Definition

For $p \in P$, we define the c.e. set

$$D_p = \bigoplus_{q \leq p} A_q.$$

Notation

For $p \in P$, we build uniformly c.e. sets A_p.

Definition

For $p \in P$, we define the c.e. set

$$D_p = \bigoplus_{q \leq p} A_q.$$

Our embedding will be the map $h(p) = D_p$.

Notation

For $p \in P$, we build uniformly c.e. sets A_p.

Definition

For $p \in P$, we define the c.e. set

$$D_p = \bigoplus_{q \leq p} A_q.$$

Our embedding will be the map $h(p) = D_p$.

We also have the following notation for $p \in P$.

Definition

$$\overline{D_p} := \bigoplus_{q \neq p} A_q.$$

Notation

We use the following notation for graphs.

Notation

We use the following notation for graphs.

Definition

- $\mathcal{M}_e$ is the eth (partial) computable graph with domain ω where $E(x,y) \iff \Phi_e(x,y) = 1$ and $\neg E(x,y) \iff \Phi_e(x,y) = 0$.

Notation

We use the following notation for graphs.

Definition

- $\mathcal{M}_e$ is the eth (partial) computable graph with domain ω where $E(x,y) \iff \Phi_e(x,y) = 1$ and $\neg E(x,y) \iff \Phi_e(x,y) = 0$.
- $\mathcal{M}_i^{D_p}$ is the ith (partial) D_p-computable graph with domain ω where $E(x,y) \iff \Phi_i^{D_p}(x,y) = 1$ and $\neg E(x,y) \iff \Phi_i^{D_p}(x,y) = 0$.

Goals as requirements

We have the following requirements:

Goals as requirements

We have the following requirements:

- $N_e^p : \Phi_e^{\overline{D_p}} \neq A_p$,

Goals as requirements

We have the following requirements:

- $N_e^p : \Phi_e^{\overline{D_p}} \neq A_p$,
- S_e : if $\mathcal{G} \cong \mathcal{M}_e$, then there exists a computable isomorphism $f_e : \mathcal{G} \to \mathcal{M}_e$,

Goals as requirements

We have the following requirements:

- N_e^p : $\Phi_e^{\overline{D_p}} \neq A_p$,
- S_e : if $\mathcal{G} \cong \mathcal{M}_e$, then there exists a computable isomorphism $f_e : \mathcal{G} \to \mathcal{M}_e$,
- for $p \in P_0$, T_i^p : if $\mathcal{G} \cong \mathcal{M}_i^{D_p}$, then there exists a D_p-computable isomorphism $g_i^{D_p} : \mathcal{G} \to \mathcal{M}_i^{D_p}$,

Goals as requirements

We have the following requirements:

- $N_e^p : \Phi_e^{\overline{D_p}} \neq A_p$,
- S_e : if $\mathcal{G} \cong \mathcal{M}_e$, then there exists a computable isomorphism $f_e : \mathcal{G} \to \mathcal{M}_e$,
- for $p \in P_0$, T_i^p : if $\mathcal{G} \cong \mathcal{M}_i^{D_p}$, then there exists a D_p-computable isomorphism $g_i^{D_p} : \mathcal{G} \to \mathcal{M}_i^{D_p}$, and
- for $q \in P_1$, R_e^q : $\Phi_e^{D_q} : \mathcal{G} \to \mathcal{B}_q$ is not an isomorphism where $\mathcal{B}_q$ is a D_q-computable copy of $\mathcal{G}$ we build.

Goals as requirements

We have the following requirements:

- $N_e^p : \Phi_e^{\overline{D_p}} \neq A_p$,
- S_e : if $\mathcal{G} \cong \mathcal{M}_e$, then there exists a computable isomorphism $f_e : \mathcal{G} \to \mathcal{M}_e$,
- for $p \in P_0$, T_i^p : if $\mathcal{G} \cong \mathcal{M}_i^{D_p}$, then there exists a D_p-computable isomorphism $g_i^{D_p} : \mathcal{G} \to \mathcal{M}_i^{D_p}$, and
- for $q \in P_1$, R_e^q : $\Phi_e^{D_q} : \mathcal{G} \to \mathcal{B}_q$ is not an isomorphism where $\mathcal{B}_q$ is a D_q-computable copy of $\mathcal{G}$ we build.

The N_e^p requirements ensure that h is an embedding of P into the c.e. degrees.

Goals as requirements

We have the following requirements:

- $N_e^p : \Phi_e^{\overline{D_p}} \neq A_p$,
- S_e : if $\mathcal{G} \cong \mathcal{M}_e$, then there exists a computable isomorphism $f_e : \mathcal{G} \to \mathcal{M}_e$,
- for $p \in P_0$, T_i^p : if $\mathcal{G} \cong \mathcal{M}_i^{D_p}$, then there exists a D_p-computable isomorphism $g_i^{D_p} : \mathcal{G} \to \mathcal{M}_i^{D_p}$, and
- for $q \in P_1$, R_e^q : $\Phi_e^{D_q} : \mathcal{G} \to \mathcal{B}_q$ is not an isomorphism where $\mathcal{B}_q$ is a D_q-computable copy of $\mathcal{G}$ we build.

The N_e^p requirements ensure that h is an embedding of P into the c.e. degrees. The S_e requirements ensure that $\mathcal{G}$ is c.c.

Goals as requirements

We have the following requirements:

- $N_e^p : \Phi_e^{\overline{D_p}} \neq A_p$,
- S_e : if $\mathcal{G} \cong \mathcal{M}_e$, then there exists a computable isomorphism $f_e : \mathcal{G} \to \mathcal{M}_e$,
- for $p \in P_0$, T_i^p : if $\mathcal{G} \cong \mathcal{M}_i^{D_p}$, then there exists a D_p-computable isomorphism $g_i^{D_p} : \mathcal{G} \to \mathcal{M}_i^{D_p}$, and
- for $q \in P_1$, R_e^q : $\Phi_e^{D_q} : \mathcal{G} \to \mathcal{B}_q$ is not an isomorphism where $\mathcal{B}_q$ is a D_q-computable copy of $\mathcal{G}$ we build.

The N_e^p requirements ensure that h is an embedding of P into the c.e. degrees. The S_e requirements ensure that $\mathcal{G}$ is c.c. The T_i^p requirements ensure that $\mathcal{G}$ is c.c. relative to all degrees in $h(P_0)$.

Goals as requirements

We have the following requirements:

- $N_e^p : \Phi_e^{\overline{D_p}} \neq A_p$,
- S_e : if $\mathcal{G} \cong \mathcal{M}_e$, then there exists a computable isomorphism $f_e : \mathcal{G} \to \mathcal{M}_e$,
- for $p \in P_0$, T_i^p : if $\mathcal{G} \cong \mathcal{M}_i^{D_p}$, then there exists a D_p-computable isomorphism $g_i^{D_p} : \mathcal{G} \to \mathcal{M}_i^{D_p}$, and
- for $q \in P_1$, R_e^q : $\Phi_e^{D_q} : \mathcal{G} \to \mathcal{B}_q$ is not an isomorphism where $\mathcal{B}_q$ is a D_q-computable copy of $\mathcal{G}$ we build.

The N_e^p requirements ensure that h is an embedding of P into the c.e. degrees. The S_e requirements ensure that $\mathcal{G}$ is c.c. The T_i^p requirements ensure that $\mathcal{G}$ is c.c. relative to all degrees in $h(P_0)$. The R_e^q requirements ensure that $\mathcal{G}$ is not c.c. relative to any degree in $h(P_1)$.

Building $\mathcal{G}$ in stages

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

Building $\mathcal{G}$ in stages

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

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

Building $\mathcal{G}$ in stages

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

At stage $s = 0$, we set the domain of $\mathcal{G}$ 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 $\mathcal{G}$ in stages

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

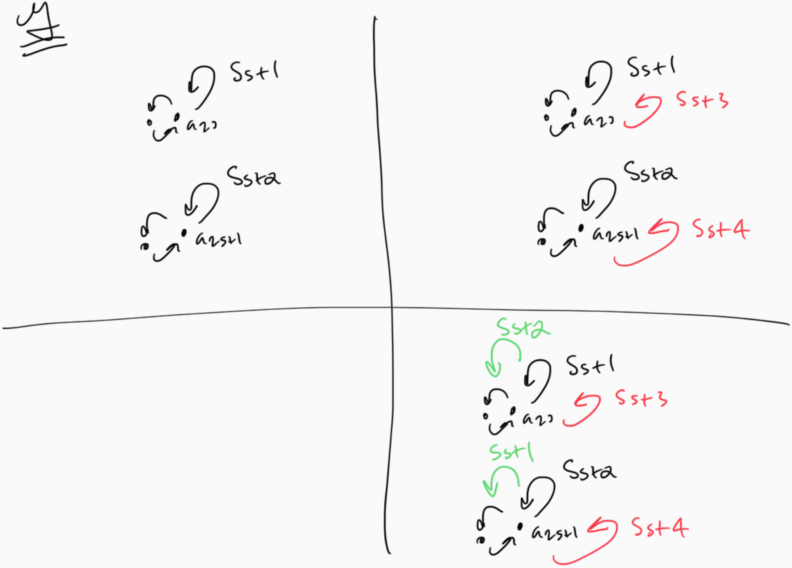
At stage $s = 0$, we set the domain of $\mathcal{G}$ to be empty.

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 $\mathcal{G}$ with its loops is the $2s$**th connected component** or just the $2s$th component of $\mathcal{G}$.

Configuration of loops in $\mathcal{G}$



Basic strategies: N_e^p

This is our basic strategy to satisfy all N_e^p for $p \in P$.

Basic strategies: N_e^p

This is our basic strategy to satisfy all N_e^p for $p \in P$.

Let s be the current stage of the construction and let α be an N_e^p-strategy.

1. If α is first eligible to act at stage s, it defines its witness x_α to be a large unused number.

Basic strategies: N_e^p

This is our basic strategy to satisfy all N_e^p for $p \in P$.

Let s be the current stage of the construction and let α be an N_e^p-strategy.

1. If α is first eligible to act at stage s, it defines its witness x_α to be a large unused number.
2. Check if $\Phi_e^{\overline{D_p}}(x_\alpha)[s]\downarrow= 0$ and keep x_α out of A_p.

Basic strategies: N_e^p

This is our basic strategy to satisfy all N_e^p for $p \in P$.

Let s be the current stage of the construction and let α be an N_e^p-strategy.

1. If α is first eligible to act at stage s, it defines its witness x_α to be a large unused number.
2. Check if $\Phi_e^{\overline{D_p}}(x_\alpha)[s] \downarrow = 0$ and keep x_α out of A_p. If not, α takes no action at stage s.

Basic strategies: N_e^p

This is our basic strategy to satisfy all N_e^p for $p \in P$.

Let s be the current stage of the construction and let α be an N_e^p-strategy.

1. If α is first eligible to act at stage s, it defines its witness x_α to be a large unused number.
2. Check if $\Phi_e^{\overline{D_p}}(x_\alpha)[s] \downarrow= 0$ and keep x_α out of A_p. If not, α takes no action at stage s. If so, α enumerates x_α into A_p and restrains $A_p \upharpoonright (\mathrm{use}(\Phi_e^{\overline{D_p}}(x_\alpha)) + 1)$.

Basic strategies: S_e

This is our basic strategy to satisfy all S_e requirements to make $\mathcal{G}$ c.c.

Basic strategies: S_e

This is our basic strategy to satisfy all S_e requirements to make $\mathcal{G}$ c.c.

Let s be the current stage of the construction and let α be an S_e-strategy.

1. If α is first eligible to act at stage s, it sets its parameter $n_\alpha = 0$. It looks for copies in $\mathcal{M}_e[s]$ of the $2n_\alpha$th and $(2n_\alpha + 1)$st components of $\mathcal{G}[s]$. It defines $f_\alpha[s]$ to be the empty map.

Basic strategies: S_e

This is our basic strategy to satisfy all S_e requirements to make $\mathcal{G}$ c.c.

Let s be the current stage of the construction and let α be an S_e-strategy.

1. If α is first eligible to act at stage s, it sets its parameter $n_\alpha = 0$. It looks for copies in $\mathcal{M}_e[s]$ of the $2n_\alpha$th and $(2n_\alpha + 1)$st components of $\mathcal{G}[s]$. It defines $f_\alpha[s]$ to be the empty map.
2. If n_α is defined and $f_\alpha[s-1]$ is defined for all $m < n_\alpha$, α looks for copies of the $2n_\alpha$th and $(2n_\alpha + 1)$st components of $\mathcal{G}[s]$.

3. If no copies of the $2n_\alpha$th and $(2n_\alpha + 1)$st components are found, α takes no additional action at stage s, retains the value of n_α, and sets $f_\alpha[s] = f_\alpha[s-1]$.

Basic strategies: S_e

3. If no copies of the $2n_\alpha$th and $(2n_\alpha + 1)$st components are found, α takes no additional action at stage s, retains the value of n_α, and sets $f_\alpha[s] = f_\alpha[s-1]$. If copies are found, α extends $f_\alpha[s-1]$ to $f_\alpha[s]$ by matching the components in $\mathcal{G}[s]$ to the copies found in $\mathcal{M}_e[s]$ and increments n_α by 1.

Basic strategies: T_i^p

Let $p \in P_0$. Our basic strategy to satisfy all T_i^p requirements to make $\mathcal{G}$ c.c. relative to D_p is similar to our S_e-strategy. Let α be a T_i^p-strategy.

Basic strategies: T_i^p

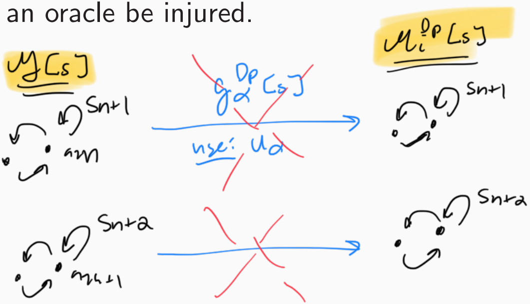
Let $p \in P_0$. Our basic strategy to satisfy all T_i^p requirements to make $\mathcal{G}$ c.c. relative to D_p is similar to our S_e-strategy. Let α be a T_i^p-strategy.

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

Basic strategies: T_i^p

Let $p \in P_0$. Our basic strategy to satisfy all T_i^p requirements to make $\mathcal{G}$ c.c. relative to D_p is similar to our S_e-strategy. Let α be a T_i^p-strategy.

For each n, we try to find copies of the $2n$th and $(2n+1)$st components of $\mathcal{G}$ in $\mathcal{M}_i^{D_p}$. But now because D_p is a c.e. set, loops in $\mathcal{M}_i^{D_p}$ can be injured or embeddings using a finite part of D_p as an oracle be injured.



if at a later stage, a # $x < u_\alpha$ enters D_p, then $g_\alpha^{D_p}[s]$ will disappear on the $2n$th and $(2n+1)$st components of $\mathcal{G}$.

Basic strategies: T_i^p

When α is next eligible to act at stage s, it will check if $D_p[t] \neq D_p[s]$ where t is the previous α-stage.

Basic strategies: T_i^p

When α is next eligible to act at stage s, it will check if $D_p[t] \neq D_p[s]$ where t is the previous α-stage.

If $D_p[t] \neq D_p[s]$, then α will update its parameter n_α accordingly depending on what type of injury occurred.

When α is next eligible to act at stage s, it will check if $D_p[t] \neq D_p[s]$ where t is the previous α-stage.

If $D_p[t] \neq D_p[s]$, then α will update its parameter n_α accordingly depending on what type of injury occurred. Otherwise, it will proceed to try and match the $2n_\alpha$th and $(2n_\alpha + 1)$st components of $\mathcal{G}$ for the n_α parameter it had at the beginning of stage s.

Basic strategies: R_e^q

Finally, for $q \in P_1$, we do the following to satisfy all R_e^q requirements to make $\mathcal{G}$ not c.c. relative to D_q.

Basic strategies: R_e^q

Finally, for $q \in P_1$, we do the following to satisfy all R_e^q requirements to make $\mathcal{G}$ not c.c. relative to D_q.

We will build a D_q-computable graph $\mathcal{B}_q$ which is isomorphic to $\mathcal{G}$ in stages, similarly to how we built $\mathcal{G}$.

Basic strategies: R_e^q

Finally, for $q \in P_1$, we do the following to satisfy all R_e^q requirements to make $\mathcal{G}$ not c.c. relative to D_q.

We will build a D_q-computable graph $\mathcal{B}_q$ which is isomorphic to $\mathcal{G}$ in stages, similarly to how we built $\mathcal{G}$. At stage $s = 0$, let $\mathcal{B}_q = \emptyset$.

Finally, for $q \in P_1$, we do the following to satisfy all R_e^q requirements to make $\mathcal{G}$ not c.c. relative to D_q.

We will build a D_q-computable graph $\mathcal{B}_q$ which is isomorphic to $\mathcal{G}$ in stages, similarly to how we built $\mathcal{G}$. At stage $s = 0$, let $\mathcal{B}_q = \emptyset$. At stage $s > 0$, add two new root nodes b_{2s}^q and b_{2s+1}^q and attach to each one a 2-loop.

Basic strategies: R_e^q

Finally, for $q \in P_1$, we do the following to satisfy all R_e^q requirements to make $\mathcal{G}$ not c.c. relative to D_q.

We will build a D_q-computable graph $\mathcal{B}_q$ which is isomorphic to $\mathcal{G}$ in stages, similarly to how we built $\mathcal{G}$. At stage $s = 0$, let $\mathcal{B}_q = \emptyset$. At stage $s > 0$, add two new root nodes b_{2s}^q and b_{2s+1}^q and attach to each one a 2-loop. Attach a $(5s+1)$-loop to b_{2s}^q and a $(5s+2)$-loop to b_{2s+1}^q.

$\underline{\mathcal{B}_q}$

Basic strategies: R_e^q

This is our diagonalization strategy to satisfy all R_e^q.

Basic strategies: R_e^q

This is our diagonalization strategy to satisfy all R_e^q.

Let s be the current stage of the construction and let α be an R_e^q-strategy.

1. If α is first eligible to act at stage s, it defines its parameter n_α to be a large unused number.

Basic strategies: R_e^q

This is our diagonalization strategy to satisfy all R_e^q.

Let s be the current stage of the construction and let α be an R_e^q-strategy.

1. If α is first eligible to act at stage s, it defines its parameter n_α to be a large unused number.
2. α checks if $\Phi_e^{D_q}[s]$ maps the $2n_\alpha$th and $(2n_\alpha + 1)$st components of $\mathcal{G}[s]$ to the corresponding copies in $\mathcal{M}_e^{D_q}[s]$.

Basic strategies: R_e^q

This is our diagonalization strategy to satisfy all R_e^q.

Let s be the current stage of the construction and let α be an R_e^q-strategy.

1. If α is first eligible to act at stage s, it defines its parameter n_α to be a large unused number.
2. α checks if $\Phi_e^{D_q}[s]$ maps the $2n_\alpha$th and $(2n_\alpha + 1)$st components of $\mathcal{G}[s]$ to the corresponding copies in $\mathcal{M}_e^{D_q}[s]$. If not, α takes no further action.

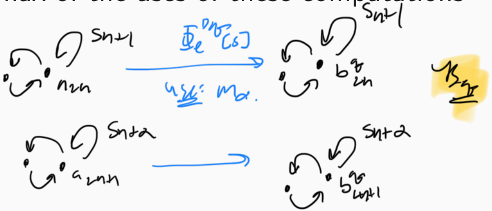
Basic strategies: R_e^q

This is our diagonalization strategy to satisfy all R_e^q.

Let s be the current stage of the construction and let α be an R_e^q-strategy.

1. If α is first eligible to act at stage s, it defines its parameter n_α to be a large unused number.
2. α checks if $\Phi_e^{D_q}[s]$ maps the $2n_\alpha$th and $(2n_\alpha + 1)$st components of $\mathcal{G}[s]$ to the corresponding copies in $\mathcal{M}_e^{D_q}[s]$. If not, α takes no further action. If α sees such a computation, it defines m_α to be the max of the uses of these computations and restrains $D_q \upharpoonright m_\alpha$.

$\mathcal{M}$



Basic strategies: R_e^q

3. α attaches a $(5n+3)$-loop to a_{2n} and b_{2n}^q and a $(5n+4)$-loop to a_{2n+1} and b_{2n+1}^q.

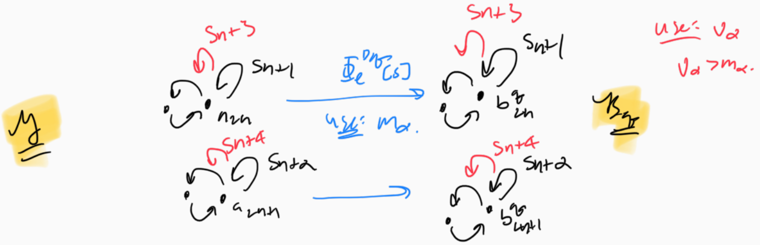
Basic strategies: R_e^q

3. α attaches a $(5n+3)$-loop to a_{2n} and b_{2n}^q and a $(5n+4)$-loop to a_{2n+1} and b_{2n+1}^q. Let v_α be the use associated with these loops appearing in $\mathcal{B}_q$. Note that $v_\alpha > m_\alpha$.

Basic strategies: R_e^q

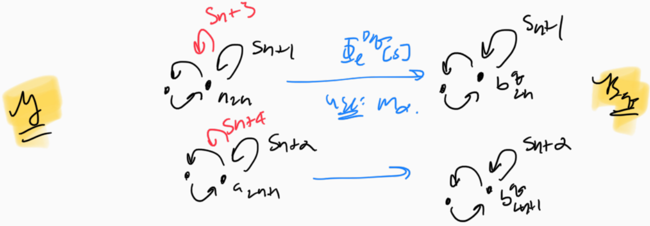
3. α attaches a $(5n+3)$-loop to a_{2n} and b^q_{2n} and a $(5n+4)$-loop to a_{2n+1} and b^q_{2n+1}. Let v_α be the use associated with these loops appearing in $\mathcal{B}_q$. Note that $v_\alpha > m_\alpha$.
4. α now issues a challenge to all higher priority requirements which are S_e and T^p_i:

3. α attaches a $(5n+3)$-loop to a_{2n} and b^q_{2n} and a $(5n+4)$-loop to a_{2n+1} and b^q_{2n+1}. Let v_α be the use associated with these loops appearing in $\mathcal{B}_q$. Note that $v_\alpha > m_\alpha$.
4. α now issues a challenge to all higher priority requirements which are S_e and T^p_i: they must now extend their embeddings, if possible, to include these new loops.



Basic strategies: R_e^q

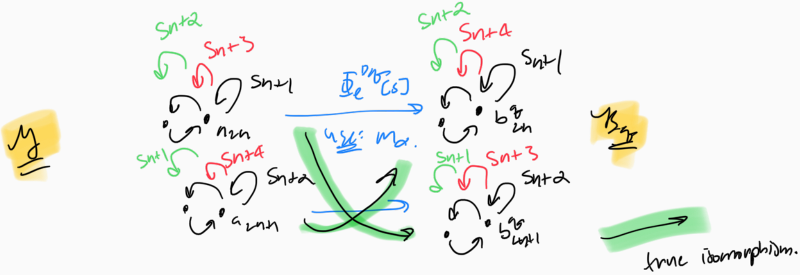
5. If all higher S_e and T_i^p requirements can meet this challenge and α becomes eligible to act again at a later stage, it enumerates v_α into A_q. This makes the $(5n+3)$- and $(5n+4)$-loops in $\mathcal{B}_q$ disappear.



6. α reattaches a $(5n+3)$-loop to b^q_{2n+1} and a $(5n+4)$-loop to b^q_{2n}. It also attaches a $(5n+1)$-loop to a_{2n+1} and to b^q_{2n+1}, and a $(5n+2)$-loop to a_{2n} and to b^q_{2n}.

6. α reattaches a $(5n+3)$-loop to b^q_{2n+1} and a $(5n+4)$-loop to b^q_{2n}. It also attaches a $(5n+1)$-loop to a_{2n+1} and to b^q_{2n+1}, and a $(5n+2)$-loop to a_{2n} and to b^q_{2n}.

Our final configuration of loops in $\mathcal{B}_q$ is now:



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 R_e^q-strategy want to diagonalize while the S_e and T_i^p-strategies want to build embeddings:

Interactions between strategies

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

Interaction 1

The R_e^q-strategy want to diagonalize while the S_e and T_i^p-strategies want to build embeddings: this was resolved by having R_e^q "wait" for higher priority S_e and T_i^p requirements and the homogenizing part of step 6 in the R_e^q-strategy.

Interactions between strategies

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

Interaction 1

The R_e^q-strategy want to diagonalize while the S_e and T_i^p-strategies want to build embeddings: this was resolved by having R_e^q "wait" for higher priority S_e and T_i^p requirements and the homogenizing part of step 6 in the R_e^q-strategy.

Interaction 2

The N_e^p-strategy enumerating numbers into A_p to achieve independence of degrees:

Interactions between strategies

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

Interaction 1

The R_e^q-strategy want to diagonalize while the S_e and T_i^p-strategies want to build embeddings: this was resolved by having R_e^q "wait" for higher priority S_e and T_i^p requirements and the homogenizing part of step 6 in the R_e^q-strategy.

Interaction 2

The N_e^p-strategy enumerating numbers into A_p to achieve independence of degrees: this is resolved on a tree of strategies and by letting T_i^p check for any changes in D_p up to a finite part each stage.

Interactions between strategies

The last important interaction comes from the poset ordering on P.

The last important interaction comes from the poset ordering on P.

Interaction 3

An R_e^q-strategy β and a T_i^p-strategy α when $q < p$ in P and T_i^p is of higher priority than R_e^q:

The last important interaction comes from the poset ordering on P.

Interaction 3

An R^q_e-strategy β and a T^p_i-strategy α when $q < p$ in P and T^p_i is of higher priority than R^q_e: the T^p_i-strategy needs an additional step for when it is challenged to enumerate any uses associated to the $2n_\beta$th and $(2n_\beta + 1)$st components of $\mathcal{G}$ into A_p. This lets us lift uses for T^p_i.

Thank You

Thanks for attending my talk! I'd be happy to answer any questions.

References

[1] Chris Ash et al. **"Generic copies of countable structures".** *APAL* 42.3 (1989), pp. 195–205.

[2] Rodney Downey, Matthew Harrison-Trainor, and Alexander Melnikov. **"Relativizing computable categoricity".** *PAMS* 149.9 (2021), pp. 3999–4013.

[3] Rodney G. Downey et al. **"The complexity of computable categoricity".** *Advances in Mathematics* 268 (2015), pp. 423–466.

[4] Ju. L. Erš. **"Theorie Der Numerierungen III".** *MLQ* 23.19-24 (1977), pp. 289–371. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/malq.19770231902`.

[5] Sergey S. Goncharov, Steffen Lempp, and Reed Solomon. **“The computable dimension of ordered abelian groups”.** *Advances in Mathematics* 175.1 (2003), pp. 102–143.

[6] S. S. Gončarov. **“The problem of the number of nonautoequivalent constructivizations”.** *Algebra i Logika* 19.6 (1980), pp. 621–639, 745.

[7] J. B. Remmel. **“Recursively Categorical Linear Orderings”.** *PAMS* 83.2 (1981), pp. 387–391. (Visited on 10/10/2022).