Computing In The Realm Of The Uncountable

Chong Chi Tat 庄志达

National University of Singapore

chongct@nus.edu.sg

Fudan University, 23 August 2021

Generalizing Recursion Theory

Key characteristics of a computable procedure:

- Driven by an algorithm
- Mechanizable via a (Turing) machine

Church-Turing thesis: What is intuitively computable is indeed computable

Motivation for a generalized recursion theory (GRT):

1. The notion of computation should be applicable, in a broad sense, to domains other than $\mathbb{N}$;

Generalizing Recursion Theory

Key characteristics of a computable procedure:

- Driven by an algorithm
- Mechanizable via a (Turing) machine

Church-Turing thesis: What is intuitively computable is indeed computable

Motivation for a generalized recursion theory (GRT):

1. The notion of computation should be applicable, in a broad sense, to domains other than $\mathbb{N}$;

Generalizing Recursion Theory

Key characteristics of a computable procedure:

- Driven by an algorithm
- Mechanizable via a (Turing) machine

Church-Turing thesis: What is intuitively computable is indeed computable

Motivation for a generalized recursion theory (GRT):

1. The notion of computation should be applicable, in a broad sense, to domains other than $\mathbb{N}$;

Generalizing Recursion Theory

Key characteristics of a computable procedure:

- Driven by an algorithm
- Mechanizable via a (Turing) machine

Church-Turing thesis: What is intuitively computable is indeed computable

Motivation for a generalized recursion theory (GRT):

1. The notion of computation should be applicable, in a broad sense, to domains other than $\mathbb{N}$;

Generalizing Recursion Theory

Key characteristics of a computable procedure:

- Driven by an algorithm
- Mechanizable via a (Turing) machine

Church-Turing thesis: What is intuitively computable is indeed computable

Motivation for a generalized recursion theory (GRT):

1. The notion of computation should be applicable, in a broad sense, to domains other than $\mathbb{N}$;

Generalizing Recursion Theory

Key characteristics of a computable procedure:

- Driven by an algorithm
- Mechanizable via a (Turing) machine

Church-Turing thesis: What is intuitively computable is indeed computable

Motivation for a generalized recursion theory (GRT):

1. The notion of computation should be applicable, in a broad sense, to domains other than $\mathbb{N}$;

Generalizing Recursion Theory

Key characteristics of a computable procedure:

- Driven by an algorithm
- Mechanizable via a (Turing) machine

Church-Turing thesis: What is intuitively computable is indeed computable

Motivation for a generalized recursion theory (GRT):

1. The notion of computation should be applicable, in a broad sense, to domains other than $\mathbb{N}$;

Kreisel: Two erroneous views on GRT

1. There is only one correct generalization but it is not clear which.
2. There are possibilities too many to enumerate.

Kreisel's view. There are only a few GRT's which

1. Have the potential to advance classical recursion theory (CRT);
2. Can enhance our understanding of the mathematical character of CRT.

Kreisel: Two erroneous views on GRT

1. There is only one correct generalization but it is not clear which.
2. There are possibilities too many to enumerate.

Kreisel's view. There are only a few GRT's which

1. Have the potential to advance classical recursion theory (CRT);
2. Can enhance our understanding of the mathematical character of CRT.

Kreisel: Two erroneous views on GRT

1. There is only one correct generalization but it is not clear which.
2. There are possibilities too many to enumerate.

Kreisel's view. There are only a few GRT's which

1. Have the potential to advance classical recursion theory (CRT);
2. Can enhance our understanding of the mathematical character of CRT.

Kreisel: Two erroneous views on GRT

1. There is only one correct generalization but it is not clear which.
2. There are possibilities too many to enumerate.

Kreisel's view. There are only a few GRT's which

1. Have the potential to advance classical recursion theory (CRT);
2. Can enhance our understanding of the mathematical character of CRT.

Kreisel: Two erroneous views on GRT

1. There is only one correct generalization but it is not clear which.
2. There are possibilities too many to enumerate.

Kreisel's view. There are only a few GRT's which

1. Have the potential to advance classical recursion theory (CRT);
2. Can enhance our understanding of the mathematical character of CRT.

Kreisel: Two erroneous views on GRT

1. There is only one correct generalization but it is not clear which.
2. There are possibilities too many to enumerate.

Kreisel's view. There are only a few GRT's which

1. Have the potential to advance classical recursion theory (CRT);
2. Can enhance our understanding of the mathematical character of CRT.

Defining characteristics of the central notions of CRT

- R.E. $= \Sigma_1$-definable (in the language of Peano arithmetic)
- Recursive $= \Delta_1$-definable
- "Finite" $=$ coded by a number in $\mathbb{N}$

Guiding principle: These characterizations should be central to any generalization.

Defining characteristics of the central notions of CRT

- R.E. $=\Sigma_1$-definable (in the language of Peano arithmetic)
- Recursive $=\Delta_1$-definable
- "Finite" $=$ coded by a number in $\mathbb{N}$

Guiding principle: These characterizations should be central to any generalization.

Defining characteristics of the central notions of CRT

- R.E. $= \Sigma_1$-definable (in the language of Peano arithmetic)
- Recursive $= \Delta_1$-definable
- "Finite" $=$ coded by a number in $\mathbb{N}$

Guiding principle: These characterizations should be central to any generalization.

Defining characteristics of the central notions of CRT

- R.E. $= \Sigma_1$-definable (in the language of Peano arithmetic)
- Recursive $= \Delta_1$-definable
- "Finite" $=$ coded by a number in $\mathbb{N}$

Guiding principle: These characterizations should be central to any generalization.

Computing in an uncountable domain: Example 1

(Kripke-Platek, Takeuti, Kreisel-Sacks) *Admissible recursion theory:*

$(L_\alpha, \in)$, $\alpha \geq \omega$ a limit ordinal, which satisfies Σ_1-replacement.

- R.E. $= \Sigma_1$, Recursive $= \Delta_1$ (in the language of ZF)
- $K \subset \alpha$ is "finite" if $K \in L_\alpha$

Computing in an uncountable domain: Example 1

(Kripke-Platek, Takeuti, Kreisel-Sacks) *Admissible recursion theory:*

$(L_\alpha, \in)$, $\alpha \geq \omega$ a limit ordinal, which satisfies Σ_1-replacement.

- R.E. $= \Sigma_1$, Recursive $= \Delta_1$ (in the language of ZF)
- $K \subset \alpha$ is "finite" if $K \in L_\alpha$

Computing in an uncountable domain: Example 1

(Kripke-Platek, Takeuti, Kreisel-Sacks) *Admissible recursion theory:*

$(L_\alpha, \in)$, $\alpha \geq \omega$ a limit ordinal, which satisfies Σ_1-replacement.

- R.E. $= \Sigma_1$, Recursive $= \Delta_1$ (in the language of ZF)
- $K \subset \alpha$ is "finite" if $K \in L_\alpha$

Computing in an uncountable domain: Example 1

(Kripke-Platek, Takeuti, Kreisel-Sacks) *Admissible recursion theory:*

$(L_\alpha, \in)$, $\alpha \geq \omega$ a limit ordinal, which satisfies Σ_1-replacement.

- R.E. $= \Sigma_1$, Recursive $= \Delta_1$ (in the language of ZF)
- $K \subset \alpha$ is "finite" if $K \in L_\alpha$

Computing in an uncountable domain: Example 1

- There is a well-developed theory of computation for $(L_\alpha, \in)$
- The notion of an α-degree provides a natural way of calibrating the relative complexity of subsets of α.
- Ideas and methods from α-recursion theory have been adapted to study GRT over nonstandard models of fragments of PA.
- These have been successfully applied to investigate Ramsey type combinatorial problems in reverse mathematics.

Computing in an uncountable domain: Example 1

- There is a well-developed theory of computation for $(L_\alpha, \in)$
- The notion of an α-degree provides a natural way of calibrating the relative complexity of subsets of α.
- Ideas and methods from α-recursion theory have been adapted to study GRT over nonstandard models of fragments of PA.
- These have been successfully applied to investigate Ramsey type combinatorial problems in reverse mathematics.

Computing in an uncountable domain: Example 1

- There is a well-developed theory of computation for $(L_\alpha, \in)$
- The notion of an α-degree provides a natural way of calibrating the relative complexity of subsets of α.
- Ideas and methods from α-recursion theory have been adapted to study GRT over nonstandard models of fragments of PA.
- These have been successfully applied to investigate Ramsey type combinatorial problems in reverse mathematics.

Computing in an uncountable domain: Example 1

- There is a well-developed theory of computation for $(L_\alpha, \in)$
- The notion of an α-degree provides a natural way of calibrating the relative complexity of subsets of α.
- Ideas and methods from α-recursion theory have been adapted to study GRT over nonstandard models of fragments of PA.
- These have been successfully applied to investigate Ramsey type combinatorial problems in reverse mathematics.

Computing in an uncountable domain: Example 1

- There is a well-developed theory of computation for $(L_\alpha, \in)$
- The notion of an α-degree provides a natural way of calibrating the relative complexity of subsets of α.
- Ideas and methods from α-recursion theory have been adapted to study GRT over nonstandard models of fragments of PA.
- These have been successfully applied to investigate Ramsey type combinatorial problems in reverse mathematics.

Computing in an uncountable domain: Example 1

Some striking diferences with $\mathbb{N}$:

An r.e. set is *maximal* if it is maximal in the lattice of r.e. sets modulo finite sets.

- (Martin) In $\mathbb{N}$, an r.e. degree **a** contains a maximal set iff it is high, i.e. $\mathbf{a}' = \mathbf{0}''$.
- (Leman-Simpson) If α is an uncountable admissible ordinal, then there is no maximal set in $(L_\alpha, \in)$.
- Thus "maximality", despite its definition, is a countability notion.

Computing in an uncountable domain: Example 1

Some striking diferences with $\mathbb{N}$:

An r.e. set is *maximal* if it is maximal in the lattice of r.e. sets modulo finite sets.

- (Martin) In $\mathbb{N}$, an r.e. degree **a** contains a maximal set iff it is high, i.e. $\mathbf{a}' = \mathbf{0}''$.
- (Leman-Simpson) If α is an uncountable admissible ordinal, then there is no maximal set in $(L_\alpha, \in)$.
- Thus "maximality", despite its definition, is a countability notion.

Computing in an uncountable domain: Example 1

Some striking diferences with $\mathbb{N}$:

An r.e. set is *maximal* if it is maximal in the lattice of r.e. sets modulo finite sets.

- (Martin) In $\mathbb{N}$, an r.e. degree **a** contains a maximal set iff it is high, i.e. $\mathbf{a}' = \mathbf{0}''$.
- (Leman-Simpson) If α is an uncountable admissible ordinal, then there is no maximal set in $(L_\alpha, \in)$.
- Thus "maximality", despite its definition, is a countability notion.

Computing in an uncountable domain: Example 1

Some striking diferences with $\mathbb{N}$:

An r.e. set is *maximal* if it is maximal in the lattice of r.e. sets modulo finite sets.

- (Martin) In $\mathbb{N}$, an r.e. degree **a** contains a maximal set iff it is high, i.e. $\mathbf{a}' = \mathbf{0}''$.
- (Leman-Simpson) If α is an uncountable admissible ordinal, then there is no maximal set in $(L_\alpha, \in)$.
- Thus "maximality", despite its definition, is a countability notion.

Computing in an uncountable domain: Example 1

Some striking diferences with $\mathbb{N}$:

An r.e. set is *maximal* if it is maximal in the lattice of r.e. sets modulo finite sets.

- (Martin) In $\mathbb{N}$, an r.e. degree **a** contains a maximal set iff it is high, i.e. $\mathbf{a}' = \mathbf{0}''$.
- (Leman-Simpson) If α is an uncountable admissible ordinal, then there is no maximal set in $(L_\alpha, \in)$.
- Thus "maximality", despite its definition, is a countability notion.

Computing in an uncountable domain: Example 1

- (Sacks) If $\alpha \geq \omega$ is countable and admissible, then $\alpha = \omega_1^T$ for some $T \subseteq \omega$, i.e. α is the least ordinal for which $(L_\alpha[T], \in)$ is admissible.
- (S. Friedman) If $\kappa > \omega$ is a regular cardinal, then there exist admissibles $\kappa < \alpha < \kappa^+$ such that *no* $X \subset \kappa$ satisfies "$(L_\alpha[X], \in)$ is admissible".

Computing in an uncountable domain: Example 1

- (Sacks) If $\alpha \geq \omega$ is countable and admissible, then $\alpha = \omega_1^T$ for some $T \subseteq \omega$, i.e. α is the least ordinal for which $(L_\alpha[T], \in)$ is admissible.
- (S. Friedman) If $\kappa > \omega$ is a regular cardinal, then there exist admissibles $\kappa < \alpha < \kappa^+$ such that *no* $X \subset \kappa$ satisfies "$(L_\alpha[X], \in)$ is admissible".

Computing in an uncountable domain: Example 1

- (Sacks) If $\alpha \geq \omega$ is countable and admissible, then $\alpha = \omega_1^T$ for some $T \subseteq \omega$, i.e. α is the least ordinal for which $(L_\alpha[T], \in)$ is admissible.
- (S. Friedman) If $\kappa > \omega$ is a regular cardinal, then there exist admissibles $\kappa < \alpha < \kappa^+$ such that *no* $X \subset \kappa$ satisfies "$(L_\alpha[X], \in)$ is admissible".

Computing in an uncountable domain: Example 1

In $\mathbb{N}$, the following holds for Turing degrees:

$$(*)\ \ \forall \mathbf{a} \geq \mathbf{0}' \exists \mathbf{b_1}, \mathbf{b_2} > \mathbf{a} (\mathbf{b_1}, \mathbf{b_2} \text{ are incomparable})$$

- (S. Friedman) Assume $V = L$. The Turing degrees in $(L_{\omega_{\omega_1}}, \in)$ above $\mathbf{0}'$ are well-ordered, with successor being the Turing jump.
- (Harrington, Solovay) There exist incomparable $\aleph_\omega$-degrees above $\mathbf{0}'$.
- In general, $(*)$ holds for regular cardinals and singular cardinals κ of countable cofinality, and is false otherwise.
- In fact for such κ, if $V = L$ or if V is a forcing extension of L that preserves *GCH* at κ, then there is a $\mathbf{d} \geq \mathbf{0}'$ such that

  $$\{\mathbf{e} : \mathbf{e} \geq \mathbf{d}\} \text{ are well-ordered with Turing jump as successor.}$$

Hence $(*)$ is a property about countable cofinality.

Computing in an uncountable domain: Example 1

In $\mathbb{N}$, the following holds for Turing degrees:

$$(*)\ \ \forall \mathbf{a} \geq \mathbf{0}' \exists \mathbf{b_1}, \mathbf{b_2} > \mathbf{a} (\mathbf{b_1}, \mathbf{b_2} \text{ are incomparable})$$

- (S. Friedman) Assume $V = L$. The Turing degrees in $(L_{\omega_{\omega_1}}, \in)$ above $\mathbf{0}'$ are well-ordered, with successor being the Turing jump.
- (Harrington, Solovay) There exist incomparable $\aleph_\omega$-degrees above $\mathbf{0}'$.
- In general, $(*)$ holds for regular cardinals and singular cardinals κ of countable cofinality, and is false otherwise.
- In fact for such κ, if $V = L$ or if V is a forcing extension of L that preserves GCH at κ, then there is a $\mathbf{d} \geq \mathbf{0}'$ such that

  $\{\mathbf{e} : \mathbf{e} \geq \mathbf{d}\}$ are well-ordered with Turing jump as successor.

Hence $(*)$ is a property about countable cofinality.

Computing in an uncountable domain: Example 1

In $\mathbb{N}$, the following holds for Turing degrees:

$$(*)\ \ \forall \mathbf{a} \geq \mathbf{0}' \exists \mathbf{b_1}, \mathbf{b_2} > \mathbf{a} (\mathbf{b_1}, \mathbf{b_2} \text{ are incomparable})$$

- (S. Friedman) Assume $V = L$. The Turing degrees in $(L_{\omega_{\omega_1}}, \in)$ above $\mathbf{0}'$ are well-ordered, with successor being the Turing jump.
- (Harrington, Solovay) There exist incomparable $\aleph_\omega$-degrees above $\mathbf{0}'$.
- In general, $(*)$ holds for regular cardinals and singular cardinals κ of countable cofinality, and is false otherwise.
- In fact for such κ, if $V = L$ or if V is a forcing extension of L that preserves GCH at κ, then there is a $\mathbf{d} \geq \mathbf{0}'$ such that

$$\{\mathbf{e} : \mathbf{e} \geq \mathbf{d}\} \text{ are well-ordered with Turing jump as successor.}$$

Hence $(*)$ is a property about countable cofinality.

Computing in an uncountable domain: Example 1

In $\mathbb{N}$, the following holds for Turing degrees:

$$(*)\ \ \forall \mathbf{a} \geq \mathbf{0}' \exists \mathbf{b_1}, \mathbf{b_2} > \mathbf{a} (\mathbf{b_1}, \mathbf{b_2} \text{ are incomparable})$$

- (S. Friedman) Assume $V = L$. The Turing degrees in $(L_{\omega_{\omega_1}}, \in)$ above $\mathbf{0}'$ are well-ordered, with successor being the Turing jump.
- (Harrington, Solovay) There exist incomparable $\aleph_\omega$-degrees above $\mathbf{0}'$.
- In general, $(*)$ holds for regular cardinals and singular cardinals κ of countable cofinality, and is false otherwise.
- In fact for such κ, if $V = L$ or if V is a forcing extension of L that preserves GCH at κ, then there is a $\mathbf{d} \geq \mathbf{0}'$ such that

$$\{\mathbf{e} : \mathbf{e} \geq \mathbf{d}\} \text{ are well-ordered with Turing jump as successor.}$$

Hence $(*)$ is a property about countable cofinality.

Computing in an uncountable domain: Example 1

In $\mathbb{N}$, the following holds for Turing degrees:

$$(*)\ \ \forall \mathbf{a} \geq \mathbf{0}' \exists \mathbf{b_1}, \mathbf{b_2} > \mathbf{a} (\mathbf{b_1}, \mathbf{b_2} \text{ are incomparable})$$

- (S. Friedman) Assume $V = L$. The Turing degrees in $(L_{\omega_{\omega_1}}, \in)$ above $\mathbf{0}'$ are well-ordered, with successor being the Turing jump.
- (Harrington, Solovay) There exist incomparable $\aleph_\omega$-degrees above $\mathbf{0}'$.
- In general, $(*)$ holds for regular cardinals and singular cardinals κ of countable cofinality, and is false otherwise.
- In fact for such κ, if $V = L$ or if V is a forcing extension of L that preserves *GCH* at κ, then there is a $\mathbf{d} \geq \mathbf{0}'$ such that

  $\{\mathbf{e} : \mathbf{e} \geq \mathbf{d}\}$ are well-ordered with Turing jump as successor.

Hence $(*)$ is a property about countable cofinality.

Computing in an uncountable domain: Example 1

In $\mathbb{N}$, the following holds for Turing degrees:

$$(*)\ \ \forall \mathbf{a} \geq \mathbf{0}' \exists \mathbf{b_1}, \mathbf{b_2} > \mathbf{a} (\mathbf{b_1}, \mathbf{b_2} \text{ are incomparable})$$

- (S. Friedman) Assume $V = L$. The Turing degrees in $(L_{\omega_{\omega_1}}, \in)$ above $\mathbf{0}'$ are well-ordered, with successor being the Turing jump.
- (Harrington, Solovay) There exist incomparable $\aleph_\omega$-degrees above $\mathbf{0}'$.
- In general, $(*)$ holds for regular cardinals and singular cardinals κ of countable cofinality, and is false otherwise.
- In fact for such κ, if $V = L$ or if V is a forcing extension of L that preserves *GCH* at κ, then there is a $\mathbf{d} \geq \mathbf{0}'$ such that

  $\{\mathbf{e} : \mathbf{e} \geq \mathbf{d}\}$ are well-ordered with Turing jump as successor.

Hence $(*)$ is a property about countable cofinality.

Computing in an uncountable domain: Example 1

The statement

- $\{\mathbf{0^{(n)}} : n \in \omega\}$ has a least upper bound

is false in $\mathbb{N}$ but true in $(L_{\omega_1}, \in)$ under the assumption $2^\omega \subset L$.

Question. How much do these results reflect the true nature of computation in the uncountable realm?

For example,

- Can the $\aleph_{\omega_1}$-degrees above $\mathbf{0}'$ be indeed well-ordered with Turing jump as the successor, if "$V =$ Ultimate L" (assuming GCH holds at $\aleph_{\omega_1}$)?
- How much computation theory can one develop over an uncountable domain which is not endowed with an effective well-ordering?

Computing in an uncountable domain: Example 1

The statement

- $\{\mathbf{0^{(n)}} : n \in \omega\}$ has a least upper bound

is false in $\mathbb{N}$ but true in $(L_{\omega_1}, \in)$ under the assumption $2^\omega \subset L$.

Question. How much do these results reflect the true nature of computation in the uncountable realm?

For example,

- Can the $\aleph_{\omega_1}$-degrees above $\mathbf{0}'$ be indeed well-ordered with Turing jump as the successor, if "$V =$ Ultimate L" (assuming GCH holds at $\aleph_{\omega_1}$)?
- How much computation theory can one develop over an uncountable domain which is not endowed with an effective well-ordering?

Computing in an uncountable domain: Example 1

The statement

- $\{\mathbf{0^{(n)}} : n \in \omega\}$ has a least upper bound

is false in $\mathbb{N}$ but true in $(L_{\omega_1}, \in)$ under the assumption $2^\omega \subset L$.

Question. How much do these results reflect the true nature of computation in the uncountable realm?

For example,

- Can the $\aleph_{\omega_1}$-degrees above $\mathbf{0}'$ be indeed well-ordered with Turing jump as the successor, if "$V =$ Ultimate L" (assuming GCH holds at $\aleph_{\omega_1}$)?
- How much computation theory can one develop over an uncountable domain which is not endowed with an effective well-ordering?

Computing in an uncountable domain: Example 1

The statement

- $\{\mathbf{0^{(n)}} : n \in \omega\}$ has a least upper bound

is false in $\mathbb{N}$ but true in $(L_{\omega_1}, \in)$ under the assumption $2^\omega \subset L$.

Question. How much do these results reflect the true nature of computation in the uncountable realm?

For example,

- Can the $\aleph_{\omega_1}$-degrees above $\mathbf{0}'$ be indeed well-ordered with Turing jump as the successor, if "$V =$ Ultimate L" (assuming GCH holds at $\aleph_{\omega_1}$)?
- How much computation theory can one develop over an uncountable domain which is not endowed with an effective well-ordering?

Computing in an uncountable domain: Example 2

Computation over $\mathbb{C}$

Questions.

1. What is the "correct" notion of a basic unit in $\mathbb{C}$?
2. What operations on $\mathbb{C}$ are "computable"?
3. When is $X \subset \mathbb{C}$ a "finite" set?

Consider three instances:

- The ω_1-recursion theory approach
- The computable analysis perspective
- The Blum-Shub-Smale model

Computing in an uncountable domain: Example 2

Computation over $\mathbb{C}$

Questions.

1. What is the "correct" notion of a basic unit in $\mathbb{C}$?
2. What operations on $\mathbb{C}$ are "computable"?
3. When is $X \subset \mathbb{C}$ a "finite" set?

Consider three instances:

- The ω_1-recursion theory approach
- The computable analysis perspective
- The Blum-Shub-Smale model

Computing in an uncountable domain: Example 2

Computation over $\mathbb{C}$

Questions.

1. What is the “correct” notion of a basic unit in $\mathbb{C}$?
2. What operations on $\mathbb{C}$ are “computable”?
3. When is $X \subset \mathbb{C}$ a “finite” set?

Consider three instances:

- The ω_1-recursion theory approach
- The computable analysis perspective
- The Blum-Shub-Smale model

Computing in an uncountable domain: Example 2

Computation over $\mathbb{C}$

Questions.

1. What is the “correct” notion of a basic unit in $\mathbb{C}$?
2. What operations on $\mathbb{C}$ are “computable”?
3. When is $X \subset \mathbb{C}$ a “finite” set?

Consider three instances:

- The ω_1-recursion theory approach
- The computable analysis perspective
- The Blum-Shub-Smale model

Computing in an uncountable domain: Example 2

Computation over $\mathbb{C}$

Questions.

1. What is the "correct" notion of a basic unit in $\mathbb{C}$?
2. What operations on $\mathbb{C}$ are "computable"?
3. When is $X \subset \mathbb{C}$ a "finite" set?

Consider three instances:

- The ω_1-recursion theory approach
- The computable analysis perspective
- The Blum-Shub-Smale model

Computing in an uncountable domain: Example 2

Computation over $\mathbb{C}$

Questions.

1. What is the "correct" notion of a basic unit in $\mathbb{C}$?
2. What operations on $\mathbb{C}$ are "computable"?
3. When is $X \subset \mathbb{C}$ a "finite" set?

Consider three instances:

- The ω_1-recursion theory approach
- The computable analysis perspective
- The Blum-Shub-Smale model

Computing in an uncountable domain: Example 2

Computation over $\mathbb{C}$

Questions.

1. What is the “correct” notion of a basic unit in $\mathbb{C}$?
2. What operations on $\mathbb{C}$ are “computable”?
3. When is $X \subset \mathbb{C}$ a “finite” set?

Consider three instances:

- The ω_1-recursion theory approach
- The computable analysis perspective
- The Blum-Shub-Smale model

Computing in an uncountable domain: Example 2

Computation over $\mathbb{C}$

Questions.

1. What is the “correct” notion of a basic unit in $\mathbb{C}$?
2. What operations on $\mathbb{C}$ are “computable”?
3. When is $X \subset \mathbb{C}$ a “finite” set?

Consider three instances:

- The ω_1-recursion theory approach
- The computable analysis perspective
- The Blum-Shub-Smale model

Computing in an uncountable domain: Example 2

Computation over $\mathbb{C}$

Questions.

1. What is the "correct" notion of a basic unit in $\mathbb{C}$?
2. What operations on $\mathbb{C}$ are "computable"?
3. When is $X \subset \mathbb{C}$ a "finite" set?

Consider three instances:

- The ω_1-recursion theory approach
- The computable analysis perspective
- The Blum-Shub-Smale model

Computing in an uncountable domain: Example 2

Complex dynamics of quadratic polynomials:
For $c \in \mathbb{C}$, let

$f_c(z) = z^2 + c$
$f_c^{(n+1)}(z) = f_c(f_c^{(n)}(z))$

The filled Julia set of f_c is

$$K_c = \{z : \lim_{n\to\infty} f_c^{(n)}(z) \not\to \infty\}$$

And $J_c = \partial K_c$ is the Julia set of f_c.

Computing in an uncountable domain: Example 2

Complex dynamics of quadratic polynomials:
For $c \in \mathbb{C}$, let

$$f_c(z) = z^2 + c$$
$$f_c^{(n+1)}(z) = f_c(f_c^{(n)}(z))$$

The filled Julia set of f_c is

$$K_c = \{z : \lim_{n\to\infty} f_c^{(n)}(z) \not\to \infty\}$$

And $J_c = \partial K_c$ is the Julia set of f_c.

Computing in an uncountable domain: Example 2

Complex dynamics of quadratic polynomials:
For $c \in \mathbb{C}$, let

$$f_c(z) = z^2 + c$$
$$f_c^{(n+1)}(z) = f_c(f_c^{(n)}(z))$$

The filled Julia set of f_c is

$$K_c = \{z : \lim_{n \to \infty} f_c^{(n)}(z) \not\to \infty\}$$

And $J_c = \partial K_c$ is the Julia set of f_c.

Computing in an uncountable domain: Example 2

Problems: Study

1. The "computability" of J_c;
2. Relative complexity of Julia sets;
3. "Iterated jumps" of a Julia set.

Computing in an uncountable domain: Example 2

Problems: Study

1. The "computability" of J_c;
2. Relative complexity of Julia sets;
3. "Iterated jumps" of a Julia set.

Computing in an uncountable domain: Example 2

Problems: Study

1. The "computability" of J_c;
2. Relative complexity of Julia sets;
3. "Iterated jumps" of a Julia set.

ω_1^L-recursion theory approach

In $L_{\omega_1^L}$,

- Every $z \in \mathbb{C}$ is "given", hence computable.
- Every $J_c \subset L_{\omega_1^L}$ is also computable.
- Hence all J_c's have the same trivial ω_1^L-degree.

Computability questions regarding J_c are therefore not interesting.

ω_1^L-recursion theory approach

In $L_{\omega_1^L}$,

- Every $z \in \mathbb{C}$ is "given", hence computable.
- Every $J_c \subset L_{\omega_1^L}$ is also computable.
- Hence all J_c's have the same trivial ω_1^L-degree.

Computability questions regarding J_c are therefore not interesting.

ω_1^L-recursion theory approach

In $L_{\omega_1^L}$,

- Every $z \in \mathbb{C}$ is "given", hence computable.
- Every $J_c \subset L_{\omega_1^L}$ is also computable.
- Hence all J_c's have the same trivial ω_1^L-degree.

Computability questions regarding J_c are therefore not interesting.

ω_1^L-recursion theory approach

In $L_{\omega_1^L}$,

- Every $z \in \mathbb{C}$ is “given”, hence computable.
- Every $J_c \subset L_{\omega_1^L}$ is also computable.
- Hence all J_c's have the same trivial ω_1^L-degree.

Computability questions regarding J_c are therefore not interesting.

Computable analysis approach

Define $z \in \mathbb{C}$ to be computable if it has a recursive approximation.

Define $X \subset \mathbb{C}$ to be "computable" if it has a dense subset with each member having a recursive approximation.

- Thus, X is computable if it can be "drawn on the computer" with any prescribed precision.
- It does not provide an algorithm that decides, in finite time, whether a given z is a member of X.

(Braverman and Yampolsky) *Outcomes:*

- All hyperbolic and parabolic J_c's are computable (relative to c).
- There is a Siegel disc with a computable c such that J_c is not computable.

Computable analysis approach

Define $z \in \mathbb{C}$ to be computable if it has a recursive approximation.
Define $X \subset \mathbb{C}$ to be "computable" if it has a dense subset with each member having a recursive approximation.

- Thus, X is computable if it can be "drawn on the computer" with any prescribed precision.
- It does not provide an algorithm that decides, in finite time, whether a given z is a member of X.

(Braverman and Yampolsky) *Outcomes:*

- All hyperbolic and parabolic J_c's are computable (relative to c).
- There is a Siegel disc with a computable c such that J_c is not computable.

Computable analysis approach

Define $z \in \mathbb{C}$ to be computable if it has a recursive approximation.
Define $X \subset \mathbb{C}$ to be "computable" if it has a dense subset with each member having a recursive approximation.

- Thus, X is computable if it can be "drawn on the computer" with any prescribed precision.
- It does not provide an algorithm that decides, in finite time, whether a given z is a member of X.

(Braverman and Yampolsky) *Outcomes:*

- All hyperbolic and parabolic J_c's are computable (relative to c).
- There is a Siegel disc with a computable c such that J_c is not computable.

Computable analysis approach

Define $z \in \mathbb{C}$ to be computable if it has a recursive approximation.
Define $X \subset \mathbb{C}$ to be "computable" if it has a dense subset with each member having a recursive approximation.

- Thus, X is computable if it can be "drawn on the computer" with any prescribed precision.
- It does not provide an algorithm that decides, in finite time, whether a given z is a member of X.

(Braverman and Yampolsky) *Outcomes:*

- All hyperbolic and parabolic J_c's are computable (relative to c).
- There is a Siegel disc with a computable c such that J_c is not computable.

Computable analysis approach

Define $z \in \mathbb{C}$ to be computable if it has a recursive approximation.
Define $X \subset \mathbb{C}$ to be "computable" if it has a dense subset with each member having a recursive approximation.

- Thus, X is computable if it can be "drawn on the computer" with any prescribed precision.
- It does not provide an algorithm that decides, in finite time, whether a given z is a member of X.

(Braverman and Yampolsky) *Outcomes:*

- All hyperbolic and parabolic J_c's are computable (relative to c).
- There is a Siegel disc with a computable c such that J_c is not computable.

Computable analysis approach

Define $z \in \mathbb{C}$ to be computable if it has a recursive approximation.
Define $X \subset \mathbb{C}$ to be "computable" if it has a dense subset with each member having a recursive approximation.

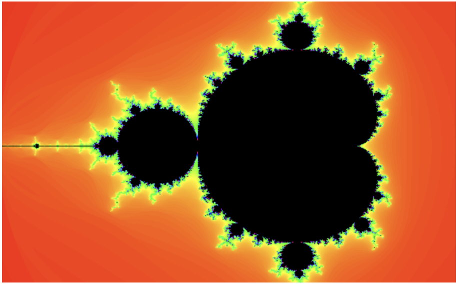
- Thus, X is computable if it can be "drawn on the computer" with any prescribed precision.
- It does not provide an algorithm that decides, in finite time, whether a given z is a member of X.

(Braverman and Yampolsky) *Outcomes:*

- All hyperbolic and parabolic J_c's are computable (relative to c).
- There is a Siegel disc with a computable c such that J_c is not computable.

Computable analysis approach

- For each r.e. degree in $\mathbb{N}$, there is a J_c of that degree.
- (Hertling) If the hyperbolicity conjecture* is true, then the Mandelbrot set is computable.



Mandelbrot set $M = \{c : J_c \text{ is connected}\}$

**Hyperbolicity Conjecture*: Hyperbolic J_c's are dense in M.

Computable analysis approach

However, from the viewpoint of computation theory, there are issues to be addressed.

1. In the language of PA, "computable" is not Δ_1 but Δ_2 definable.
2. Even if J_c is computable, there is no algorithm to decide, in finite time, if a given $z \in \mathbb{C}$ is in J_c.
3. More generally, given a (real) finite set X, there is no procedure to decide in finite time, if $X \subset J_c$ or $X \cap J_c = \emptyset$ for such X.

Computable analysis approach

However, from the viewpoint of computation theory, there are issues to be addressed.

1. In the language of PA, "computable" is not Δ_1 but Δ_2 definable.
2. Even if J_c is computable, there is no algorithm to decide, in finite time, if a given $z \in \mathbb{C}$ is in J_c.
3. More generally, given a (real) finite set X, there is no procedure to decide in finite time, if $X \subset J_c$ or $X \cap J_c = \emptyset$ for such X.

Computable analysis approach

However, from the viewpoint of computation theory, there are issues to be addressed.

1. In the language of PA, "computable" is not Δ_1 but Δ_2 definable.
2. Even if J_c is computable, there is no algorithm to decide, in finite time, if a given $z \in \mathbb{C}$ is in J_c.
3. More generally, given a (real) finite set X, there is no procedure to decide in finite time, if $X \subset J_c$ or $X \cap J_c = \emptyset$ for such X.

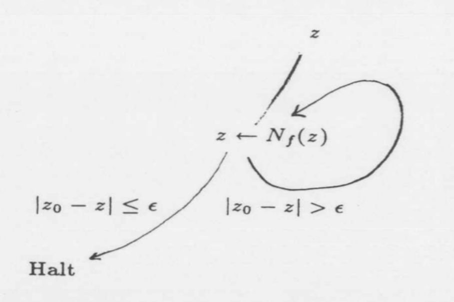
Computable analysis approach

However, from the viewpoint of computation theory, there are issues to be addressed.

1. In the language of PA, “computable” is not Δ_1 but Δ_2 definable.
2. Even if J_c is computable, there is no algorithm to decide, in finite time, if a given $z \in \mathbb{C}$ is in J_c.
3. More generally, given a (real) finite set X, there is no procedure to decide in finite time, if $X \subset J_c$ or $X \cap J_c = \emptyset$ for such X.

Blum-Shub-Smale (BSS) model

- Every $z \in \mathbb{C}$ is a basic unit of the model.
- Rational maps are basic operations.
- A computation or an algorithm can be viewed as a ‘flowchart”



BSS model

- R.E $= \Sigma_1$ in a two-sorted language.
- $X \subset \mathbb{C}$ is "computable" if there is an algorithm that decides, for each z, whether $z \in X$ in (real) finite time.
- A set X is computable iff "$z \in X$" is Δ_1.

BSS model

- R.E $= \Sigma_1$ in a two-sorted language.
- $X \subset \mathbb{C}$ is "computable" if there is an algorithm that decides, for each z, whether $z \in X$ in (real) finite time.
- A set X is computable iff "$z \in X$" is Δ_1.

BSS model

- R.E $= \Sigma_1$in a two-sorted language.
- $X \subset \mathbb{C}$ is "computable" if there is an algorithm that decides, for each z, whether $z \in X$ in (real) finite time.
- A set X is computable iff "$z \in X$" is Δ_1.

BSS model

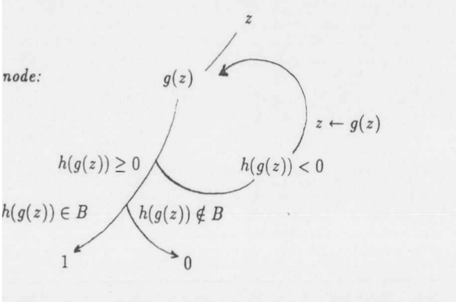
- (BSS) The Mandelbrot set is not computable.
- (BSS) Except for trivial caes such as $c = 0$ (where $J_c =$ unit circle), all J_c's are not computable.

BSS model

- (BSS) The Mandelbrot set is not computable.
- (BSS) Except for trivial caes such as $c = 0$ (where $J_c =$ unit circle), all J_c's are not computable.

BSS model

One can define a notion of *pointwise* Turing reducibility $\leq_{wT}$, allowing queries of the form “Is $z \in J_d$?” in the flowchart:



BSS model

This “weak” reducibility notion offers a way to investigae the relative complexity of Julia sets:

- (Chong) There exist $c \neq d$ such that J_c and J_d are incomparable under $\leq_{wT}$.
- (Chong) If J is the Julia set of a rational map which is locally connected and K_c° has more than one compoinent, then $\emptyset <_{wT} < K_c^\circ <_{wT} J_c$.

However, pointwise computability and reducibility do not capture the uncountability aspect of $\mathbb{C}$ since only information about finite subsets are needed for a decision.

BSS model

This “weak” reducibility notion offers a way to investigae the relative complexity of Julia sets:

- (Chong) There exist $c \neq d$ such that J_c and J_d are incomparable under $\leq_{wT}$.
- (Chong) If J is the Julia set of a rational map which is locally connected and K_c° has more than one compoinent, then $\emptyset <_{wT} < K_c^\circ <_{wT} J_c$.

However, pointwise computability and reducibility do not capture the uncountability aspect of $\mathbb{C}$ since only information about finite subsets are needed for a decision.

BSS model

This "weak" reducibility notion offers a way to investigae the relative complexity of Julia sets:

- (Chong) There exist $c \neq d$ such that J_c and J_d are incomparable under $\leq_{wT}$.
- (Chong) If J is the Julia set of a rational map which is locally connected and K_c° has more than one compoinent, then $\emptyset <_{wT} < K_c^\circ <_{wT} J_c$.

However, pointwise computability and reducibility do not capture the uncountability aspect of $\mathbb{C}$ since only information about finite subsets are needed for a decision.

BSS model

This "weak" reducibility notion offers a way to investigae the relative complexity of Julia sets:

- (Chong) There exist $c \neq d$ such that J_c and J_d are incomparable under $\leq_{wT}$.
- (Chong) If J is the Julia set of a rational map which is locally connected and K_c° has more than one compoinent, then $\emptyset <_{wT} < K_c^\circ <_{wT} J_c$.

However, pointwise computability and reducibility do not capture the uncountability aspect of $\mathbb{C}$ since only information about finite subsets are needed for a decision.

BSS model

This "weak" reducibility notion offers a way to investigae the relative complexity of Julia sets:

- (Chong) There exist $c \neq d$ such that J_c and J_d are incomparable under $\leq_{wT}$.
- (Chong) If J is the Julia set of a rational map which is locally connected and K_c° has more than one compoinent, then $\emptyset <_{wT} < K_c^\circ <_{wT} J_c$.

However, pointwise computability and reducibility do not capture the uncountability aspect of $\mathbb{C}$ since only information about finite subsets are needed for a decision.

BSS model

Questions: What is the "correct" noion of

1. "Finiteness" in $\mathbb{C}$?
2. Recursive/computable set?
3. Turing reducibility $\leq_T$ in this model?

Proposal. Take the cue from α-recursion theory and focus on definability:

- $X \subset \mathbb{C}$ is finite iff it is bounded and Δ_1 definable.
- $A \subset \mathbb{C}$ is recursive iff for each finite set X, "$X \subset A$" and "$X \subset \mathbb{C} \setminus A$" are Σ_1 definable.
- $A \leq_T B$ (A is Turing reducible to B) iff there is an algorithm to decide (in real finite time), for each finite X, if $X \subset A$ or $X \subset \mathbb{C} \setminus A$ using finite information about B.

BSS model

Questions: What is the "correct" noion of

1. "Finiteness" in $\mathbb{C}$?
2. Recursive/computable set?
3. Turing reducibility $\leq_T$ in this model?

Proposal. Take the cue from α-recursion theory and focus on definability:

- $X \subset \mathbb{C}$ is finite iff it is bounded and Δ_1 definable.
- $A \subset \mathbb{C}$ is recursive iff for each finite set X, "$X \subset A$" and "$X \subset \mathbb{C} \setminus A$" are Σ_1 definable.
- $A \leq_T B$ (A is Turing reducible to B) iff there is an algorithm to decide (in real finite time), for each finite X, if $X \subset A$ or $X \subset \mathbb{C} \setminus A$ using finite information about B.

BSS model

Questions: What is the "correct" noion of

1. "Finiteness" in $\mathbb{C}$?
2. Recursive/computable set?
3. Turing reducibility $\leq_T$ in this model?

Proposal. Take the cue from α-recursion theory and focus on definability:

- $X \subset \mathbb{C}$ is finite iff it is bounded and Δ_1 definable.
- $A \subset \mathbb{C}$ is recursive iff for each finite set X, "$X \subset A$" and "$X \subset \mathbb{C} \setminus A$" are Σ_1 definable.
- $A \leq_T B$ (A is Turing reducible to B) iff there is an algorithm to decide (in real finite time), for each finite X, if $X \subset A$ or $X \subset \mathbb{C} \setminus A$ using finite information about B.

BSS model

Questions: What is the "correct" noion of

1. "Finiteness" in $\mathbb{C}$?
2. Recursive/computable set?
3. Turing reducibility $\leq_T$ in this model?

Proposal. Take the cue from α-recursion theory and focus on definability:

- $X \subset \mathbb{C}$ is finite iff it is bounded and Δ_1 definable.
- $A \subset \mathbb{C}$ is recursive iff for each finite set X, "$X \subset A$" and "$X \subset \mathbb{C} \setminus A$" are Σ_1 definable.
- $A \leq_T B$ (A is Turing reducible to B) iff there is an algorithm to decide (in real finite time), for each finite X, if $X \subset A$ or $X \subset \mathbb{C} \setminus A$ using finite information about B.

BSS model

Questions: What is the "correct" noion of

1. "Finiteness" in $\mathbb{C}$?
2. Recursive/computable set?
3. Turing reducibility $\leq_T$ in this model?

Proposal. Take the cue from α-recursion theory and focus on definability:

- $X \subset \mathbb{C}$ is finite iff it is bounded and Δ_1 definable.
- $A \subset \mathbb{C}$ is recursive iff for each finite set X, "$X \subset A$" and "$X \subset \mathbb{C} \setminus A$" are Σ_1 definable.
- $A \leq_T B$ (A is Turing reducible to B) iff there is an algorithm to decide (in real finite time), for each finite X, if $X \subset A$ or $X \subset \mathbb{C} \setminus A$ using finite information about B.

BSS model

Questions: What is the “correct” noion of

1. “Finiteness” in $\mathbb{C}$?
2. Recursive/computable set?
3. Turing reducibility $\leq_T$ in this model?

Proposal. Take the cue from α-recursion theory and focus on definability:

- $X \subset \mathbb{C}$ is finite iff it is bounded and Δ_1 definable.
- $A \subset \mathbb{C}$ is recursive iff for each finite set X, “$X \subset A$” and “$X \subset \mathbb{C} \setminus A$” are Σ_1 definable.
- $A \leq_T B$ (A is Turing reducible to B) iff there is an algorithm to decide (in real finite time), for each finite X, if $X \subset A$ or $X \subset \mathbb{C} \setminus A$ using finite information about B.

BSS model

Questions: What is the "correct" noion of

1. "Finiteness" in $\mathbb{C}$?
2. Recursive/computable set?
3. Turing reducibility $\leq_T$ in this model?

Proposal. Take the cue from α-recursion theory and focus on definability:

- $X \subset \mathbb{C}$ is finite iff it is bounded and Δ_1 definable.
- $A \subset \mathbb{C}$ is recursive iff for each finite set X, "$X \subset A$" and "$X \subset \mathbb{C} \setminus A$" are Σ_1 definable.
- $A \leq_T B$ (A is Turing reducible to B) iff there is an algorithm to decide (in real finite time), for each finite X, if $X \subset A$ or $X \subset \mathbb{C} \setminus A$ using finite information about B.

BSS model

Questions: What is the “correct” noion of

1. “Finiteness” in $\mathbb{C}$?
2. Recursive/computable set?
3. Turing reducibility $\leq_T$ in this model?

Proposal. Take the cue from α-recursion theory and focus on definability:

- $X \subset \mathbb{C}$ is finite iff it is bounded and Δ_1 definable.
- $A \subset \mathbb{C}$ is recursive iff for each finite set X, “$X \subset A$” and “$X \subset \mathbb{C} \setminus A$” are Σ_1 definable.
- $A \leq_T B$ (A is Turing reducible to B) iff there is an algorithm to decide (in real finite time), for each finite X, if $X \subset A$ or $X \subset \mathbb{C} \setminus A$ using finite information about B.

Conclusion: Computing in the uncountable realm

1. Contrary to Kreisel's view, there are many possible GRTs, specific to each application.
2. Each GRT has to take into account the context in which the theory is introduced.
3. There is no universal GRT as any axiomatic approach will likely require supplementary features to be applicable.

Conclusion: Computing in the uncountable realm

1. Contrary to Kreisel's view, there are many possible GRTs, specific to each application.
2. Each GRT has to take into account the context in which the theory is introduced.
3. There is no universal GRT as any axiomatic approach will likely require supplementary features to be applicable.

Conclusion: Computing in the uncountable realm

1. Contrary to Kreisel's view, there are many possible GRTs, specific to each application.
2. Each GRT has to take into account the context in which the theory is introduced.
3. There is no universal GRT as any axiomatic approach will likely require supplementary features to be applicable.

Conclusion: Computing in the uncountable realm

1. Contrary to Kreisel's view, there are many possible GRTs, specific to each application.
2. Each GRT has to take into account the context in which the theory is introduced.
3. There is no universal GRT as any axiomatic approach will likely require supplementary features to be applicable.

Conclusion: Computing in the uncountable realm

- The computable analysis perspective has some foundational issues, but has useful applications in “real world computing”.
- Models such as BSS or ordinal recursion theory are appealing from the foundational point of view, but they are abstractions of our intuition of countable computation.
- Hence,

  How much does, or should, uncountable mathematics reflect reality?

Conclusion: Computing in the uncountable realm

- The computable analysis perspective has some foundational issues, but has useful applications in “real world computing”.
- Models such as BSS or ordinal recursion theory are appealing from the foundational point of view, but they are abstractions of our intuition of countable computation.
- Hence,

  How much does, or should, uncountable mathematics reflect reality?

Conclusion: Computing in the uncountable realm

- The computable analysis perspective has some foundational issues, but has useful applications in "real world computing".
- Models such as BSS or ordinal recursion theory are appealing from the foundational point of view, but they are abstractions of our intuition of countable computation.
- Hence,

  How much does, or should, uncountable mathematics reflect reality?

Question

What is Reality and what is the role of mathematics in it ?