Instant structures and categoricity

Keng Meng Ng
(joint work with Alexander Melnikov and Iskander Kalimullin)

Nanyang Technological University, Singapore

August 2015

Motivating questions

- Study how computation interacts with various mathematical concepts.
- Complexity of constructions and objects we use in mathematics (how to calibrate?)
- Can formalize this more syntactically (reverse math, etc).
- Or more model theoretically...

Motivating questions

- Study how computation interacts with various mathematical concepts.
- Complexity of constructions and objects we use in mathematics (how to calibrate?)
- Can formalize this more syntactically (reverse math, etc).
- Or more model theoretically...

Motivating questions I: Presentations

- In computable model / structure theory, can different effective concepts
  - presentations of a structure,
  - complexity of isomorphisms within an isomorphism type,
  - investigations can descend into a more degree-theoretic approach.
- For instance, classically, given any structure $\mathcal{A}$, a *copy* or a *presentation* is simply $\mathcal{B} = (\mathrm{dom}(\mathcal{B}), R^{\mathcal{B}}, f^{\mathcal{B}}, \cdots)$ such that $\mathcal{B} \cong \mathcal{A}$.
- If $\mathcal{A}$ is countable and the language is computable, then this allows us to talk about $\deg(\mathcal{B})$.
- A countable $\mathcal{A}$ can have presentations of different Turing degrees, so it's not easy to define the "Turing degree" of a structure.

Motivating questions I: Presentations

- In computable model / structure theory, can different effective concepts
  - presentations of a structure,
  - complexity of isomorphisms within an isomorphism type,
  - investigations can descend into a more degree-theoretic approach.
- For instance, classically, given any structure $\mathcal{A}$, a *copy* or a *presentation* is simply $\mathcal{B} = (\mathrm{dom}(\mathcal{B}), R^{\mathcal{B}}, f^{\mathcal{B}}, \cdots)$ such that $\mathcal{B} \cong \mathcal{A}$.
- If $\mathcal{A}$ is countable and the language is computable, then this allows us to talk about $\deg(\mathcal{B})$.
- A countable $\mathcal{A}$ can have presentations of different Turing degrees, so it's not easy to define the "Turing degree" of a structure.

Motivating questions I: Presentations

- In computable model / structure theory, can different effective concepts
  - presentations of a structure,
  - complexity of isomorphisms within an isomorphism type,
  - investigations can descend into a more degree-theoretic approach.
- For instance, classically, given any structure $\mathcal{A}$, a *copy* or a *presentation* is simply $\mathcal{B} = (\mathrm{dom}(\mathcal{B}), R^{\mathcal{B}}, f^{\mathcal{B}}, \cdots)$ such that $\mathcal{B} \cong \mathcal{A}$.
- If $\mathcal{A}$ is countable and the language is computable, then this allows us to talk about $\deg(\mathcal{B})$.
- A countable $\mathcal{A}$ can have presentations of different Turing degrees, so it's not easy to define the "Turing degree" of a structure.

Motivating questions I: Presentations

- In computable model / structure theory, can different effective concepts
  - presentations of a structure,
  - complexity of isomorphisms within an isomorphism type,
  - investigations can descend into a more degree-theoretic approach.
- For instance, classically, given any structure $\mathcal{A}$, a *copy* or a *presentation* is simply $\mathcal{B} = (\mathrm{dom}(\mathcal{B}), R^{\mathcal{B}}, f^{\mathcal{B}}, \cdots)$ such that $\mathcal{B} \cong \mathcal{A}$.
- If $\mathcal{A}$ is countable and the language is computable, then this allows us to talk about deg($\mathcal{B}$).
- A countable $\mathcal{A}$ can have presentations of different Turing degrees, so it's not easy to define the "Turing degree" of a structure.

Motivating questions II: Complexity of Isomorphisms

- So one way of measuring precisely the complexity of a structure $\mathcal{A}$ is to look at
$$\mathit{Spec}(\mathcal{A}) = \{\deg(\mathcal{B}) \mid \mathcal{B} \cong \mathcal{A}\}.$$
- This gives a finer analysis (classically indistinguishable).
- Extensive study of degree spectra.
- Classically $\mathcal{A}$ and $\mathcal{B}$ are considered the same if $\mathcal{A} \cong \mathcal{B}$.
- However, from an effective point of view, even if $\mathcal{A} \cong \mathcal{B}$ are computable, they may have very different “hidden" effective properties.
- Standard example: $(\omega, <) \cong \mathcal{A}$ where you arrange for $2n$ and $2n+2$ to be adjacent in $\mathcal{A}$ iff $n \in \emptyset'$.

Motivating questions II: Complexity of Isomorphisms

- So one way of measuring precisely the complexity of a structure $\mathcal{A}$ is to look at
$$Spec(\mathcal{A}) = \{\deg(\mathcal{B}) \mid \mathcal{B} \cong \mathcal{A}\}.$$
- This gives a finer analysis (classically indistinguishable).
- Extensive study of degree spectra.
- Classically $\mathcal{A}$ and $\mathcal{B}$ are considered the same if $\mathcal{A} \cong \mathcal{B}$.
- However, from an effective point of view, even if $\mathcal{A} \cong \mathcal{B}$ are computable, they may have very different “hidden" effective properties.
- Standard example: $(\omega, <) \cong \mathcal{A}$ where you arrange for $2n$ and $2n + 2$ to be adjacent in $\mathcal{A}$ iff $n \in \emptyset'$.

Motivating questions II: Complexity of Isomorphisms

- In the standard example $(\omega, <) \cong \mathcal{A}$, “successivity" was the hidden property. Any isomorphism must transfer all definable properties, so this says that...
- This gives another way of defining precisely the complexity of a structure:

  The degree of categoricity of $\mathcal{A}$ is the least degree **d** *such that* **d** *computes an isomorphism between any two copies of $\mathcal{A}$.*

- Again, gives a finer analysis of classically indistinguishable concepts.

Motivating questions II: Complexity of Isomorphisms

- In the standard example $(\omega, <) \cong \mathcal{A}$, “successivity" was the hidden property. Any isomorphism must transfer all definable properties, so this says that...
- This gives another way of defining precisely the complexity of a structure:

  > *The degree of categoricity of $\mathcal{A}$ is the least degree* ***d*** *such that* ***d*** *computes an isomorphism between any two copies of $\mathcal{A}$.*

- Again, gives a finer analysis of classically indistinguishable concepts.

Motivating questions

- Often it is better to look at *computable* isomorphisms, i.e.

Definition

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

- Aim of the project: Systematic approach to all these considerations, with even stricter / finer effective restrictions.

Introduction

Definition (Mal'cev, Rabin, 60's)

A structure is computable if it has domain $\mathbb{N}$ and all operations and relations are uniformly computable.

- Equivalent variations (allow domain to be computable or c.e.).
- Seen to unify all earlier effective algebraic concepts, e.g. explicitly presented fields, recursively presented group with solvable word problem, etc.
- This has grown since into a large body of research;

  groups, fields, Boolean algebras, linear orders, model theory, reverse mathematics.

Introduction

- Our investigation is to place even finer restrictions:

  When does a computable structure have a feasible presentation?

- One way: structure presented by a *finite automaton* (won't discuss here).
- Another way: structure presented (as usual) by a Turing machine, but with restricted time complexity.
  - Most popular notion: polynomial time structures (Cenzer, Remmel, Downey).
  - Of course this depends on how the domain is represented (as $\mathbb{N}$ or $2^{<\omega}$).

Introduction

- Our investigation is to place even finer restrictions:

  When does a computable structure have a feasible presentation?

- One way: structure presented by a *finite automaton* (won't discuss here).
- Another way: structure presented (as usual) by a Turing machine, but with restricted time complexity.
  - Most popular notion: polynomial time structures (Cenzer, Remmel, Downey).
  - Of course this depends on how the domain is represented (as $\mathbb{N}$ or $2^{<\omega}$).

Introduction

- Again, there's a large body of work (80's) done on polynomial time (mostly) algebras.
- Our starting point is a series of papers of Cenzer, Remmel (and other co-authors), on various classes of "feasible" structures.
- In computable structures we allow algorithms to be extremely inefficient.
- Sometimes, every computable structure has a polynomial-time copy:

  Linear orders, certain kinds of BAs, some commutative groups.
  In many cases, proofs are focussed on first making structure "primitive recursive", then getting poly-time for free.

Introduction

- Again, there's a large body of work (80's) done on polynomial time (mostly) algebras.
- Our starting point is a series of papers of Cenzer, Remmel (and other co-authors), on various classes of "feasible" structures.
- In computable structures we allow algorithms to be extremely inefficient.
- Sometimes, every computable structure has a polynomial-time copy:

  Linear orders, certain kinds of BAs, some commutative groups.
  In many cases, proofs are focussed on first making structure "primitive recursive", then getting poly-time for free.

Introduction

- In the negative direction, to show a structure has no polynomial time copy, it's easier to argue it has no primitive recursive copy.

Definition (Cenzer, Remmel)

A structure is primitive recursive if it's domain, operations and relations are all primitive recursive.

- *Not that different from being computable*: For instance, even if $\mathcal{A}$ has a primitive recursive copy, new elements can be enumerated very slowly.
- (Alaev) Every computable locally finite structure has a primitive recursive copy.

Introduction

- In the negative direction, to show a structure has no polynomial time copy, it's easier to argue it has no primitive recursive copy.

Definition (Cenzer, Remmel)

A structure is primitive recursive if it's domain, operations and relations are all primitive recursive.

- *Not that different from being computable*: For instance, even if $\mathcal{A}$ has a primitive recursive copy, new elements can be enumerated very slowly.
- (Alaev) Every computable locally finite structure has a primitive recursive copy.

Introduction

- Instead, we will focus on structures with no possible way to delay revealing the structure:

Definition

A structure is instant if it has domain $\mathbb{N}$, and all operations and relations are primitive recursive (on $\mathbb{N}$).

- We only consider finite languages.
- Already used by Cenzer and Remmel as a technical tool.
- We will instead: systematic study of instant versus computable.
- Intuition: Instant structures have to decide right away what to do with the next element. (Cannot pass from subset to $\mathbb{N}$).

Introduction

- Instead, we will focus on structures with no possible way to delay revealing the structure:

Definition

A structure is instant if it has domain $\mathbb{N}$, and all operations and relations are primitive recursive (on $\mathbb{N}$).

- We only consider finite languages.
- Already used by Cenzer and Remmel as a technical tool.
- We will instead: systematic study of instant versus computable.
- Intuition: Instant structures have to decide right away what to do with the next element. (Cannot pass from subset to $\mathbb{N}$).

Introduction

- We can place effectivity on math structures in two ways. In the same vein, we can ask:

Question (1)

When does a computable structure have an instant copy?

Question (2)

How many instant copies does an instant structure have, up to instant isomorphisms?

- We contrast to the computable case; often different, sometimes even unclear.
- Reveals how “reticent or forth-coming" a structure is.

Introduction

- We can place effectivity on math structures in two ways. In the same vein, we can ask:

Question (1)

When does a computable structure have an instant copy?

Question (2)

How many instant copies does an instant structure have, up to instant isomorphisms?

- We contrast to the computable case; often different, sometimes even unclear.
- Reveals how “reticent or forth-coming" a structure is.

Introduction

- We can place effectivity on math structures in two ways. In the same vein, we can ask:

Question (1)

When does a computable structure have an instant copy?

Question (2)

How many instant copies does an instant structure have, up to instant isomorphisms?

- We contrast to the computable case; often different, sometimes even unclear.
- Reveals how “reticent or forth-coming" a structure is.

When does a structure have an instant copy?

Theorem (Kalimullin, Melnikov, N)

Each computable structure in the following classes has an instant copy:

- *Equivalence structures,*
- *linear orders,*
- *torsion-free abelian groups,*
- *boolean algebras,*
- *abelian p-groups.*

Proof.

Each of these structures possesses a certain amount of reticence. Allows us to indefinitely delay without having to commit to anything important. □

When does a structure have an instant copy?

Proof.

- Let's discuss equivalence structures, simplest example.
- Take a computable equivalence structure $\mathcal{A}$ with infinitely many distinct classes.
- We build instant $\mathcal{B}$ such that $\mathcal{B} \cong \mathcal{A}$.
- $\mathcal{B}$ is instant: By stage s we have declared the relations between $\{0, \cdots, s\}$.
- The isomorphism between $\mathcal{A}$ and $\mathcal{B}$ is computable, but not primitive recursive.

$\square$

When does a structure have an instant copy?

- The classes above all have a “computable" basis if some sort, which is used for delaying when building an instant copy.
- However, this is not sufficient to ensure the existence of an instant copy:

Theorem (Cenzer, Remmel, KMN)

There is a computable torsion abelian group with no instant copy.

Question

- *Find a reasonable sufficient condition for the existence of an instant copy.*
- *Formalization of a “basis" of some sort, which can by used for delaying.*

When does a structure have an instant copy?

- The classes above all have a “computable" basis if some sort, which is used for delaying when building an instant copy.
- However, this is not sufficient to ensure the existence of an instant copy:

Theorem (Cenzer, Remmel, KMN)

There is a computable torsion abelian group with no instant copy.

Question

- *Find a reasonable sufficient condition for the existence of an instant copy.*
- *Formalization of a “basis" of some sort, which can by used for delaying.*

When does a structure have an instant copy?

Theorem (Cenzer, Remmel, KMN)

There is a computable torsion abelian group with no instant copy.

Proof.

- Take $\mathcal{A} = \bigoplus_{p \in S} \mathbb{Z}_p$, for some infinite c.e. set S of primes.
- A diagonalization strategy making $\mathcal{A} \ncong \mathcal{P}_e$ works as follows:
  - Take $a_e \in \mathcal{P}_e$ and instantly generate $a_e,\ 2a_e,\ 3a_e,\ 4a_e, \cdots$.
  - If we see that $ma_e \neq 0$ then we know it is safe to put small primes $p < \sqrt{m}$ into S.
  - If we see $ma_e = 0$ then we avoid putting primes $\sqrt{m} < p < m$ into S.
  - If a_e has infinite order then $\mathcal{A} \ncong \mathcal{P}_e$, and we work below $m \to \infty$. $\square$

When does a structure have an instant copy?

Remark about proof:

When satisfying $\mathcal{A} \not\cong \mathcal{P}_e$ above, there are two outcomes:

$$\Pi^0_2 \text{ } (a_e \text{ has infinite order}), \text{ and } \Sigma^0_2 \text{ } (a_e \text{ has finite order}).$$

The total nature of $\mathcal{P}_e$ allows us to reduce the guesses to

$$\Pi^0_2 \text{ (infinite order) versus } \Sigma^0_1 \text{ (finite order).}$$

- (Kalimullin, Melnikov, N) There is a computable ordered (Archimedean) abelian group without an instant copy.

When does a structure have an instant copy?

Remark about proof:

When satisfying $\mathcal{A} \not\cong \mathcal{P}_e$ above, there are two outcomes:

$$\Pi^0_2 \text{ (}a_e\text{ has infinite order), and } \Sigma^0_2 \text{ (}a_e\text{ has finite order).}$$

The total nature of $\mathcal{P}_e$ allows us to reduce the guesses to

$$\Pi^0_2 \text{ (infinite order) versus } \Sigma^0_1 \text{ (finite order).}$$

- (Kalimullin, Melnikov, N) There is a computable ordered (Archimedean) abelian group without an instant copy.

When does a structure have an instant copy?

- We turn to pure relational languages. Our original conjecture was that every computable graph has an instant copy. Indeed:

Fact

Every computable locally finite graph has an instant copy.

- Converse is not true, for example the random graph and the infinite star have instant copies.
- Perhaps every computable graph has an instant copy.

Theorem (Kalimullin, Melnikov, N)

There is a computable graph with no instant copy.

When does a structure have an instant copy?

- We turn to pure relational languages. Our original conjecture was that every computable graph has an instant copy. Indeed:

Fact

Every computable locally finite graph has an instant copy.

- Converse is not true, for example the random graph and the infinite star have instant copies.
- Perhaps every computable graph has an instant copy.

Theorem (Kalimullin, Melnikov, N)

There is a computable graph with no instant copy.

Instant categoricity

- We want to look at the complexity of an instant structure by the complexity of isomorphisms between instant copies, i.e. instant categoricity.

Definition

An instant structure $\mathcal{A}$ is instantly categorical if for every instant $\mathcal{B} \cong \mathcal{A}$ there is an instant isomorphism $f : \mathcal{A} \mapsto \mathcal{B}$.

- What does an "instant isomorphism" mean?

"f and f^{-1} are both primitive recursive."

- Another candidate is to say that "$\mathrm{Graph}(f)$ is primitive recursive", but we will not adopt this.
- For computable isomorphisms, these are all equivalent.

Instant categoricity

- We want to look at the complexity of an instant structure by the complexity of isomorphisms between instant copies, i.e. instant categoricity.

Definition

An instant structure $\mathcal{A}$ is instantly categorical if for every instant $\mathcal{B} \cong \mathcal{A}$ there is an instant isomorphism $f : \mathcal{A} \mapsto \mathcal{B}$.

- What does an "instant isomorphism" mean?

  "f and f^{-1} are both primitive recursive."

- Another candidate is to say that "$\mathrm{Graph}(f)$ is primitive recursive", but we will not adopt this.
- For computable isomorphisms, these are all equivalent.

Instant categoricity: Examples

1. The additive group $\bigoplus_{i \in \omega} \mathbb{Z}_p$ is instantly categorical.
   - Given an instant copy $\mathcal{A}$, some $a \in \mathcal{A}$, and some $S \subseteq \mathcal{A}$, it is primitive recursive to check if a is linearly independent over S.
   - A back-and-forth argument works.
2. The dense linear order $(\mathbb{Q}, <)$ is surprisingly not instantly categorical.
   - However, it is categorical for primitive recursive $\mathrm{Graph}(f)$.
   - A back-and-forth argument does not work.
3. The structure (ω, Succ) is also not instantly categorical.
   - It is also not categorical for primitive recursive $\mathrm{Graph}(f)$.

Instant categoricity: Examples

1. The additive group $\bigoplus_{i \in \omega} \mathbb{Z}_p$ is instantly categorical.
   - Given an instant copy $\mathcal{A}$, some $a \in \mathcal{A}$, and some $S \subseteq \mathcal{A}$, it is primitive recursive to check if a is linearly independent over S.
   - A back-and-forth argument works.
2. The dense linear order $(\mathbb{Q}, <)$ is surprisingly not instantly categorical.
   - However, it is categorical for primitive recursive $\mathrm{Graph}(f)$.
   - A back-and-forth argument does not work.
3. The structure (ω, Succ) is also not instantly categorical.
   - It is also not categorical for primitive recursive $\mathrm{Graph}(f)$.

Instant categoricity: Examples

1. The additive group $\bigoplus_{i\in\omega} \mathbb{Z}_p$ is instantly categorical.
   - Given an instant copy $\mathcal{A}$, some $a \in \mathcal{A}$, and some $S \subseteq \mathcal{A}$, it is primitive recursive to check if a is linearly independent over S.
   - A back-and-forth argument works.
2. The dense linear order $(\mathbb{Q}, <)$ is surprisingly not instantly categorical.
   - However, it is categorical for primitive recursive $\mathrm{Graph}(f)$.
   - A back-and-forth argument does not work.
3. The structure (ω, Succ) is also not instantly categorical.
   - It is also not categorical for primitive recursive $\mathrm{Graph}(f)$.

Instant categoricity: Characterizations

Theorem (KMN)

In each of the following classes, a structure is instantly categorical if and only if it is trivial.

- *Equivalence structures: only classes of size 1, or finitely many classes at most one of which is infinite.*
- *Linear orders: finite.*
- *Boolean algebras: finite.*
- *Abelian p-groups:* $pG = 0$.
- *Torsion-free abelian groups: trivial group* $\{0\}$.

Instant categoricity and rigidity

- The examples of instantly categorical structures we've seen so far were far from rigid ($\oplus\mathbb{Z}_p$, equivalence structures).

Theorem (KMN)

- *There is a rigid functional structure which is not instantly categorical* (ω, Succ).
- *There is a rigid functional structure which is instantly categorical.*
- *However, no instantly categorical relational structure can be rigid.*

Instant categoricity vs Computable categoricity

- We saw that (ω, Succ) is an example of a computably categorical but not instantly categorical structure.
- A very natural conjecture would be that every instantly categorical structure is computably categorical.
- This is true for many natural classes (equivalence structures, linear orders, Boolean algebras, abelian *p*-groups, TFAGs).

Theorem (KMN)

There is an instantly categorical structure which is not computably categorical.

Instant categoricity vs Computable categoricity

- We saw that (ω, Succ) is an example of a computably categorical but not instantly categorical structure.
- A very natural conjecture would be that every instantly categorical structure is computably categorical.
- This is true for many natural classes (equivalence structures, linear orders, Boolean algebras, abelian *p*-groups, TFAGs).

Theorem (KMN)

There is an instantly categorical structure which is not computably categorical.

Instant categoricity and Relativization

Theorem (KMN)

Let f be total and not primitive recursive. Then there is a structure $\mathcal{A}$ which is instant relative to f but $\mathcal{A}$ has no instant copy.

Theorem (KMN)

There is a structure $\mathcal{A}$ which is instantly categorical relative to $0'$ but $\mathcal{A}$ is not instantly categorical.

Instant categoricity and Relativization

Theorem (KMN)

Let f be total and not primitive recursive. Then there is a structure $\mathcal{A}$ which is instant relative to f but $\mathcal{A}$ has no instant copy.

Theorem (KMN)

There is a structure $\mathcal{A}$ which is instantly categorical relative to $0'$ but $\mathcal{A}$ is not instantly categorical.

Questions

- Find a sufficient condition for a computable structure to have an instant copy.
- Connection with definability, Scott sentences. Note: back-and-forth works differently.
- How to define relatively instantly categorical?
- Primitive recursive analogue of 1-decidability (n-decidability).
- More work to be done on relativization, which will lead to investigations like spectra questions, degrees of categoricity, etc.
- Thank you.

Questions

- Find a sufficient condition for a computable structure to have an instant copy.
- Connection with definability, Scott sentences. Note: back-and-forth works differently.
- How to define relatively instantly categorical?
- Primitive recursive analogue of 1-decidability (n-decidability).
- More work to be done on relativization, which will lead to investigations like spectra questions, degrees of categoricity, etc.
- Thank you.