Computable Structures and Isomorphisms

Jonny Stephenson
Joint work with B. Csima and M. Deveau

Valparaiso University

June 4, 2018

Background

Definition

A computable structure $\mathcal{A}$ consists of a computable domain, and computable relations and functions.

In this talk, we focus on linear orders.

Definition

A structure $\mathcal{A}$ is $\mathbf{c}$-computably categorical if $\mathbf{c}$ can compute an isomorphism between any two computable copies of $\mathcal{A}$.
The degree of categoricity of the structure $\mathcal{A}$ is $\mathbf{d}$ if $\mathcal{A}$ is $\mathbf{c}$-computably categorical precisely when $\mathbf{d} \leq_T \mathbf{c}$.

A standard example

We consider $\mathbb{N}$ as a linear order:

Example

Isomorphisms between computable copies of the linear order $\mathbb{N}$ are always $\mathbf{0}'$-computable: you just need to know the least element and the successor relation.

There are computable copies with isomorphism of Turing degree $\mathbf{0}'$ between them. One typical approach is to use predetermined elements as coding locations.

So $\mathbf{0}'$ is the degree of categoricity.

On closer inspection

We build two copies $\mathcal{A}$ and $\mathcal{B}$, with isomorphism $f : \mathcal{A} \to \mathcal{B}$.

There's a computable subset U of coding locations in $\mathcal{A}$ for which $f(U)$ can compute $\mathbf{0}'$, and hence f.

A common phenomenon?

We arranged that $f(U) \equiv_T f$ for a computable subset U of $\mathcal{A}$

Note that $f(U) \leq_T f$ is always true for computable U. It is the other inequality that is interesting.

This is a technique that comes up quite often: we can use it to show that a **d**-categorical structure actually has degree of categoricity **d**.

We (Csima, Deveau, S.) asked:

Question

Is it generally true that if $f : \mathcal{A} \to \mathcal{B}$ is an isomorphism of computable structures, then $f \equiv_T f(U)$ for some computable subset U of $\mathcal{A}$?
If not, when is it true?

Note that we need to restrict U to be unary for this to be interesting: for any structure, we can easily build a binary relation whose image encodes the isomorphism in a *very* uniform way.

Example

Theorem

There are computable copies $\mathcal{A}$ and $\mathcal{B}$ of the natural numbers with isomorphism $f : \mathcal{A} \to \mathcal{B}$ and such that:
There is no computable subset U of $\mathcal{A}$ with $f(U) \equiv_T f$.
There is no computable subset U of $\mathcal{B}$ with $f^{-1}(U) \equiv_T f$.
Furthermore, $f \equiv_T \mathbf{0}'$.

The main kind of requirements look like $\Phi_e^{f(U)} \neq f$ and $\Phi_e^{f^{-1}(U)} \neq f$ for computable U.

The basic strategy for meeting the requirement $\Phi_e^{f(U)} \neq f$ is to pick a witness x and try to make $\Phi_e^{f(U)}(x) \neq f(x)$.

If we are threatened by a computation $\Phi_e^{f(U)}(x) = f(x)$ during the construction, insert a predecessor to $f(x)$ in $\mathcal{B}$.

Problem: this disrupts the use of the computation, which relied on certain elements of $\mathcal{B}$ being in/out of $f(U)$!

To restore the use, we need to realign the isomorphism. This is possible if U is both infinite and coinfinite (see arm-waving and blackboard).

If U is finite or cofinite, so is $f(U)$, which gives us a win (since we make f noncomputable).

The previous theorem *requires* that we construct two new copies of the natural numbers, rather than working with the usual copy $\mathcal{N}$.

Theorem

Let $\mathcal{A}$ be a computable copy of the natural numbers. Let $f : \mathcal{A} \to \mathcal{N}$ be the isomorphism between them.
Then there is a computable subset U of $\mathcal{A}$ with $f(U) \equiv_T f$.

but on the other hand:

Theorem

There is a computable copy $\mathcal{B}$ of the natural numbers such that if $f : \mathcal{N} \to \mathcal{B}$ is the isomorphism between them, then for no computable U does $f(U) \equiv_T f$.

(Just do half of the requirements...)

So there is a fundamental asymmetry arising from the *direction* of the isomorphism.

Why do we get this asymmetry?

Building a decidable copy of the natural numbers means that we can't put elements in out of order to code.

Could this somehow be a phenomenon about mapping into vs out of a decidable structure?

We investigated the case of the order type ω^2, to see how isomorphisms into the usual decidable copy $\mathcal{N}^2$ behave.

Theorem

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

The technique is similar to the one used for the natural numbers. Still finite injury, but a slightly nastier process.
It is possible to make the isomorphism from this theorem of degree $\mathbf{0}''$, which is the degree of categoricity, but this turns the construction into an infinite injury argument...

Further thoughts

Our examples are all cases in which the degree of categoricity can be confirmed by looking at unary relations, but we've shown that we at least need to be careful about which copies we look at.

Question

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

Question

Can the underlying assumption of rigidity be dropped?

Question

Is there a typical behaviour in any sense?