TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

Complexity of Descriptions and Relations in Computable Structure Theory

Ekaterina Fokina

Technische Universität Wien

Logique à Paris 2023

Institut de Mathématiques de Jussieu-Paris Rive Gauche
May 10, 2023

Motivating questions

Motivating Question 1

How hard is it to determine that an effectively given structure has particular properties?

Motivating Question 2

How hard is it to classify effectively given structures from a class, i.e., to determine when two structures are equivalent?

Motivating Question 3

How to classify structures on-the-fly? I.e., seeing a finite part of a structure from a fixed class, how can we determine which of the structures we are observing?

Motivating Question 4

How to go beyond computable?

Oracle Turing machines and reducibility

To formalize the idea of computability, we use

Turing machines.

Oracle Turing machines and reducibility

To formalize the idea of non-computability, we use

Turing machines with oracles.

Let X, Y be sets of natural numbers.

Definition

1. Then X **is** Y**-computable,** $X \leq_T Y$**,** iff there exists an oracle Turing machine that decides (the membership in) X using Y as an oracle.
2. The sets $X \equiv_T Y$ iff $X \leq_T Y$ and $Y \leq_T X$.

Turing degrees

Definition

A **Turing degree** is an equivalence class of $\equiv_T$.

Definition

- **0** is the degree of $\emptyset$ (all computable sets).
- $\mathbf{0}'$ is the degree of the halting set, or

$$K = \{e : \varphi_e(e) \downarrow\}$$

- if $\boldsymbol{d}$ is the degree of a set X, then $\boldsymbol{d}'$ is the degree of X', where

$$X' = \{e : \Phi_e^X(e) \downarrow\}.$$

Arithmetical hierarchy

Let Σ^0_1 be the collection of all the sets of the form

$$\{x : \exists n Q(n, x) \text{ for a computable } Q(x, n)\}$$

(i.e., all the computably enumerable, or c.e., sets).

Let Π^0_1 consist of the complements of Σ^0_1 sets, i.e. sets of the form

$$\{x : \forall n Q(n, x) \text{ for a computable } Q(x, n)\}.$$

More generally: Σ^0_n sets have the form

$$\{x : \exists n Q(n, x) \text{ for a } \Pi^0_{n-1} \text{ relation } Q(x, n)\}.$$

Π^0_n sets have the form

$$\{x : \forall n Q(n, x) \text{ for a } \Sigma^0_{n-1} \text{ relation } Q(x, n)\}.$$

Δ^0_n sets are $\Sigma^0_n \cap \Pi^0_n$.

Arithmetical hierarchy: examples

A set is Π^0_2 if it is of the form $\forall x \exists y P(n, x, y)$ or, equivalently $\exists^\infty x P(n, x)$.

A set is Σ^0_3 if it is of the form $\exists u \forall x \exists y P(n, u, x, y)$ or, equivalently $\exists u \exists^\infty x P(n, u, x)$. Sometimes such witnesses u with infinitely many x's are called *infinitary*, otherwise *finitary*.

Hyperarithmetical and analytical hierarchies

Hyperarithmetical hierarchy:
defines, for all computable ordinals α: $\Sigma^0_\alpha, \Pi^0_\alpha, \Delta^0_\alpha$ sets.

Analytical hierarchy:

Σ^1_1 sets are those presentable as $x \in X \iff \exists f \forall n R(x, f \upharpoonright n)$, where R is computable.

Π^1_1 sets are those presentable as $x \in X \iff \forall f \exists n R(x, f \upharpoonright n)$.

$\Delta^1_1 = \Sigma^1_1 \cap \Pi^1_1$ are exactly the hyperarithmetical sets.

Computable structures

For a structure in a finite vocabulary on ω, being computable simply means that the basic relations and/or functions are computable.

- A group $G = (\omega, *)$ is computable, if $*$ is a (total) computable function, i.e., we can effectively get $a * b = c$.
- A linear order $L = (\omega, <)$ is computable, if $<$ is a computable relation.

Example

Computable structures:

1. All finite structures
2. The standard model of arithmetic: $\mathcal{N} = \{\omega, +, \times, \leq, 0, 1\}$
3. The dense linear order $(\mathbb{Q}, \leq)$
4. The random graph $\mathcal{G} = (\omega, E)$
5. Field of rationals, $\mathbb{Q}(X), \mathbb{Q}(\sqrt{2})$, etc.
6. Free group on countably many generators

Computable structures, general definition

More generally,

Definition

A structure $\mathcal{A} = (A, R_0^{n_0}, R_1^{n_1}, \ldots)$ is computable if its quantifier-free diagram $D(\mathcal{A})$ is computable (thought of as a subset of ω),

where $D(\mathcal{A})$ consists of all quantifier-free sentences φ with constants from A, such that $\mathcal{A} \models \varphi$, i.e.,
$a_i * a_j = a_k, a_m * a_n \neq a_l, a_s < a_t$, etc.

Dynamic definition

Let $\mathcal{L}$ be a computable language.

TFAE:

- $\mathcal{A}$ is a computable $\mathcal{L}$-structure.
- The following holds:
  1. there exists a computable sequence $\{\mathcal{L}_n\}_{n\in\mathbb{N}}$, such that $\mathcal{L} = \bigcup_n \mathcal{L}_n$, and each $\mathcal{L}_n$ is finite;
  2. there exists a computable sequence of finite structures

$$\mathcal{A}_0 \subseteq \mathcal{A}_1 \subseteq \ldots \subseteq \mathcal{A}_n \subseteq \ldots,$$

such that for every n, $\mathcal{A}_n$ is an $\mathcal{L}_n$-structure and

$$\mathcal{A} = \bigcup_n \mathcal{A}_n.$$

Computable structures

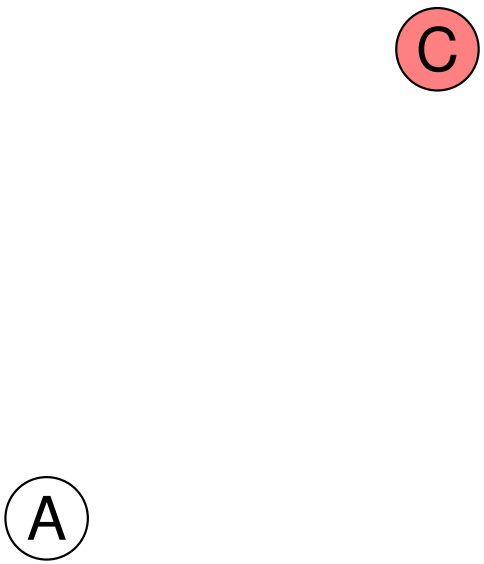
Many of the working examples in this talk will be of the form $\mathcal{A} = (A, E)$.

Computable structures

Many of the working examples in this talk will be of the form $\mathcal{A} = (A, E)$.

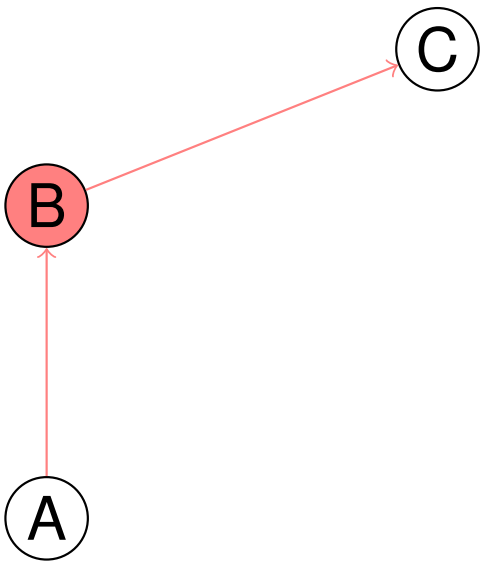
Computable structures

Many of the working examples in this talk will be of the form $\mathcal{A} = (A, E)$.



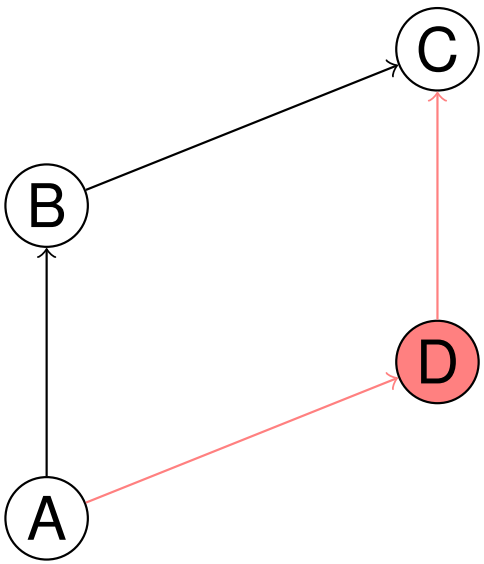
Computable structures

Many of the working examples in this talk will be of the form $\mathcal{A} = (A, E)$.



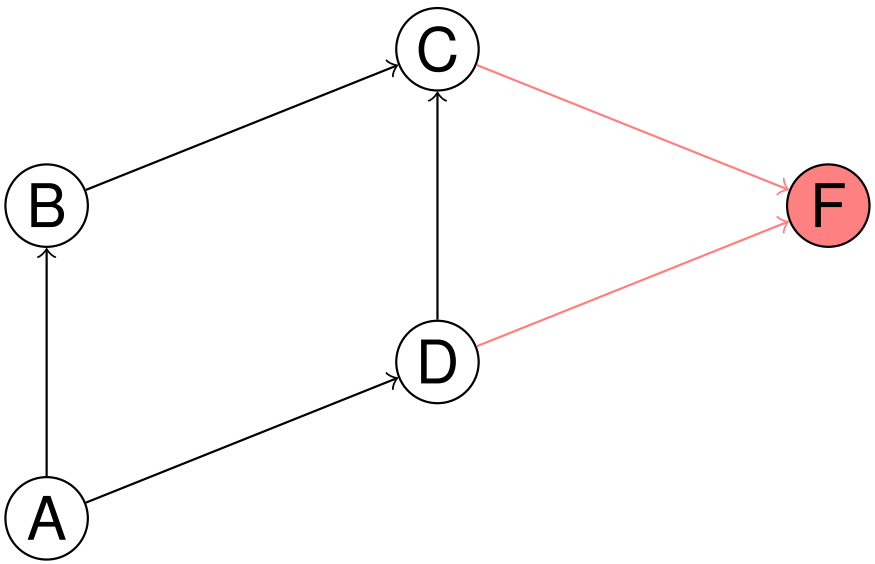
Computable structures

Many of the working examples in this talk will be of the form $\mathcal{A} = (A, E)$.



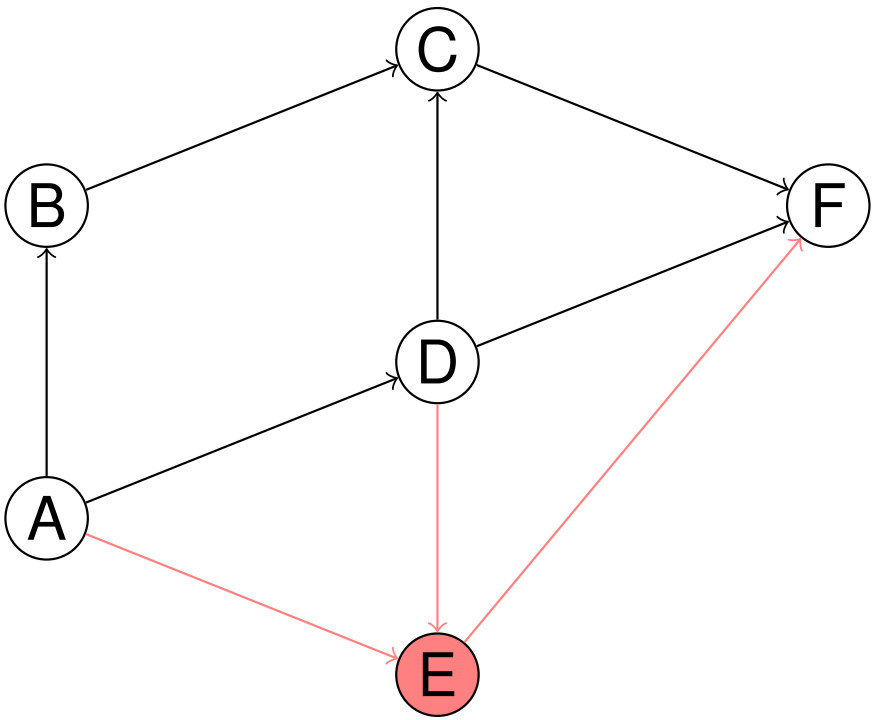
Computable structures

Many of the working examples in this talk will be of the form $\mathcal{A} = (A, E)$.



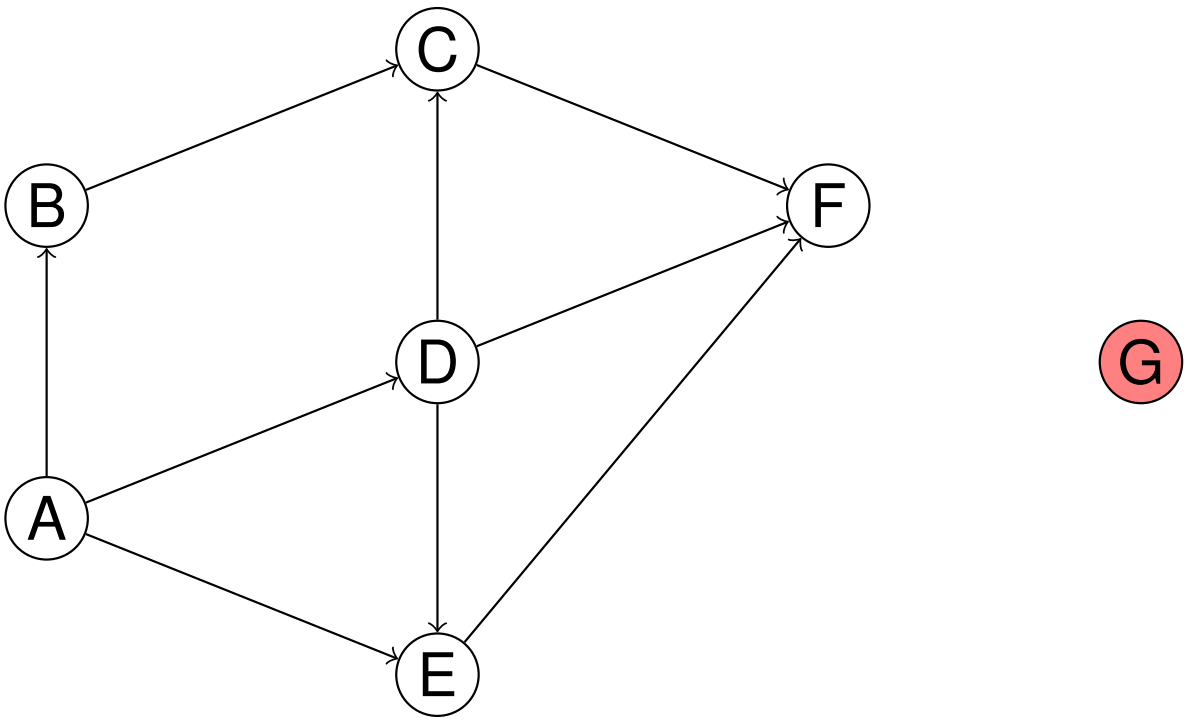
Computable structures

Many of the working examples in this talk will be of the form $\mathcal{A} = (A, E)$.



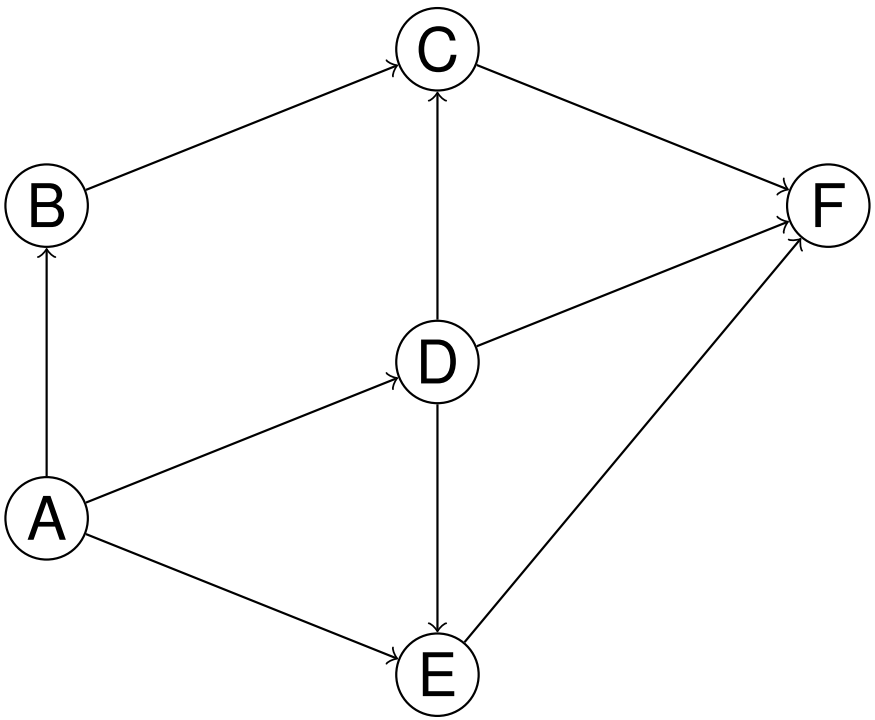
Computable structures

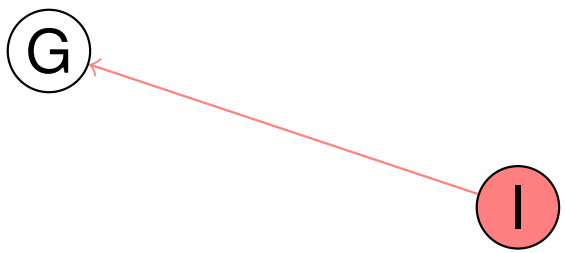
Many of the working examples in this talk will be of the form $\mathcal{A} = (A, E)$.



Computable structures

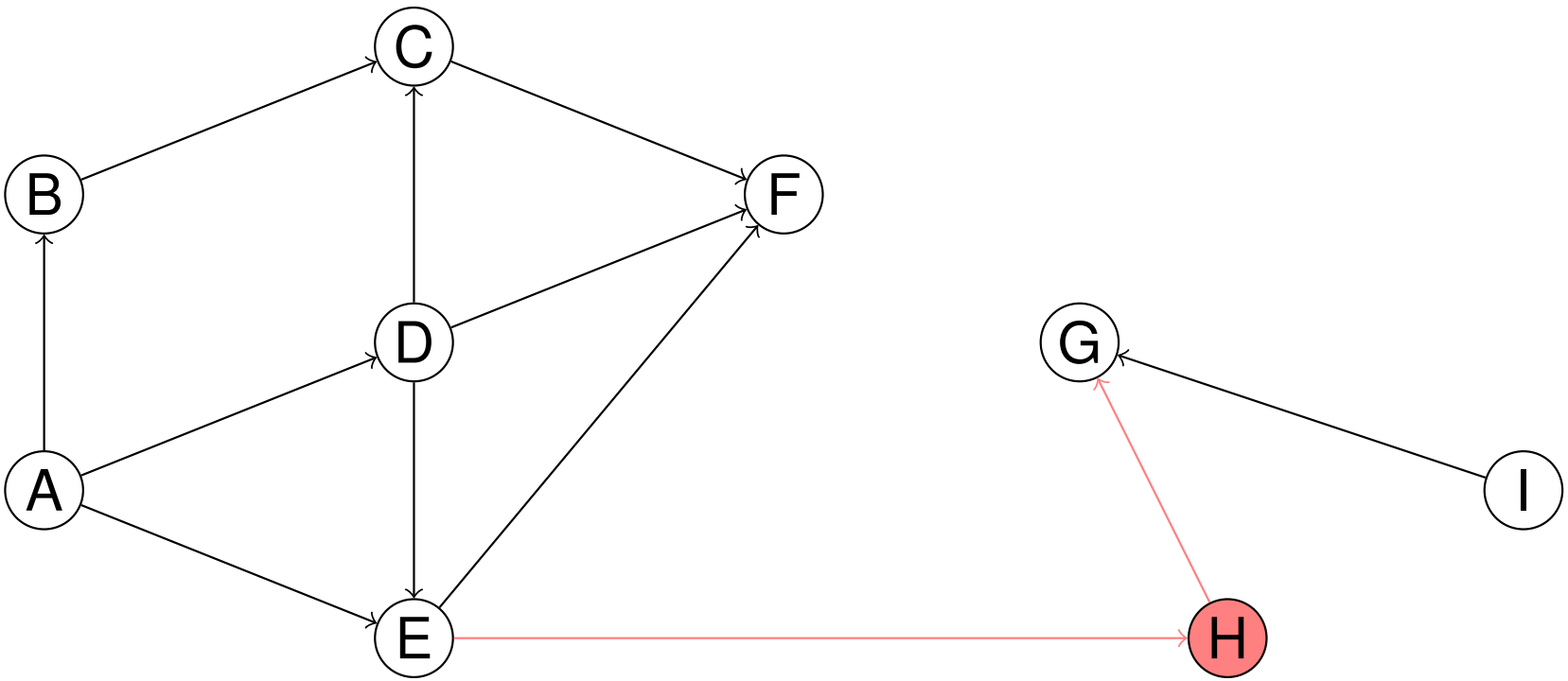
Many of the working examples in this talk will be of the form $\mathcal{A} = (A, E)$.





Computable structures

Many of the working examples in this talk will be of the form $\mathcal{A} = (A, E)$.



Existence of computable presentations

One of the main questions of computable model theory:

Question

Does a particular structure $\mathcal{A}$ have a computable presentation? More generally, what complexity can equivalent copies of the structure have?

Definition

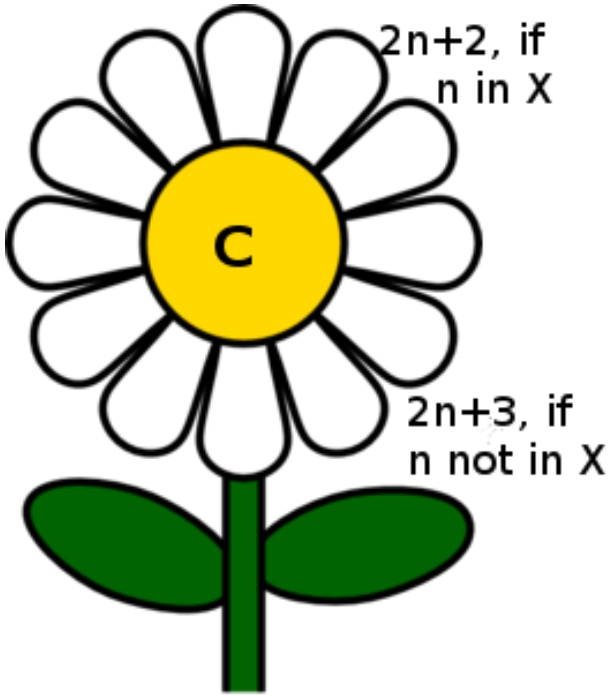
A structure $\mathcal{A}$ is **d**-computable if its atomic diagram $D(\mathcal{A})$ is a **d**-computable subset of ω.

Non-computable structures

Example 1: Daisy graphs.

For $X \subseteq \omega$, consider a graph $\mathcal{G}_X$ with:

- a special vertex c, and
- for $n \in X$, a cycle of length $2n+2$ starting at c, and
- for $n \notin X$, a cycle of length $2n+3$ starting at c.



Then if X is not computable, $\mathcal{G}_X$ is not computable.

Example 2.

Let X be a non-computable set. Consider a linear order $\mathcal{A} = (X, \leq^{\mathcal{A}})$, where $x \leq^{\mathcal{A}} y \iff x \leq y$. Then $\mathcal{A}$ is not computable by definition but it is isomorphic to the standard $(\mathbb{N}, \leq)$, which is computable.

Computable presentations

Definition

A **presentation** of a structure $\mathcal{A}$ is an isomorphic copy $\mathcal{B}$ of $\mathcal{A}$.

Example

For a set X, the daisy graph $\mathcal{G}_X$ is $deg(X)$-computable.

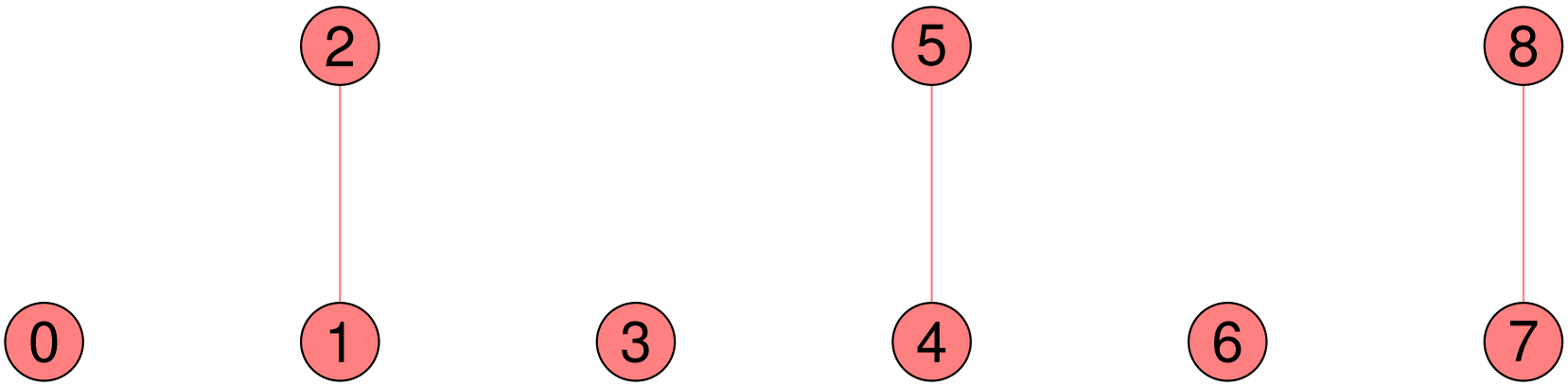
Computable equivalence structures I

Definition

An **equivalence structure** is a structure $\mathcal{A} = (A, E)$, where E is an equivalence relation on A.

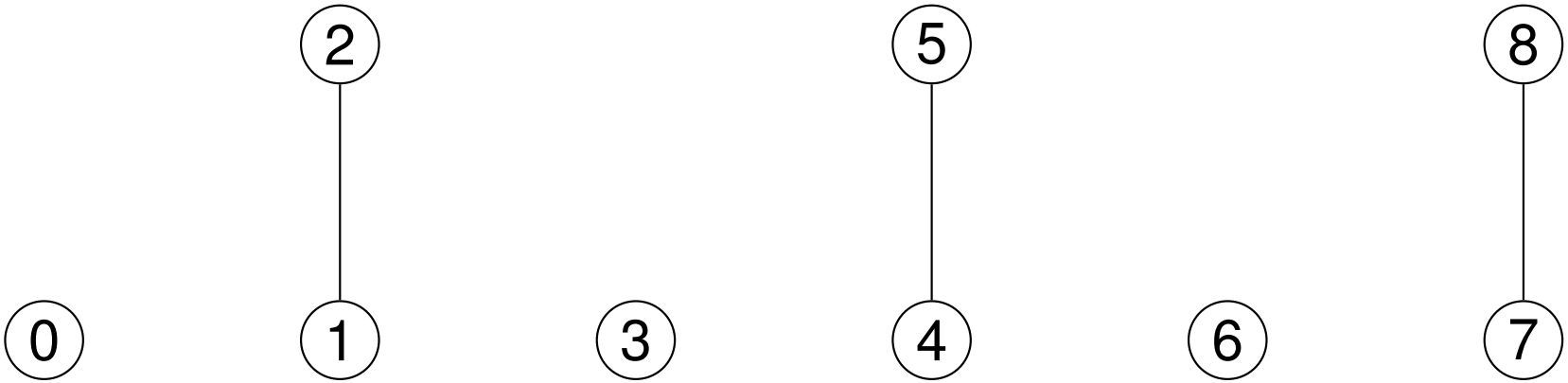
Example

An equivalence structure with infinitely many equivalence classes of size one and infinitely many classes of size two (and no other classes) has a computable presentation.

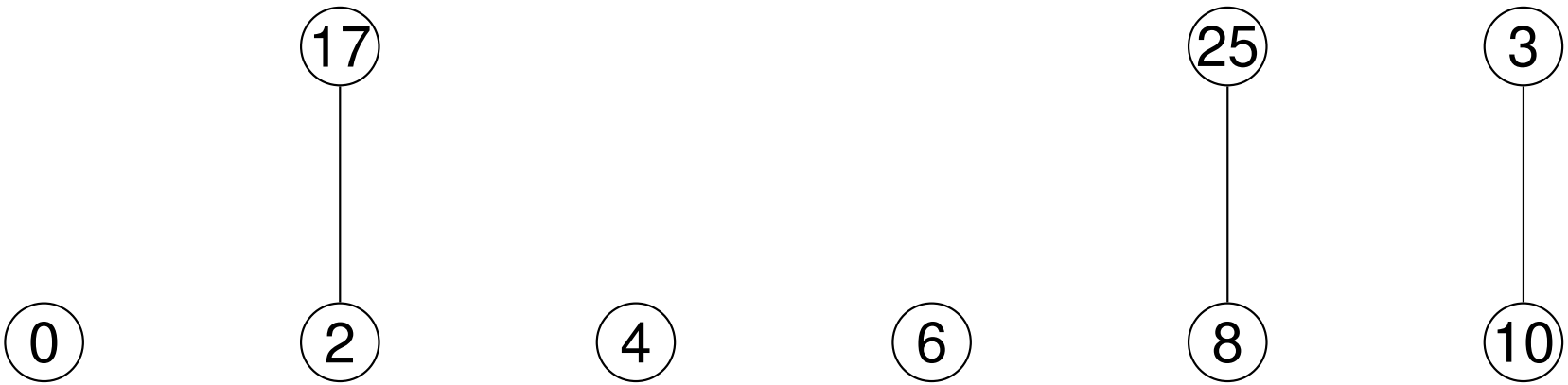


Another presentation of the same structure

The structure $\mathcal{A}$ from the previous slide:



Let X be a c.e. set. Define $\mathcal{A}_X$ as follows. If $n \in X_s$, we let $(2n, 2s+1) \in E$. Otherwise $|[2n]_E| = 1$. For example, if $5 \in A_1, 4 \in A_{12}$ and $1 \in A_8$:



Why are the presentations different?

Both presentations are computable but there is a significant difference.

- For every element of $\mathcal{A}$, the size of its equivalence class can be found effectively.
- For $\mathcal{A}_X$, the function $f(x)$ which gives back the size of $[x]_E$ in $\mathcal{A}_X$ is $\equiv_T X$. In particular, if X is not computable, $f(x)$ is not computable.
- Obviously, $\mathcal{A} \cong \mathcal{A}_X$, for every infinite and co-infinite X. However, if X is not computable (but still c.e.), then the isomorphism between $\mathcal{A}$ and $\mathcal{A}_X$ is not computable (otherwise we could pull back and compute the sizes of equivalence classes in $\mathcal{A}_X$).

This is something we will use later today.

Computable equivalence structures II

Theorem (Ash & Knight 2000, or Calvert, Cenzer, Harizanov, Morozov, 2006)

An equivalence structure has a computable presentation (i.e., isomorphic copy) if

1. $\chi(\mathcal{A}) = \{(n,k) : c_n(\mathcal{A}) \geq k\}$ *is* Σ^0_2 *and*
2. *one of the following:*
   - $\{n : c_n(\mathcal{A}) \neq 0\}$ *is finite, or*
   - $c_\infty(\mathcal{A})$ *is infinite, or*
   - *there is a computable function* f *such that for each* n, $f(n,s)$ *is non-decreasing in* s, *with limit* $f^*(n) \geq n$, *where* $c_{f^*(n)}(\mathcal{A}) \neq 0$.

More computable structures

- A hyperarithmetic ordinal always has a computable isomorphic copy (Spector, 1955).
- Every hyperarithmetic linear order has a bi-embeddable computable copy (Montalbán, 2005).
- Every hyperarithmetic Boolean algebra, a hyperarithmetic tree, a hyperarithmetic abelian p-group have a computable bi-embeddable copy (Greenberg, Montalbán, 2008).
- Every equivalence structure has a computable bi-embeddable copy (F., Rossegger, San Mauro, 2017).

Typical questions in CST

Question

- *When does a structure with specific algebraic, model-theoretic, etc. properties have a computable presentation, up to a chosen equivalence relation?*
- *What happens when it has no computable presentation?*
- *What is the complexity of these equivalence relations between structures?*

In this talk we consider the other direction:

Question

- *Given a computable structure, does it have specific algebraic, algorithmic, model-theoretic properties? How hard ist it to answer this question?*
- *Given two computable structures, how hard is it to determine if they are isomorphic, or otherwise equivalent (bi-embeddable, etc.)?*

Indexing computable structures

Recall: we can measure the complexity of subsets of the natural numbers using the arithmetical/hyperarithmetical/analytical hierarchies.

Using a Gödel numbering, we identify formulas with natural numbers.
The complexity of a structure $\mathcal{A}$ is the complexity of its atomic diagram $D(\mathcal{A})$.
One can effectively enumerate all the atomic diagrams of computable structures: $\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n, \ldots$

Index sets of classes of structures

Let K be a class of countable $\mathcal{L}$–structures closed under isomorphism.
Consider the set K^c of all computable structures from K.
Identify each computable $\mathcal{L}$–structure with its index $i \in \omega$.Then K^c can be identified with the set $I(K)$ of indices of its members:

$$I(K) = \{i \in \omega | \mathcal{A}_i \in K^c\}.$$

Goncharov and Knight, 2002: $I(K)$ is hyperarithmetical $\iff$ $K^c = Mod^c_\varphi$ for a computable infinitary sentence φ.

Complexity of algebraic properties: upper bounds

Proposition (folklore, to find in Goncharov-Knight 2002)

The index set $I(K)$ is a Π^0_2-set for the following classes K:

1. *linear orderings;*
2. *Boolean algebras;*
3. *Abelian p-groups;*
4. *equivalence structures;*
5. *vector spaces over $\mathbb{Q}$;*
6. *structures for a fixed computable language.*

Idea of the proof: examine the complexity of axioms.
Notice: the index set for a class K must be at least Π^0_2, thus, the above level of complexity is sharp.

Sharpness of the estimated complexity: lower bounds

Definition

Let Γ be a complexity class (e.g., Σ^0_3, Π^1_1, etc.). $I(K)$ is *m*-**complete** Γ if $I(K)$ is Γ and for any $S \in \Gamma$, there is a computable function f such that

$$n \in S \text{ iff } f(n) \in I(K).$$

In other words, there is a computable sequence of computable $\mathcal{L}$-structures $\{\mathcal{A}_n\}_{n\in\omega}$ for which $n \in S$ iff $\mathcal{A}_n \in K$.

Proposition (Kleene, Spector)

The index set $I(K)$ is Σ^1_1-complete for the following classes K:

1. *well orderings;*
2. *superatomic Boolean algebras;*
3. *reduced Abelian p-groups.*

Complexity of model-theoretic properties

Theorem

1. *(White; Pavlovsky) The index set of computable prime models is an m-complete* $\Pi^0_{\omega+2}$ *set.*
2. *(White) The index set of computable homogeneous models is an m-complete* $\Pi^0_{\omega+2}$ *set.*
3. *(F.) The index set of structures with decidable countably categorical theories is an m-complete* $\Sigma^0_3 - \Sigma^0_3$ *set.*
4. *(Calvert et al.) The index set of computable structures with noncomputable Scott ranks is m-complete* Σ^1_1*.*

Complexity of computability-theoretic properties

Theorem

1. *(F.) The index set of decidable structures is Σ^0_3-complete.*
2. *(White) The index set of hyperarithmetically categorical structures is Π^1_1-complete.*
3. *(Downey et al.'13) The index set of relatively computably categorical structures is Σ^0_3-complete.*
4. *(Downey et al.'15) The index set of computably categorical structures is Π^1_1-complete.*

Proof sketch: upper bound

Definition

A structure $\mathcal{A}$ is *decidable* if its complete diagram, i.e., the set of all sentences with constants from $|\mathcal{A}|$ that are true in $\mathcal{A}$, is computable.

Theorem

The index set of decidable structures is Σ^0_3-complete.

$I(K)$ is Σ^0_3.
$n \in I(K)$ iff $(\exists m)$[the function φ_m has the following properties:

1. φ_m is total, has values only in $\{0, 1\}$ and
2. $(\forall k)$(if k is the number of an open sentence, then $\varphi_m(k) = 1 \Leftrightarrow$ the sentence with the number k is true in $\mathcal{M}_n$)
3. $\&(\forall k)$(if k is the number of a conjunction of 2 sentences with numbers k_1, k_2, then $\varphi_m(k) = 1 \Leftrightarrow \varphi_m(k_1) = 1 \& \varphi_m(k_2) = 1$)
4. $\& \ldots \&(\forall k)$(if k is the number of a sentence of the form $\forall y \psi_{k'}$, then $\varphi_m(k) = 1 \Leftrightarrow (\forall l)(\varphi_m(k_l) = 1$, where k_l is the number of the sentence obtained from $\psi_{k'}$ by substitution of all free entries of variable y for l))].

Proof sketch: lower bound

Lemma (Ershov)

There exists a decidable linear ordering $L = (N, \preceq)$ of the type $\omega + \omega*$, such that the initial segment of the type ω is not c.e.

Now take an arbitrary $A \in \Sigma^0_3$. Then there exists a computable predicate $Q(n, x, y)$ such that

$$n \in A \Longleftrightarrow (\exists x)(\exists^\infty y) Q(n, x, y).$$

We construct a computable sequence $\{L_n\}$ of computable structures, such that

$n \in A \Leftrightarrow L_n$ decidable;

$n \notin A \Leftrightarrow L_n$ is not decidable.

Let L be from the lemma above. For all $x \in L$ and for all n we consider the set $L_{(n,x)} = \{\langle x', y' \rangle | x' \leq x \text{ and } Q(n, x', y')\}$ that is uniformly computable. Let $R_{(n,x)}$ be a linear ordering of $L_{(n,x)}$ such that

if $L_{(n,x)}$ is infinite then $(L_{(n,x)}, R_{(n,x)})$ has the type η;

if $L_{(n,x)}$ is finite then $(L_{(n,x)}, R_{(n,x)})$ is a linear ordering.

We define now $L_n \leftrightharpoons \sum_{x \in L} L_{(n,x)}$.

Proof sketch: lower bound II

if $L_{(n,x)}$ is infinite then $(L_{(n,x)}, R_{(n,x)})$ has the type η;

if $L_{(n,x)}$ is finite then $(L_{(n,x)}, R_{(n,x)})$ is a linear ordering.

We define now $L_n \leftrightharpoons \sum_{x \in L} L_{(n,x)}$.
If $n \in A$ then $(\exists x_0)(\exists^\infty y) Q(n, x_0, y)$. Thus, for all x such that $x_0 \leq x$ the set $L_{(n,x)}$ is infinite. By the definition of $R_{(n,x)}$

$$L_n \cong \sum_{k=0}^{r(n)} S_k + P_k,$$

where S_k is finite and $P_k \cong \eta$ for every $k \leq r(n)$, $r(n)$ is finite and depends on n. Thus, L_n is decidable. If $n \notin A$ then for all x the set $L_{(n,x)}$ is finite and $L_n \cong \omega + \omega*$. By the above lemma L_n is computable as L is computable. At the same time L_n is not decidable, as otherwise we could enumerate ω, which then would be c.e.

1 Introduction and Main Definitions

2 CST and Complexity of Descriptions

3 Equivalence Relations in CST

4 Classification Based on Learning

5 Beyond computable

Why equivalence relations

Theorems on classification/uniqueness **up to an equivalence relation** appear everywhere through out mathematics:

Theorem

Let (X, d) be a metric space. Then there exists its completion (X^*, d^*) and it is unique up to isometry.

Theorem

If $\mathcal{A}_1, \mathcal{A}_2$ are two smooth structures on $\mathbb{R}$ then there exists a diffeomorphism $F : (\mathbb{R}, \mathcal{A}_1) \to (\mathbb{R}, \mathcal{A}_2)$.

Homeomorphisms, homotheties, (quasi)-isometries, ...
Equivalence of functions, norms, ...

Measuring the complexity

Let K be a class of structures and E be an equivalence relation. Recall: $I(K)$ is the set of indices of structures from K in a fixed enumeration of all computable structures in the vocabulary of K. Goncharov & Knight:

Definition

$$I(E,K) = \{(m,n) | m,n \in I(K) \text{ and } \mathcal{A}_m E \mathcal{A}_n\}$$

Definition

Let Γ be a complexity class (e.g., Σ^0_3, Π^1_1, etc.). $I(E,K)$ is m-**complete** Γ if $I(E,K)$ is Γ and for any $S \in \Gamma$, there is a computable function f such that

$$n \in S \text{ iff } f(n) \in I(E,K).$$

In other words, there is a computable sequence of pairs of computable $\mathcal{L}$-structures $\{(\mathcal{A}_n, \mathcal{B}_n)\}_{n\in\omega}$ from K for which $n \in S$ iff $\mathcal{A}_n E \mathcal{B}_n$.

Examples

Example (Calvert)

The isomorphism problem for the class K of:

- vector spaces over Q is Π^0_3;
- algebraically closed fields of a given characteristic is Π^0_3;
- reduced Abelian p-groups of length ω is Π^0_3;
- torsion free Abelian groups of finite characteristic is Σ^0_3.

Example

The isomorphism problem is Σ^1_1-complete for:

- Abelian p-groups;
- trees;
- Boolean algebras;
- linear orderings.

Bi-embeddability

Example (Carson, F., Harizanov, Knight, Maher, Quinn, Wallbaum)

The complexity of the isomorphism problem and the bi-embeddability problem coincide:

- vector spaces over Q;
- torsion free Abelian groups of finite characteristic;
- linear orderings, Boolean algebras, Abelian p-groups;
- undirected graphs.

The isomorphism problem and the bi-embeddability problem has different complexity:

- free groups on a finite number of generators;
- reduced Abelian p-groups of length ω;
- linear orderings of some special form.

Reductions of equivalence relations

Using m-reducibility of sets glues equivalence classes together, so we loose a lot of information about the structure of the equivalence relation.

Definition

A *reduction* of an equivalence relation E on X to an equivalence relation F on Y is a function $f : X \to Y$ such that

$$xEy \iff f(x)Ff(y).$$

As a function on equivalence classes, $f : X/E \to Y/F$ is injective.

If we impose no restrictions on f, then by the Axiom of Choice, a reduction from E to F simply means that F has at least as many equivalence classes as E.

The structure becomes much more interesting and complicated if we impose definitional or algorithmic requirements on the spaces and reducing functions.

Motivation from Descriptive Set Theory

H. Friedman and Stanley (1989) and Harrington, Kechris, and Louveau (1990): definable equivalence relations under Borel reducibility.

Definition

Let E and F be equivalence relations on Polish spaces X and Y respectively. Then $E \leq_B F$ if there is a Borel $h : X \to Y$, such that

$$xEy \iff h(x)Fh(y).$$

If E and F are Borel bi-reducible, we write $E \sim_B F$.
The theory of Borel reductions has since then expanded into a broad field of research with deep connections to topology, group theory, combinatorics, model theory, ergodic theory, etc.
Calvert, Cummings, Knight, S. Miller (Quinn) (2004): *tc*–reducibility as an effective analogue of the Borel reducibility.

Computable reducibility

Definition (Ershov; Lachlan, 1970's)

Let E, F be equivalence relations on (subsets of) ω. Then $E \leq_c F$ if there exists a computable function h, such that for all x, y

$$xEy \iff h(x)Fh(y).$$

If E and F are computably bi-reducible, we write $E \sim_c F$.
The history of computable reducibility is complicated. After being introduced and studied in the 1970's, it was forgotten and rediscovered multiple times, sometimes under different names. Computable reducibility has found applications in various fields, such as the theory of numberings, proof theory, computable structure theory, combinatorial algebra, and theoretical computer science. Its systematic study began only about 15 years ago.

Isomorphism on countable structures

One of the main motivating questions to study Borel reductions was to develop a general framework to measure and compare the complexity of isomorphism relations on classes of countable structures K.
For a countable language L, let $Mod(L)$ denote the collection of all countable L-models with universe ω. Then each element of $Mod(L)$ can be viewed as an element of the product space

$$X_L = \prod_{i \in I} 2^{\omega^{n_i}}$$

which is homeomorphic to the Cantor space.

Borel on top

Definition

A class K of countable structures is *on top* for $\leq_B$ if, for every countable language L, $\cong_L$ Borel reduces to $\cong_K$.

Many familiar classes are on top:

- H. Friedman, Stanley (1989): Undirected graphs, trees, linear orders;
- Camerlo, Gao (2001): Boolean algebras;
- Paolini, Shelah (preprint): Torsion-free abelian groups.

Some are not on top, e.g., torsion abelian groups.

Computable reducibility on classes of structures

1. Consider a (nice) class of structures K.
2. Identify computable structures K^c from K with the set $I(K) \subseteq \omega$ of indices in a fixed effective enumeration of computable structures.
3. Identify a relation E on K^c with the binary relation $\{(i,j)|i,j \in I(K) and \mathcal{A}_i E \mathcal{A}_j\} \subseteq \omega^2$.

Then, to compare isomorphism relations on computable structures, we considers partial computable reductions with domain containing the relevant set $I(K)$. More generally:

Definition

Let E, F be equivalence relations on (hyperarithmetical) subsets X, Y of ω respectively. Then E is computably reducible to F, $E \leq_c F$, if there exists a partial computable function h, such that $X \subseteq dom(h), h(X) \subseteq Y$ and for all $i, j \in X$,

$$iEj \iff h(i)Fh(j).$$

Bi–embeddability is Σ^1_1 complete

Theorem (F. and S. Friedman)

The equivalence relation of bi-embeddability on computable graphs is Σ^1_1 complete among equivalence relations.

Theorem (F., S. Friedman, Harizanov, Knight, McCoy, Montalbán)

The equivalence relation of isomorphism on computable structures from the following classes is complete for all Σ^1_1 equivalence relations on ω:

1. *graphs and trees,*
2. *torsion–free abelian groups,*
3. *abelian p–groups,*
4. *fields, and others.*

Notice differences from the Borel theory.

Effective isomorphism

Theorem (F., S. Friedman, Nies)

The computable isomorphism on computable structures is a Σ^0_3 complete equivalence relation for the following classes:

- *trees,*
- *equivalence structures,*
- *Boolean algebras, and others.*

The result relativizes for any computable successor ordinal.

Theorem (Greenberg, Turetsky)

The relation of hyperarithmetical isomorphism is complete for Π^1_1 equivalence relations.

Proof sketch

Recall that sets $A, B \subseteq \omega$ are *1-equivalent*, $A \equiv_1 B$, if there is a computable permutation h of ω such that $h(A) = B$.

Theorem

For each Σ^0_3 equivalence relation S, there is a computable function g such that

$$\begin{aligned} ySz &\Rightarrow W_{g(y)} \equiv_1 W_{g(z)}, \text{ and} \\ \neg ySz &\Rightarrow W_{g(y)}, W_{g(z)} \text{ are Turing incomparable.} \end{aligned}$$

As an immediate consequence, we have:

Corollary

Many-one equivalence and 1-equivalence on indices of c.e. sets are Σ^0_3 complete for equivalence relations under computable reducibility.

Proof sketch

We code the above result in equivalence structures:

Proposition

Computable isomorphism of computable equivalence relations where every class has at most 2 members is a complete Σ^0_3 equivalence relation.

Proof.

Recall the equivalence structures $\mathcal{A}_X$ built from c.e. sets X. It is easy to see that $W_y \equiv_1 W_z$ iff $\mathcal{A}_{W_y}$ is computably isomorphic to $\mathcal{A}_{W_z}$. Now we apply Corollary.

□

Proof sketch

We code the above result in trees:

Proposition

Computable isomorphism of computable trees of height 2 *where every node at level* 1 *has out-degree at most* 1 *is a complete* Σ^0_3 *equivalence relation.*

Proof.

Let h be a computable function such for each e, $T_{h(e)}$ is the tree

$$\{\emptyset\} \cup \{\langle x \rangle : x \in \omega\} \cup \{\langle x, 0 \rangle : x \in W_e\}.$$

Clearly, $W_y \equiv_1 W_z$ iff $T_{h(y)}$ is computably isomorphic to $T_{h(z)}$.
Now we apply Corollary. □

On-the-fly classification of structures

- Suppose we have a class of (countable) structures.
- Suppose we are stage by stage seeing one of the structures from the class: at each step a larger and larger finite piece of the structure.

Question

Can we, after finitely many steps, identify the structure (up to a suitable equivalence relations)?

Classifiable = after finitely many steps we correctly identify the seen structure

Examples

Structures $(\omega, \leq)$ and $(\omega^*, \leq)$

Hypothesis: ω^*

Examples

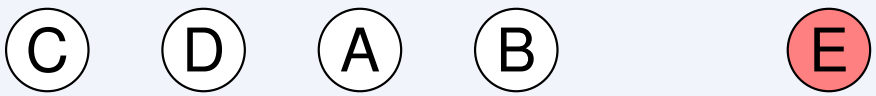
Structures $(\omega, \leq)$ and $(\omega^*, \leq)$

C D A B

Hypothesis: ω^*

Examples

Structures $(\omega, \leq)$ and $(\omega^*, \leq)$



Hypothesis: ω

Examples

Structures $(\omega, \leq)$ and $(\omega^*, \leq)$

F C D A B E

Hypothesis: ω^*

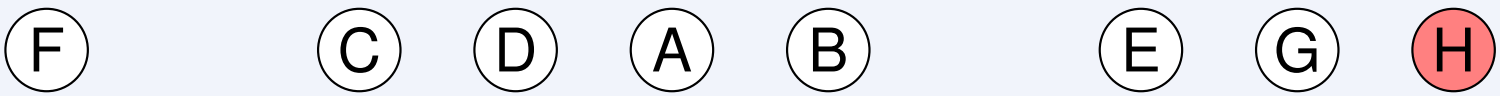
Examples

Structures $(\omega, \leq)$ and $(\omega^*, \leq)$

Hypothesis: ω

Examples

Structures $(\omega, \leq)$ and $(\omega^*, \leq)$



Hypothesis: ω

Examples

Structures $(\omega, \leq)$ and $(\omega^*, \leq)$

Ⓕ Ⓘ Ⓒ Ⓓ Ⓐ Ⓑ Ⓙ Ⓔ Ⓖ Ⓗ

Hypothesis: ω

Examples

Structures $(\omega, \leq)$ and $(\omega^*, \leq)$

Hypothesis: ω

Examples

Structures $(\omega, \leq)$ and $(\omega^*, \leq)$

Hypothesis: ω

Structures $(\omega, \leq)$ and $(\zeta, \leq)$

Hypothesis: alternating between ω and ζ.

Conclusion and overview

- How is the structure “revealed”?
- Main approach: both the positive and the negative information about the structure (motivated by computable structures).
- What does it mean “to classify”, or “to identify” the structure?
- It turns out that it is convenient to use the terminology and ideas from computational learning theory.
- The results still belong to computable structure theory, revealing interesting links to descriptive set theory and topology.

Computational Learning Theory

Computational Learning Theory (CLT): deals with the question of how a learner, provided with more and more data about some environment, is eventually able to achieve systematic knowledge about it.

- (Gold, 1967): language identification.

Most work in CLT concerns

- either learning of total functions (where the order in which the data is received matters)
- or learning of formal languages (where the order does not matter)

These paradigms model the data to be learned as an unstructured flow — but what if one deals with data having some structural content?

CLT and Structures

More recently researchers applied the machinery of CLT to algebraic structures:

- Glymour, 1985
- Martin, Osherson, 1998
- Stephan, Ventsov, 2001: learning ring ideals of commutative rings.
- Merkle, Stephan, 2004 learning isolated branches on uniformly computable sequences of trees.
- Harizanov, Stephan, 2007: learning subspaces of V_∞.
- Gao, Stephan, Wu, Yamamoto, 2012: learning closed sets in matroids.
- F., Kötzing, San Mauro, 2018: learning equivalence structures.

Our goal is: to combine the technology of CLT with notions coming from computable structure theory to develop a general framework for learning the isomorphism type of algebraic structures.

Computable Structures

To learn the isomorphism type of a given structure, one should be able to name such an isomorphism type. This is why we focus on the learning of (copies of) computable or c.e. structures.

Learning should be independent from the way in which data is presented. So, a successful learning procedure should work for all isomorphic copies of a given structure.

Learning structures

Combining computable structures and algorithmic learning:

- Let K be a class of structures. Fix some uniform effective enumeration $\{\mathcal{C}_i\}_{i\in\omega}$ of the computable structures from (possibly) a superclass K_0, up to isomorphism.
- A *learner* **M** is a total function which takes for its inputs finite substructures of a given structure $\mathcal{S}$ from K.
- For a fixed equivalence relation $\sim$, **M** *learns* $\mathcal{S}$ up to $\sim$ if, for all $\mathcal{T} \cong \mathcal{S}$, there exists $n \in \omega$ such that $\mathcal{T} \sim \mathcal{C}_n$ and $\mathbf{M}(\mathcal{T}^i)\downarrow = n$, for all but finitely many i.
- A family of structures $\mathfrak{A}$ is *learnable* up to $\sim$ if there is **M** that learns all $\mathcal{A} \in \mathfrak{A}$.

Infinitary formulas

We are able to fully characterize which families of structures are learnable. To do so, we use the logic $\mathcal{L}_{\omega_1\omega}$, which allows to take conjunctions or disjunctions of infinite sets of formulas. Suppose that K_0 is a class of L-structures, and ν is an effective enumeration of the class K_0.

Theorem (Bazhenov, F., San Mauro)

Let $K = \{\mathcal{B}_i : i \in \omega\}$ be a family of structures such that $K \subseteq K_0$, and the structures $\mathcal{B}_i$ are infinite and pairwise non-isomorphic. Then the following conditions are equivalent:

1. *The class K is learnable;*
2. *There is a sequence of Σ_2^{inf} sentences $\{\psi_i : i \in \omega\}$ such that for all i and j, we have $\mathcal{B}_j \models \psi_i$ if and only if $i = j$.*

Under some reasonable effectiveness restrictions (e.g., the sequence of sentences above should be uniformly X-computable) the class K above is learnable via an X-computable learner.

Applications

Corollary (BFS)

- *There exist learnable classes of:*
  - *lattices,*
  - *abelian groups,*
  - *linear orders (only finite classes)*
- *There are no learnable classes for*
  - *Boolean algebras,*
  - *Infinite classes of linear orders.*

Other learnability classes

We obtain different learnability classes by replacing the main ingredients of the definition with natural alternatives:

1. Learning from *text*: in **Txt**-learning the learner receives only positive information of the structure to be learnt.
2. $\cong \mapsto E$, where E is some nice equivalence relations relation between elements of $\mathbb{K}$, such as bi-embeddability ($\approx$), computable isomorphism ($\cong^{\mathbf{0}}$), computable bi-embeddability ($\approx^{\mathbf{0}}$) – and so forth.
3. modifications in convergence behaviour: e.g., in **BC**-learning (short for *behaviourally correct*) the learner is allowed to change its mind infinitely many times as far as almost all its conjectures lie in the same E-class (with E defined as in 2.).
4. Yet another dimension to consider is the complexity of the learner.

Syntactic characterization for TxtEx-learning

Theorem (Bazhenov, F., Rosseger, A. Soskova, Vatev)

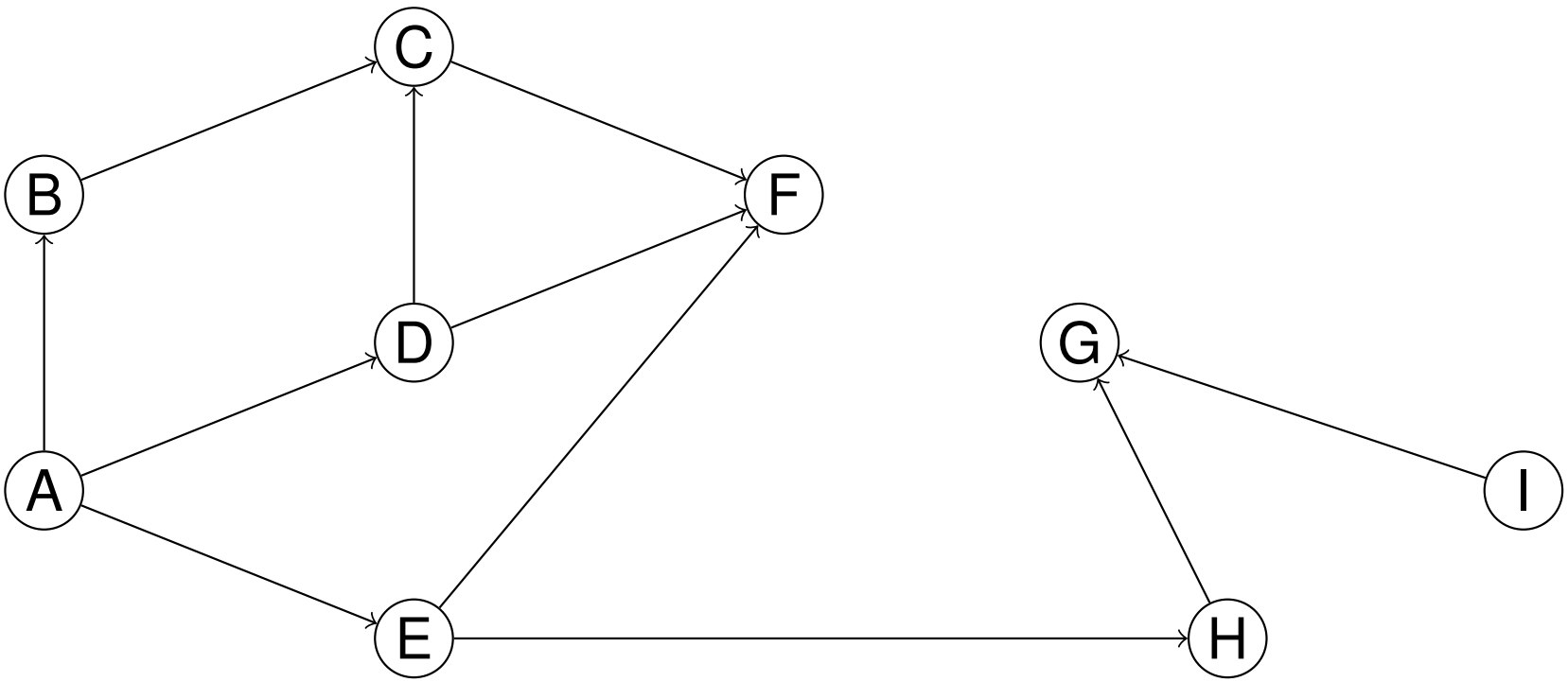
Let $K = \{\mathcal{A}_i : i \in \omega\}$ be a class of computable infinite L-structures (here we assume that $\mathcal{A}_i \not\cong \mathcal{A}_j$ for $i \neq j$), such that. The following are equivalent.

- *For every $\mathcal{A}_i \in K$ there is a Σ^p_2 sentence φ_i such that for all $\mathcal{A}_k \in K$, $\mathcal{A}_k \models \varphi_i$ if and only if $k = i$.*
- *the class K is* **Txt***-learnable.*

Again, with extra effectiveness, the class K is learnable by an X-computable learner.

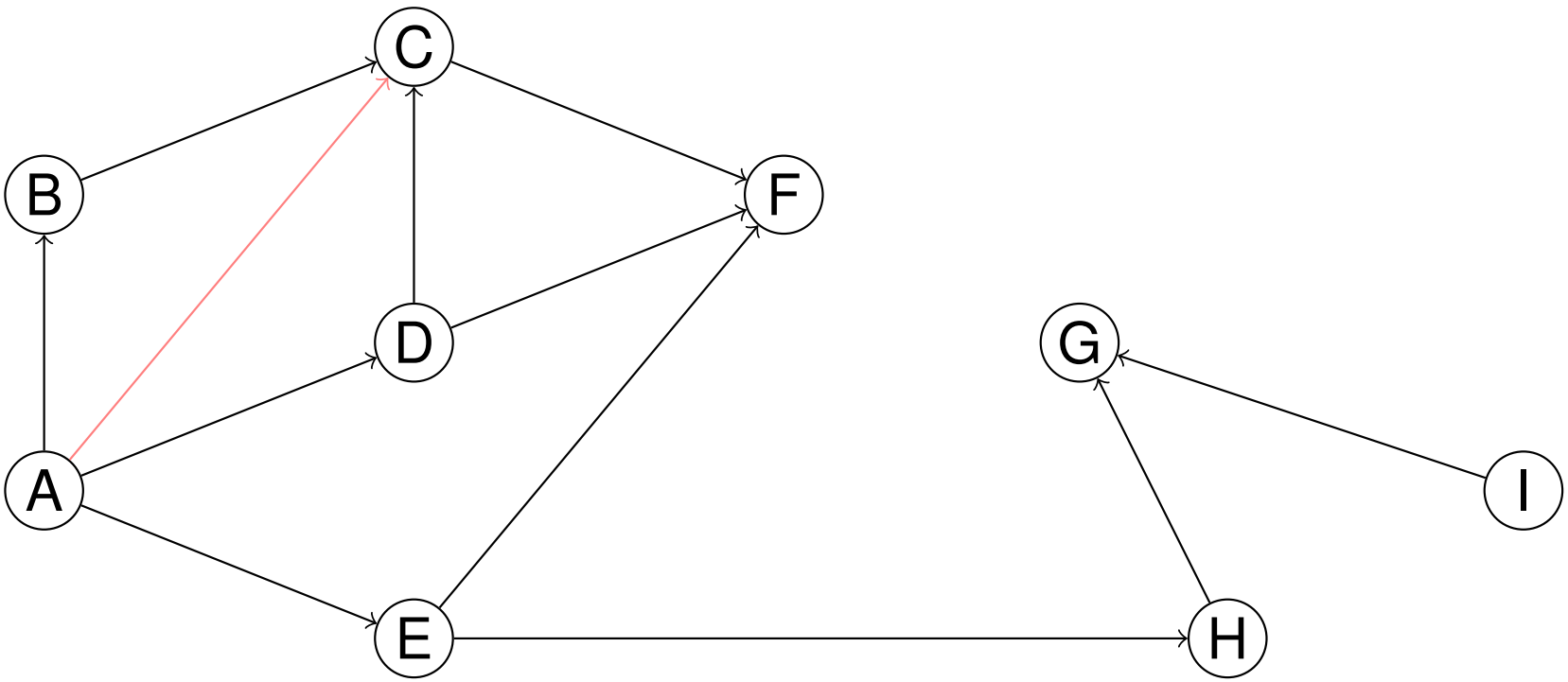
Computable and c.e. structures

We consider finite (relational) signatures L.
$\mathcal{A}$ is **c.e., or Σ^0_1, or positive** if $A = \omega$ and the interpretations of L are c.e.



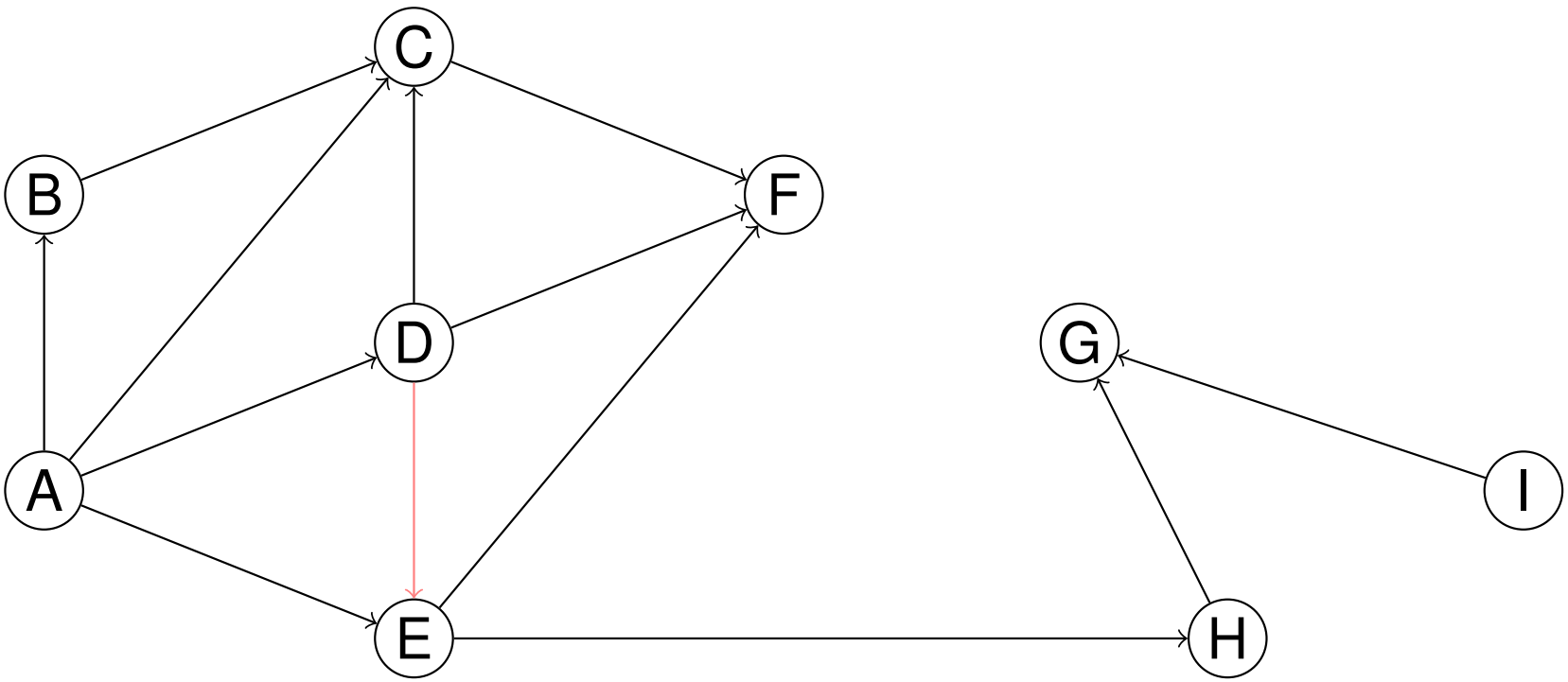
Computable and c.e. structures

We consider finite (relational) signatures L.
$\mathcal{A}$ is **c.e., or Σ^0_1, or positive** if $A = \omega$ and the interpretations of L are c.e.



Computable and c.e. structures

We consider finite (relational) signatures L.
$\mathcal{A}$ is **c.e., or Σ^0_1, or positive** if $A = \omega$ and the interpretations of L are c.e.



Σ^0_1 structures

There are several papers on Σ^0_1 structures, though the topic was considered to be quite exotic, as it somewhat failed on motivation:

- Cenzer, Harizanov, Remmel: Σ^0_1 and Π^0_1 equivalence structures
- CHR: Σ^0_1 and Π^0_1 injection structures

There is a natural case when Σ^0_1 structures become particularly useful: when we consider *partial* structures, i.e., structures with partial basic functions.

Definition

A *partial applicative structure* (pas) is a set $\mathcal{A}$ together with a partial map $\cdot$ from $\mathcal{A} \times \mathcal{A}$ to $\mathcal{A}$. We usually write ab instead of $a \cdot b$, and think of this as 'a applied to b'. If this is defined we denote this by $ab\downarrow$. By convention, application associates to the left, so we write abc instead of $(ab)c$.

Partial combinatory algebras, or PCAs

Definition

A pas $\mathcal{A}$ is called *combinatory complete* if for any term $t(x_1, \ldots, x_n, x)$, $n \geq 0$, with free variables among $x_1, \ldots, x_n, x$, there exists $b \in \mathcal{A}$ such that for all $a_1, \ldots, a_n, a \in \mathcal{A}$,

(i) $ba_1 \cdots a_n \downarrow$,

(ii) $ba_1 \cdots a_n a \simeq t(a_1, \ldots, a_n, a)$.

A pas $\mathcal{A}$ is a *partial combinatory algebra* (pca) if it is combinatory complete.

The standard example of a pca is Kleene's first model K_1, with application on the natural numbers defined by

$$n \cdot m = \varphi_n(m).$$

This is the setting of classical computability theory.
Other PCAs are closely related to the enumeration degrees in computability theory, lambda calculus, constructive mathematics, etc.

Complexity of descriptions for PCAs

Given a pca $\mathcal{A}$ on ω in which the relation $a \cdot b \downarrow= c$ is c.e., we can represent $\mathcal{A}$ by the c.e. set

$$W = \{\langle a, b, c\rangle \mid a \cdot b \downarrow= c\}.$$

The question then arises what the complexity is of the set of indices of c.e. sets that encode pcas in this way.

Definition

The index set of c.e. pcas is defined as

$$I(\text{pca}) = \{e \mid W_e \text{ is a pca}\}.$$

Using this definition of pcas, we obtain Π^0_4 as an upper bound for the complexity of the index set $I(\text{pca})$. However, we can do better.

Complexity of the index set for PCAs

Theorem (Feferman)

A pas $\mathcal{A}$ is a pca if and only if it has elements k and s with the following properties for all $a, b, c \in \mathcal{A}$:

- *k is total and $kab = a$,*
- *$sab\downarrow$ and $sabc \simeq ac(bc)$.*

Theorem (F., Terwijn)

$I(pca)$ is Σ^0_3-complete.

Upper bound: follows by counting quantifiers in the Feferman's characterization above.
Lower bound: for a Σ^0_3 set B, we construct a sequence of structures $(\mathcal{A}_x)_{x\in\omega}$, such that

- every $\mathcal{A}_x$ is a pas,
- if $x \in B$, then $\mathcal{A}_x$ is some alternative coding of K_1, and
- if $x \notin B$ then $\mathcal{A}$ is not a pca.

Thank you!