Computable Structure Theory

Barbara F. Csima

February 8, 2008

EMAIL: csima@math.uwaterloo.ca
WEB: www.math.uwaterloo.ca/∼csima
Department of Pure Mathematics
University of Waterloo

Intuitive Computability

Intuitively, we would say a set is computable if we could program a computer (with a finite program), such that running the program on any reasonable input would halt after finitely many steps and say "yes" if the input was in the set, and "no" otherwise.

Intuitive Computability

Intuitively, we would say a set is computable if we could program a computer (with a finite program), such that running the program on any reasonable input would halt after finitely many steps and say "yes" if the input was in the set, and "no" otherwise.

Example

The set of names of people in this room is computable.

Intuitive Computability

Intuitively, we would say a set is computable if we could program a computer (with a finite program), such that running the program on any reasonable input would halt after finitely many steps and say "yes" if the input was in the set, and "no" otherwise.

Example

The set of names of people in this room is computable.

Example

The set of names of descendants of people in this room may not be computable.

Computability

A Turing machine is a computer which uses finite programs, and takes as input natural numbers. A program on given input may either output a natural number, or fail to halt.

Computability

A Turing machine is a computer which uses finite programs, and takes as input natural numbers. A program on given input may either output a natural number, or fail to halt.

Definition

To each program, we can then associate a partial function from $\mathbb{N}$ to $\mathbb{N}$. Such a function is called partial computable.

Computability

A Turing machine is a computer which uses finite programs, and takes as input natural numbers. A program on given input may either output a natural number, or fail to halt.

Definition

To each program, we can then associate a partial function from $\mathbb{N}$ to $\mathbb{N}$. Such a function is called partial computable.

Definition

A function $f : \mathbb{N} \to \mathbb{N}$ is computable if it is partial computable and has a value on every $n \in \mathbb{N}$.

Computability

A Turing machine is a computer which uses finite programs, and takes as input natural numbers. A program on given input may either output a natural number, or fail to halt.

Definition

To each program, we can then associate a partial function from $\mathbb{N}$ to $\mathbb{N}$. Such a function is called partial computable.

Definition

A function $f : \mathbb{N} \to \mathbb{N}$ is computable if it is partial computable and has a value on every $n \in \mathbb{N}$.

Definition

A set $A \subseteq \mathbb{N}$ is computable if its characteristic function is computable.

Why use partial computable functions?

Note that there is no effective list of all the computable functions.

Why use partial computable functions?

Note that there is no effective list of all the computable functions.

Indeed, suppose $f_0, f_1, f_2, \ldots$ were an effective listing of all computable functions. Then the function $g = f_e(e) + 1$ would also be computable, but not appear on the list, a contradiction.

Why use partial computable functions?

Note that there is no effective list of all the computable functions.

Indeed, suppose $f_0, f_1, f_2, \ldots$ were an effective listing of all computable functions. Then the function $g = f_e(e) + 1$ would also be computable, but not appear on the list, a contradiction.

Since Turing machines use finite programs, all possible Turing programs can be listed. Hence, we have an effective listing of all partial computable functions. Let $\Phi_0, \Phi_1, \Phi_2, \ldots$ be such a listing.

The Halting Problem

The halting problem is to decide whether a given program halts on given input.

The Halting Problem

The halting problem is to decide whether a given program halts on given input.

Definition

Let $\emptyset' = \{e \in \mathbb{N} \mid \Phi_e(e) \downarrow\}$.

The Halting Problem

The halting problem is to decide whether a given program halts on given input.

Definition

Let $\emptyset' = \{e \in \mathbb{N} \mid \Phi_e(e) \downarrow\}$.

The set $\emptyset'$ is not computable.

The Halting Problem

The halting problem is to decide whether a given program halts on given input.

Definition

Let $\emptyset' = \{e \in \mathbb{N} \mid \Phi_e(e) \downarrow\}$.

The set $\emptyset'$ is not computable. Indeed, consider the function

$$g(x) = \begin{cases} \Phi_x(x) + 1 & \text{if } x \in \emptyset', \\ 0 & \text{if } x \notin \emptyset'. \end{cases} \tag{1}$$

The Halting Problem

The halting problem is to decide whether a given program halts on given input.

Definition

Let $\emptyset' = \{e \in \mathbb{N} \mid \Phi_e(e) \downarrow\}$.

The set $\emptyset'$ is not computable. Indeed, consider the function

$$g(x) = \begin{cases} \Phi_x(x) + 1 & \text{if } x \in \emptyset', \\ 0 & \text{if } x \notin \emptyset'. \end{cases} \tag{1}$$

If $\emptyset'$ were computable, then g would be computable. But then there would be some e such that $g = \Phi_e$. Since g is computable, we must have $e \in \emptyset'$. But then $g(e) = \Phi_e(e) + 1 = g(e) + 1$, a contradiction.

The Halting Problem

The halting problem is to decide whether a given program halts on given input.

Definition

Let $\emptyset' = \{e \in \mathbb{N} \mid \Phi_e(e) \downarrow\}$.

The set $\emptyset'$ is not computable. Indeed, consider the function

$$g(x) = \begin{cases} \Phi_x(x) + 1 & \text{if } x \in \emptyset', \\ 0 & \text{if } x \notin \emptyset'. \end{cases} \tag{1}$$

If $\emptyset'$ were computable, then g would be computable. But then there would be some e such that $g = \Phi_e$. Since g is computable, we must have $e \in \emptyset'$. But then $g(e) = \Phi_e(e) + 1 = g(e) + 1$, a contradiction.

Hence there is no decision procedure to decide the halting problem.

Relative Computability

Definition

For sets A and B, we say A is Turing reducible to B, $A \leq_T B$, if there is a program which computes A, using B as an oracle.

Relative Computability

Definition

For sets A and B, we say A is Turing reducible to B, $A \leq_T B$, if there is a program which computes A, using B as an oracle.

We say $A \equiv_T B$ if $A \leq_T B$ and $B \leq_T A$. A Turing degree is an equivalence class under $\equiv_T$.

Relative Computability

Definition

For sets A and B, we say A is Turing reducible to B, $A \leq_T B$, if there is a program which computes A, using B as an oracle.

We say $A \equiv_T B$ if $A \leq_T B$ and $B \leq_T A$. A Turing degree is an equivalence class under $\equiv_T$.

Definition

We let $\mathbf{0}$ denote the collection of computable sets. This is the least Turing degree.

Jump Operator

Definition

For $A \subset \mathbb{N}$, let $A' = \{e \in \mathbb{N} \mid \Phi_e^A(e) \downarrow\}$. We call this the jump of A.

Note that if $A \leq_T \emptyset'$, then $\emptyset' \leq_T A' \leq_T \emptyset''$.

Low and High

Definition

A set A is *low* if $A' \equiv_T \emptyset'$.

Low and High

Definition

A set A is *low* if $A' \equiv_T \emptyset'$.

Definition

A set $A \leq_T \emptyset'$ is high if $A' \equiv_T \emptyset''$.

Low and High

Definition

A set A is *low* if $A' \equiv_T \emptyset'$.

Definition

A set $A \leq_T \emptyset'$ is high if $A' \equiv_T \emptyset''$.

We think of low sets as close to computable, and high sets as close to the halting problem.

Computably Enumerable sets

Definition

A set A is computably enumerable (c.e.) if it is the domain of a partial computable function.

Computably Enumerable sets

Definition

A set A is computably enumerable (c.e.) if it is the domain of a partial computable function.

Equivalently, a set A is computably enumerable if there is some computer that enumerates the members of A.

Computably Enumerable sets

Definition

A set A is computably enumerable (c.e.) if it is the domain of a partial computable function.

Equivalently, a set A is computably enumerable if there is some computer that enumerates the members of A.

Example

The halting set, $\emptyset'$, is computably enumerable.

Computably Enumerable sets

Definition

A set A is computably enumerable (c.e.) if it is the domain of a partial computable function.

Equivalently, a set A is computably enumerable if there is some computer that enumerates the members of A.

Example

The halting set, $\emptyset'$, is computably enumerable.

In fact, $\emptyset'$ is the most complicated c.e. set, in the sense that any other c.e. set is Turing reducible to it.

Classical versus effective

We often ask whether effective versions of classical theorems still hold.

Classical versus effective

We often ask whether effective versions of classical theorems still hold.

Theorem

Every infinite binary branching tree has an infinite path.

Classical versus effective

We often ask whether effective versions of classical theorems still hold.

Theorem

Every infinite binary branching tree has an infinite path.

Theorem

There exists a computable binary branching tree with no computable path (paths are infinite).

Classical versus effective

We often ask whether effective versions of classical theorems still hold.

Theorem

Every infinite binary branching tree has an infinite path.

Theorem

There exists a computable binary branching tree with no computable path (paths are infinite).

Theorem (Jockusch, Soare)

Every infinite computable binary branching tree has a low path.

Computable Structure theory

We study the Turing degrees of various (usually countable) mathematical structures.

Computable Structure theory

We study the Turing degrees of various (usually countable) mathematical structures.

Definition

We say a structure $\mathcal{A}$ is computable if it has domain $\mathbb{N}$ and all functions and relations on $\mathcal{A}$ are computable.

Computable Structure theory

We study the Turing degrees of various (usually countable) mathematical structures.

Definition

We say a structure $\mathcal{A}$ is computable if it has domain $\mathbb{N}$ and all functions and relations on $\mathcal{A}$ are computable.

Example

A linear order is computable if there is a program that for each pair (a, b) computes whether $a < b$ or $b < a$.

Computable Structure theory

We study the Turing degrees of various (usually countable) mathematical structures.

Definition

We say a structure $\mathcal{A}$ is computable if it has domain $\mathbb{N}$ and all functions and relations on $\mathcal{A}$ are computable.

Example

A linear order is computable if there is a program that for each pair (a, b) computes whether $a < b$ or $b < a$.

Example

A graph is computable if the edge relation is computable

Definition

For any structure $\mathcal{A}$, Spec$(\mathcal{A})$= $\{deg(\mathcal{B}) \mid \mathcal{B} \cong \mathcal{A}\}$.

Definition

For any structure $\mathcal{A}$, Spec$(\mathcal{A})$= $\{deg(\mathcal{B}) \mid \mathcal{B} \cong \mathcal{A}\}$.

We are interested in the kinds of degree spectra that natural structures can have.

Definition

For any structure $\mathcal{A}$, Spec$(\mathcal{A})$= $\{deg(\mathcal{B}) \mid \mathcal{B} \cong \mathcal{A}\}$.

We are interested in the kinds of degree spectra that natural structures can have.

Theorem (Julia Knight)

The degree spectra of a non-trivial structure is upward closed in the Turing degrees.

Computable Categoricity

Definition

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

Computable Categoricity

Definition

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

Example

Any two computable dense linear orders without endpoints are computably isomorphic. Thus, any computable dense linear order without endpoints is computably categorical.

Computable Categoricity

Definition

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

Example

Any two computable dense linear orders without endpoints are computably isomorphic. Thus, any computable dense linear order without endpoints is computably categorical.

Example

The structure $(\mathbb{N}, <)$, the natural numbers with the usual $<$ order, is not computably categorical.

Computable Dimension

Definition

The computable dimension of $\mathcal{A}$ is the number of computable isomorphism types of $\mathcal{A}$.

Computable Dimension

Definition

The computable dimension of $\mathcal{A}$ is the number of computable isomorphism types of $\mathcal{A}$.

Theorem (Goncharov)

If any two computable pre sentations of a structure $\mathcal{A}$ are isomorphic via a $\emptyset'$-computable function then the computable dimension of $\mathcal{A}$ is either 1 or ω.

Computable Dimension

Definition

The computable dimension of $\mathcal{A}$ is the number of computable isomorphism types of $\mathcal{A}$.

Theorem (Goncharov)

If any two computable pre sentations of a structure $\mathcal{A}$ are isomorphic via a $\emptyset'$-computable function then the computable dimension of $\mathcal{A}$ is either 1 or ω.

Theorem (Goncharov)

For every $n > 1$ there exists a graph of computable dimension n.

Degrees of relations on structures

Definition

The degree spectra of a relation R on a computable structure $\mathcal{A}$ is the set of Turing degrees of the images of R in other computable copies of $\mathcal{A}$.

Degrees of relations on structures

Definition

The degree spectra of a relation R on a computable structure $\mathcal{A}$ is the set of Turing degrees of the images of R in other computable copies of $\mathcal{A}$.

Unlike degree spectra of structures, the degree spectra of relations on structures need not be upward closed in the Turing degrees.

Degrees of relations on structures

Definition

The degree spectra of a relation R on a computable structure $\mathcal{A}$ is the set of Turing degrees of the images of R in other computable copies of $\mathcal{A}$.

Unlike degree spectra of structures, the degree spectra of relations on structures need not be upward closed in the Turing degrees.

Example

The degree of the successivity relation on a computable linear ordering is always computable in $\emptyset'$.

Degrees of relations on structures

Definition

The degree spectra of a relation R on a computable structure $\mathcal{A}$ is the set of Turing degrees of the images of R in other computable copies of $\mathcal{A}$.

Unlike degree spectra of structures, the degree spectra of relations on structures need not be upward closed in the Turing degrees.

Example

The degree of the successivity relation on a computable linear ordering is always computable in $\emptyset'$.

Example

The degree of the reachability relation on a computable graph is always computably enumerable.

Intrinsically computable

Definition

A relation R on a computable structure $\mathcal{A}$ is intrinsically computable if its image is computable on every computable copy of $\mathcal{A}$.

Intrinsically computable

Definition

A relation R on a computable structure $\mathcal{A}$ is intrinsically computable if its image is computable on every computable copy of $\mathcal{A}$.

Definition

A relation R is formally computable on $\mathcal{A}$ if it is definable by a quantifier free formula with finitely many parameters from $\mathcal{A}$.

Intrinsically computable

Definition

A relation R on a computable structure $\mathcal{A}$ is intrinsically computable if its image is computable on every computable copy of $\mathcal{A}$.

Definition

A relation R is formally computable on $\mathcal{A}$ if it is definable by a quantifier free formula with finitely many parameters from $\mathcal{A}$.

It is clear that if R is formally computable on $\mathcal{A}$ then R is intrinsically computable. The converse also holds for many natural structures.

Formally computably enumerable

Definition

A relation R is formally c.e. if it is definable by an effective disjunction of existential formulas with finitely many parameters from $\mathcal{A}$.

Formally computably enumerable

Definition

A relation R is formally c.e. if it is definable by an effective disjunction of existential formulas with finitely many parameters from $\mathcal{A}$.

Example

The reachability relation on a graph $\mathcal{G}$ is formally c.e.

Formally computably enumerable

Definition

A relation R is formally c.e. if it is definable by an effective disjunction of existential formulas with finitely many parameters from $\mathcal{A}$.

Example

The reachability relation on a graph $\mathcal{G}$ is formally c.e.

Theorem (Csima, Khoussainov)

Let $\mathcal{G}$ be a computable graph with all connected components finite, and suppose the function that given each vertex $v \in \mathcal{G}$ computes the size of its component is computable.

Formally computably enumerable

Definition

A relation R is formally c.e. if it is definable by an effective disjunction of existential formulas with finitely many parameters from $\mathcal{A}$.

Example

The reachability relation on a graph $\mathcal{G}$ is formally c.e.

Theorem (Csima, Khoussainov)

Let $\mathcal{G}$ be a computable graph with all connected components finite, and suppose the function that given each vertex $v \in \mathcal{G}$ computes the size of its component is computable.
Then the reachability relation on $\mathcal{G}$ is intrinsically computable if and only if its complement is formally c.e. .

Questions of interest to me

In recent work with Valentina Harizanov, Russell Miller and Antonio Montalbán we have investigated the relationship between spectra of structures and the degree spectra of relations on structures.

Questions of interest to me

In recent work with Valentina Harizanov, Russell Miller and Antonio Montalbán we have investigated the relationship between spectra of structures and the degree spectra of relations on structures.

Particularly, a Fraïssé limit is a universal structure into which all countable structures of a theory embed.

Example

Countable dense linear ordering; random graph; countable atomless boolean algebra

Questions of interest to me

In recent work with Valentina Harizanov, Russell Miller and Antonio Montalbán we have investigated the relationship between spectra of structures and the degree spectra of relations on structures.

Particularly, a Fraïssé limit is a universal structure into which all countable structures of a theory embed.

Example

Countable dense linear ordering; random graph; countable atomless boolean algebra

We showed that under certain conditions the degree spectra of the structure agrees with its degree spectra as a relation after the embedding.

Questions of interest to me

What can the degree spectrum of a unary relation on the linear order $(\mathbb{N}, <)$ look like?

Questions of interest to me

What can the degree spectrum of a unary relation on the linear order $(\mathbb{N}, <)$ look like?

What does the degree spectrum of the successivity relation on an arbitrary linear ordering look like?

Questions of interest to me

What can the degree spectrum of a unary relation on the linear order $(\mathbb{N}, <)$ look like?

What does the degree spectrum of the successivity relation on an arbitrary linear ordering look like?

What does the degree spectrum of the reachability relation on an infinite graph look like?

Thank You!