A small step beyond the Turing degrees

Joseph S. Miller
University of Wisconsin—Madison

AMS-ASL Special Session on the Life and Legacy of Alan Turing
Joint Mathematics Meetings, Boston, MA
January 5, 2012

On computable numbers. . . .

Turing begins his seminal 1936 paper "On Computable Numbers, with an Application to the Entscheidungsproblem" as follows:

On computable numbers. . . .

Turing begins his seminal 1936 paper "On Computable Numbers, with an Application to the Entscheidungsproblem" as follows:

> The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means.

On computable numbers. . . .

Turing begins his seminal 1936 paper "On Computable Numbers, with an Application to the Entscheidungsproblem" as follows:

> The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable numbers, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. . . .

On computable numbers. . . .

Turing begins his seminal 1936 paper "On Computable Numbers, with an Application to the Entscheidungsproblem" as follows:

> The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable numbers, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. . . .

Note that:

- Turing was interested, from the beginning, in computability taken broadly (not just sets and function on $\mathbb{N}$).

On computable numbers. . . .

Turing begins his seminal 1936 paper "On Computable Numbers, with an Application to the Entscheidungsproblem" as follows:

> The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable numbers, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. . . .

Note that:

- Turing was interested, from the beginning, in computability taken broadly (not just sets and function on $\mathbb{N}$).
- He defined a real number to be "computable if its decimal can be written down by a machine."

On computable numbers. . . . A correction

The following year, Turing published a three page "Correction", half of which is devoted to

> . . . modifying the manner in which computable numbers are associated with computable sequences, the totality of computable numbers being left unaltered.

On computable numbers. . . . A correction

The following year, Turing published a three page "Correction", half of which is devoted to

> . . . modifying the manner in which computable numbers are associated with computable sequences, the totality of computable numbers being left unaltered.

What is the problem?

On computable numbers. . . . A correction

The following year, Turing published a three page "Correction", half of which is devoted to

> . . . modifying the manner in which computable numbers are associated with computable sequences, the totality of computable numbers being left unaltered.

What is the problem?

In modern terms: say we are given a computable increasing sequence $a_0 \leqslant a_1 \leqslant a_2 \leqslant \cdots$ and a computable decreasing sequence $b_0 \geqslant b_1 \geqslant b_2 \geqslant \cdots$ of rationals such that $\lim a_n = \lim b_n = x$.

On computable numbers. . . . A correction

The following year, Turing published a three page "Correction", half of which is devoted to

> . . . modifying the manner in which computable numbers are associated with computable sequences, the totality of computable numbers being left unaltered.

What is the problem?

In modern terms: say we are given a computable increasing sequence $a_0 \leqslant a_1 \leqslant a_2 \leqslant \cdots$ and a computable decreasing sequence $b_0 \geqslant b_1 \geqslant b_2 \geqslant \cdots$ of rationals such that $\lim a_n = \lim b_n = x$.

Then

1. x is a computable real number.

On computable numbers. . . . A correction

The following year, Turing published a three page "Correction", half of which is devoted to

> . . . modifying the manner in which computable numbers are associated with computable sequences, the totality of computable numbers being left unaltered.

What is the problem?

In modern terms: say we are given a computable increasing sequence $a_0 \leqslant a_1 \leqslant a_2 \leqslant \cdots$ and a computable decreasing sequence $b_0 \geqslant b_1 \geqslant b_2 \geqslant \cdots$ of rationals such that $\lim a_n = \lim b_n = x$.

Then

1. x is a computable real number.
2. We cannot (uniformly) compute the decimal expansion of x.

On computable numbers. . . . A correction

Say you are given the sequences $0.4 < 0.49 < 0.499 < \cdots$ and $0.6 > 0.51 > 0.501 > \cdots$.

On computable numbers. . . . A correction

Say you are given the sequences $0.4 < 0.49 < 0.499 < \cdots$ and $0.6 > 0.51 > 0.501 > \cdots$. You can never decide whether the first decimal digit of x should be 4 or 5,

On computable numbers. . . . A correction

Say you are given the sequences $0.4 < 0.49 < 0.499 < \cdots$ and $0.6 > 0.51 > 0.501 > \cdots$. You can never decide whether the first decimal digit of x should be 4 or 5, unless you (non-uniformly) know that $x = 0.5$.

On computable numbers. . . . A correction

Say you are given the sequences $0.4 < 0.49 < 0.499 < \cdots$ and $0.6 > 0.51 > 0.501 > \cdots$. You can never decide whether the first decimal digit of x should be 4 or 5, unless you (non-uniformly) know that $x = 0.5$.

Other versions of this problem:

On computable numbers. . . . A correction

Say you are given the sequences $0.4 < 0.49 < 0.499 < \cdots$ and $0.6 > 0.51 > 0.501 > \cdots$. You can never decide whether the first decimal digit of x should be 4 or 5, unless you (non-uniformly) know that $x = 0.5$.

Other versions of this problem:

- (Turing) The Euler constant is computable, but we cannot write a machine down for its decimal expansion (because we do not know if it has a finite decimal expansion).

On computable numbers. . . . A correction

Say you are given the sequences $0.4 < 0.49 < 0.499 < \cdots$ and $0.6 > 0.51 > 0.501 > \cdots$. You can never decide whether the first decimal digit of x should be 4 or 5, unless you (non-uniformly) know that $x = 0.5$.

Other versions of this problem:

- (Turing) The Euler constant is computable, but we cannot write a machine down for its decimal expansion (because we do not know if it has a finite decimal expansion).
- There is no machine that, given the decimal expansion of a real number x, produces the decimal expansion of $3x$.

On computable numbers. . . . A correction

Say you are given the sequences $0.4 < 0.49 < 0.499 < \cdots$ and $0.6 > 0.51 > 0.501 > \cdots$. You can never decide whether the first decimal digit of x should be 4 or 5, unless you (non-uniformly) know that $x = 0.5$.

Other versions of this problem:

- (Turing) The Euler constant is computable, but we cannot write a machine down for its decimal expansion (because we do not know if it has a finite decimal expansion).
- There is no machine that, given the decimal expansion of a real number x, produces the decimal expansion of $3x$.

In each case, the problem is rationals with finite decimal expansions.

On computable numbers. . . . A correction

Say you are given the sequences $0.4 < 0.49 < 0.499 < \cdots$ and $0.6 > 0.51 > 0.501 > \cdots$. You can never decide whether the first decimal digit of x should be 4 or 5, unless you (non-uniformly) know that $x = 0.5$.

Other versions of this problem:

- (Turing) The Euler constant is computable, but we cannot write a machine down for its decimal expansion (because we do not know if it has a finite decimal expansion).
- There is no machine that, given the decimal expansion of a real number x, produces the decimal expansion of $3x$.

In each case, the problem is rationals with finite decimal expansions.

Turing suggested a more convenient representation of real numbers.

On computable numbers. . . . A correction

Say you are given the sequences $0.4 < 0.49 < 0.499 < \cdots$ and $0.6 > 0.51 > 0.501 > \cdots$. You can never decide whether the first decimal digit of x should be 4 or 5, unless you (non-uniformly) know that $x = 0.5$.

Other versions of this problem:

- (Turing) The Euler constant is computable, but we cannot write a machine down for its decimal expansion (because we do not know if it has a finite decimal expansion).
- There is no machine that, given the decimal expansion of a real number x, produces the decimal expansion of $3x$.

In each case, the problem is rationals with finite decimal expansions.

Turing suggested a more convenient representation of real numbers. We give a different (but essentially equivalent) representation:

Redefining computable numbers

Definition

$\lambda\colon \mathbb{Q}^+ \to \mathbb{Q}$ is a *name* of a real $x \in \mathbb{R}$ if for all rationals $\varepsilon > 0$ we have $|\lambda(\varepsilon) - x| < \varepsilon$.

Redefining computable numbers

Definition

$\lambda\colon \mathbb{Q}^+ \to \mathbb{Q}$ is a *name* of a real $x \in \mathbb{R}$ if for all rationals $\varepsilon > 0$ we have $|\lambda(\varepsilon) - x| < \varepsilon$.

Note that names can be easily coded as binary sequences, so they have Turing degree.

Redefining computable numbers

Definition

$\lambda\colon \mathbb{Q}^+ \to \mathbb{Q}$ is a *name* of a real $x \in \mathbb{R}$ if for all rationals $\varepsilon > 0$ we have $|\lambda(\varepsilon) - x| < \varepsilon$.

Note that names can be easily coded as binary sequences, so they have Turing degree.

Definition

$x \in \mathbb{R}$ is *computable* if it has a computable name.

Redefining computable numbers

Definition

$\lambda\colon \mathbb{Q}^+ \to \mathbb{Q}$ is a *name* of a real $x \in \mathbb{R}$ if for all rationals $\varepsilon > 0$ we have $|\lambda(\varepsilon) - x| < \varepsilon$.

Note that names can be easily coded as binary sequences, so they have Turing degree.

Definition

$x \in \mathbb{R}$ is *computable* if it has a computable name.

Facts

- A real x is computable iff it has a computable decimal expansion.

Redefining computable numbers

Definition

$\lambda\colon \mathbb{Q}^+ \to \mathbb{Q}$ is a *name* of a real $x \in \mathbb{R}$ if for all rationals $\varepsilon > 0$ we have $|\lambda(\varepsilon) - x| < \varepsilon$.

Note that names can be easily coded as binary sequences, so they have Turing degree.

Definition

$x \in \mathbb{R}$ is *computable* if it has a computable name.

Facts

- A real x is computable iff it has a computable decimal expansion.
- We can give a machine to compute a name of the Euler constant.

Redefining computable numbers

Definition

$\lambda\colon \mathbb{Q}^+ \to \mathbb{Q}$ is a *name* of a real $x \in \mathbb{R}$ if for all rationals $\varepsilon > 0$ we have $|\lambda(\varepsilon) - x| < \varepsilon$.

Note that names can be easily coded as binary sequences, so they have Turing degree.

Definition

$x \in \mathbb{R}$ is *computable* if it has a computable name.

Facts

- A real x is computable iff it has a computable decimal expansion.
- We can give a machine to compute a name of the Euler constant.
- There is an easy (uniform) transformation from name for x to a name for $3x$.

Redefining computable numbers

Definition

$\lambda\colon \mathbb{Q}^+ \to \mathbb{Q}$ is a *name* of a real $x \in \mathbb{R}$ if for all rationals $\varepsilon > 0$ we have $|\lambda(\varepsilon) - x| < \varepsilon$.

Note that names can be easily coded as binary sequences, so they have Turing degree.

Definition

$x \in \mathbb{R}$ is *computable* if it has a computable name.

Facts

- A real x is computable iff it has a computable decimal expansion.
- We can give a machine to compute a name of the Euler constant.
- There is an easy (uniform) transformation from name for x to a name for $3x$. ($\lambda \mapsto \lambda\varepsilon.3\lambda(\varepsilon/3)$.)

Extending the notion of computability

We can now define computable functions on the real numbers.

Extending the notion of computability

We can now define computable functions on the real numbers.

Definition

A function $f\colon \mathbb{R} \to \mathbb{R}$ is *computable* if there is a computable procedure (Turing machine) that when given any name for an $x \in \mathbb{R}$ produces a name for $f(x)$.

Extending the notion of computability

We can now define computable functions on the real numbers.

Definition

A function $f\colon \mathbb{R} \to \mathbb{R}$ is *computable* if there is a computable procedure (Turing machine) that when given any name for an $x \in \mathbb{R}$ produces a name for $f(x)$.

So $f(x) = 3x$ is computable, as are all of the standard functions from calculus.

Extending the notion of computability

We can now define computable functions on the real numbers.

Definition

A function $f\colon \mathbb{R} \to \mathbb{R}$ is *computable* if there is a computable procedure (Turing machine) that when given any name for an $x \in \mathbb{R}$ produces a name for $f(x)$.

So $f(x) = 3x$ is computable, as are all of the standard functions from calculus.

This is the beginning of the field of *Computable Analysis*, which was studied as far back as the mid-fifties (by Lacombe and Grzegorczyk).

Extending the notion of computability

We can now define computable functions on the real numbers.

Definition

A function $f\colon \mathbb{R} \to \mathbb{R}$ is *computable* if there is a computable procedure (Turing machine) that when given any name for an $x \in \mathbb{R}$ produces a name for $f(x)$.

So $f(x) = 3x$ is computable, as are all of the standard functions from calculus.

This is the beginning of the field of *Computable Analysis*, which was studied as far back as the mid-fifties (by Lacombe and Grzegorczyk).

What about non-computable (continuous) functions? Can the Turing degrees be used to measure the complexity of continuous functions?

Computability of sequences of reals

For simplicity, we focus on sequences of reals instead of continuous functions.

Computability of sequences of reals

For simplicity, we focus on sequences of reals instead of continuous functions.

Definition

$\lambda\colon \mathbb{N} \times \mathbb{Q}^+ \to \mathbb{Q}$ is a *name* of a sequence $S \in [0,1]^{\mathbb{N}}$ if for all $n \in \mathbb{N}$ and all rationals $\varepsilon > 0$ we have $|\lambda(n,\varepsilon) - S(n)| < \varepsilon$.

Computability of sequences of reals

For simplicity, we focus on sequences of reals instead of continuous functions.

Definition

$\lambda\colon \mathbb{N} \times \mathbb{Q}^+ \to \mathbb{Q}$ is a *name* of a sequence $S \in [0,1]^{\mathbb{N}}$ if for all $n \in \mathbb{N}$ and all rationals $\varepsilon > 0$ we have $|\lambda(n,\varepsilon) - S(n)| < \varepsilon$.

Note that names can be easily coded as binary sequences, so have Turing degree.

Computability of sequences of reals

For simplicity, we focus on sequences of reals instead of continuous functions.

Definition

$\lambda\colon \mathbb{N} \times \mathbb{Q}^+ \to \mathbb{Q}$ is a *name* of a sequence $S \in [0,1]^{\mathbb{N}}$ if for all $n \in \mathbb{N}$ and all rationals $\varepsilon > 0$ we have $|\lambda(n,\varepsilon) - S(n)| < \varepsilon$.

Note that names can be easily coded as binary sequences, so have Turing degree.

Definition

$S \in [0,1]^{\mathbb{N}}$ is *computable* if it has a computable name.

Computability of sequences of reals

For simplicity, we focus on sequences of reals instead of continuous functions.

Definition

$\lambda\colon \mathbb{N} \times \mathbb{Q}^+ \to \mathbb{Q}$ is a *name* of a sequence $S \in [0,1]^{\mathbb{N}}$ if for all $n \in \mathbb{N}$ and all rationals $\varepsilon > 0$ we have $|\lambda(n,\varepsilon) - S(n)| < \varepsilon$.

Note that names can be easily coded as binary sequences, so have Turing degree.

Definition

$S \in [0,1]^{\mathbb{N}}$ is *computable* if it has a computable name.

How can we capture the complexity of a non-computable $S \in [0,1]^{\mathbb{N}}$?

Computability of sequences of reals

Question (Steffen Lempp)

Does every sequence $S \in [0,1]^{\mathbb{N}}$ have a least Turing degree name?

Computability of sequences of reals

Question (Steffen Lempp)

Does every sequence $S \in [0,1]^{\mathbb{N}}$ have a least Turing degree name?

The least Turing degree of any name would naturally measure the complexity of S.

Computability of sequences of reals

Question (Steffen Lempp)

Does every sequence $S \in [0,1]^{\mathbb{N}}$ have a least Turing degree name?

The least Turing degree of any name would naturally measure the complexity of S. Call it the *Turing degree* of S.

Computability of sequences of reals

Question (Steffen Lempp)

Does every sequence $S \in [0,1]^{\mathbb{N}}$ have a least Turing degree name?

The least Turing degree of any name would naturally measure the complexity of S. Call it the *Turing degree* of S.

(Actually, Lempp asked the question about names of continuous functions $f\colon [0,1] \to \mathbb{R}$. More on that later.)

Computability of sequences of reals

Question (Steffen Lempp)

Does every sequence $S \in [0,1]^{\mathbb{N}}$ have a least Turing degree name?

The least Turing degree of any name would naturally measure the complexity of S. Call it the *Turing degree* of S.

(Actually, Lempp asked the question about names of continuous functions $f\colon [0,1] \to \mathbb{R}$. More on that later.)

Easy fact

If $S \in [0,1]^{\mathbb{N}}$ contains no rationals with finite decimal expansions, then the the sequence of decimal expansions is computable from (every name for) S.

Computability of sequences of reals

Question (Steffen Lempp)

Does every sequence $S \in [0,1]^{\mathbb{N}}$ have a least Turing degree name?

The least Turing degree of any name would naturally measure the complexity of S. Call it the *Turing degree* of S.

(Actually, Lempp asked the question about names of continuous functions $f\colon [0,1] \to \mathbb{R}$. More on that later.)

Easy fact

If $S \in [0,1]^{\mathbb{N}}$ contains no rationals with finite decimal expansions, then the the sequence of decimal expansions is computable from (every name for) S. But this sequence computes a name for S, which is therefore a least Turing degree name.

Computability of sequences of reals

Question (Steffen Lempp)

Does every sequence $S \in [0,1]^{\mathbb{N}}$ have a least Turing degree name?

The least Turing degree of any name would naturally measure the complexity of S. Call it the *Turing degree* of S.

(Actually, Lempp asked the question about names of continuous functions $f\colon [0,1] \to \mathbb{R}$. More on that later.)

Easy fact

If $S \in [0,1]^{\mathbb{N}}$ contains no rationals with finite decimal expansions, then the the sequence of decimal expansions is computable from (every name for) S. But this sequence computes a name for S, which is therefore a least Turing degree name.

Note that we are back to the obstacle Turing identified.

Computable Diagonalizability

Question

Is there a sequence $S \in [0,1]^{\mathbb{N}}$ without Turing degree?

Computable Diagonalizability

Question

Is there a sequence $S \in [0,1]^{\mathbb{N}}$ without Turing degree?

The key to this question will be the following notion:

Definition

A sequence $S \in [0,1]^{\mathbb{N}}$ is *computably diagonalizable* if there is an $r \in [0,1]$ such that $r \notin S$ and r is computable from S.

Computable Diagonalizability

Question

Is there a sequence $S \in [0,1]^{\mathbb{N}}$ without Turing degree?

The key to this question will be the following notion:

Definition

A sequence $S \in [0,1]^{\mathbb{N}}$ is *computably diagonalizable* if there is an $r \in [0,1]$ such that $r \notin S$ and r is computable from S.

By $r \notin S$, we mean that $(\forall n) r \neq S(n)$.

Computable Diagonalizability

Question

Is there a sequence $S \in [0,1]^{\mathbb{N}}$ without Turing degree?

The key to this question will be the following notion:

Definition

A sequence $S \in [0,1]^{\mathbb{N}}$ is *computably diagonalizable* if there is an $r \in [0,1]$ such that $r \notin S$ and r is computable from S.

By $r \notin S$, we mean that $(\forall n) r \neq S(n)$.

By "r is computable from S" we mean that every name of S computes (a name of) r.

Computable Diagonalizability

Question

Is there a sequence $S \in [0,1]^{\mathbb{N}}$ without Turing degree?

The key to this question will be the following notion:

Definition

A sequence $S \in [0,1]^{\mathbb{N}}$ is *computably diagonalizable* if there is an $r \in [0,1]$ such that $r \notin S$ and r is computable from S.

By $r \notin S$, we mean that $(\forall n) r \neq S(n)$.

By "r is computable from S" we mean that every name of S computes (a name of) r.

We will show two things:

- If $S \in [0,1]^{\mathbb{N}}$ has Turing degree, then it is computably diagonalizable.
- There is an $S \in [0,1]^{\mathbb{N}}$ that is not computably diagonalizable.

Computable Diagonalizability

Proposition

If $S \in [0,1]^{\mathbb{N}}$ has Turing degree, then it is computably diagonalizable.

Computable Diagonalizability

Proposition

If $S \in [0,1]^{\mathbb{N}}$ has Turing degree, then it is computably diagonalizable.

Proof.

Every name of S can compute an $r \in [0,1]$, $r \notin S$. (But different names compute different reals.)

Computable Diagonalizability

Proposition

If $S \in [0,1]^{\mathbb{N}}$ has Turing degree, then it is computably diagonalizable.

Proof.

Every name of S can compute an $r \in [0,1]$, $r \notin S$. (But different names compute different reals.) However, if $S \in [0,1]^{\mathbb{N}}$ has Turing degree, then it has a least Turing degree name λ.

Computable Diagonalizability

Proposition

If $S \in [0,1]^{\mathbb{N}}$ has Turing degree, then it is computably diagonalizable.

Proof.

Every name of S can compute an $r \in [0,1]$, $r \notin S$. (But different names compute different reals.) However, if $S \in [0,1]^{\mathbb{N}}$ has Turing degree, then it has a least Turing degree name λ. Take $r \in [0,1]$ such that $r \notin S$ and r is computable from λ. Then every name of S computes r. □

Computable Diagonalizability

Proposition

If $S \in [0,1]^{\mathbb{N}}$ has Turing degree, then it is computably diagonalizable.

Proof.

Every name of S can compute an $r \in [0,1]$, $r \notin S$. (But different names compute different reals.) However, if $S \in [0,1]^{\mathbb{N}}$ has Turing degree, then it has a least Turing degree name λ. Take $r \in [0,1]$ such that $r \notin S$ and r is computable from λ. Then every name of S computes r. □

There is a converse (though we don't need it):

Theorem

If $S \in [0,1]^{\mathbb{N}}$ does not have Turing degree, then S is computably equivalent to a $T \in [0,1]^{\mathbb{N}}$ that is not computably diagonalizable.

Diagonalizing against Computable Diagonalizability

Fact

Diagonalizing against Computable Diagonalizability

Fact

If $S \in [0,1]^{\mathbb{N}}$ computes r, then there is a single Turing functional taking every name of S to a name of r.

Diagonalizing against Computable Diagonalizability

Fact

If $S \in [0,1]^{\mathbb{N}}$ computes r, then there is a single Turing functional taking every name of S to a name of r.

Let $\{\psi_e\}_{e\in\mathbb{N}}$ be an effective listing of all partial computable functions from (names of) elements of $[0,1]^{\mathbb{N}}$ to (names of) reals in $[0,1]$.

Diagonalizing against Computable Diagonalizability

Fact

If $S \in [0,1]^{\mathbb{N}}$ computes r, then there is a single Turing functional taking every name of S to a name of r.

Let $\{\psi_e\}_{e\in\mathbb{N}}$ be an effective listing of all partial computable functions from (names of) elements of $[0,1]^{\mathbb{N}}$ to (names of) reals in $[0,1]$.

We extend each ψ_e to a total *mutivalued* function $\widehat{\psi}_e$ such that

- If $\psi_e(\lambda) = r$, then $\widehat{\psi}_e(\lambda) = \{r\}$.
- Otherwise, $\widehat{\psi}_e(\lambda)$ is a closed interval.

Diagonalizing against Computable Diagonalizability

Fact

If $S \in [0,1]^{\mathbb{N}}$ computes r, then there is a single Turing functional taking every name of S to a name of r.

Let $\{\psi_e\}_{e\in\mathbb{N}}$ be an effective listing of all partial computable functions from (names of) elements of $[0,1]^{\mathbb{N}}$ to (names of) reals in $[0,1]$.

We extend each ψ_e to a total *mutivalued* function $\widehat{\psi}_e$ such that

- If $\psi_e(\lambda) = r$, then $\widehat{\psi}_e(\lambda) = \{r\}$.
- Otherwise, $\widehat{\psi}_e(\lambda)$ is a closed interval.

We may assume that $\widehat{\psi}_e$ is coherent, in that its behavior on different names of the same sequence will be the same.

Diagonalizing against Computable Diagonalizability

Fact

If $S \in [0,1]^{\mathbb{N}}$ computes r, then there is a single Turing functional taking every name of S to a name of r.

Let $\{\psi_e\}_{e\in\mathbb{N}}$ be an effective listing of all partial computable functions from (names of) elements of $[0,1]^{\mathbb{N}}$ to (names of) reals in $[0,1]$.

We extend each ψ_e to a total *mutivalued* function $\widehat{\psi}_e$ such that

- If $\psi_e(\lambda) = r$, then $\widehat{\psi}_e(\lambda) = \{r\}$.
- Otherwise, $\widehat{\psi}_e(\lambda)$ is a closed interval.

We may assume that $\widehat{\psi}_e$ is coherent, in that its behavior on different names of the same sequence will be the same. So they can be viewed as functions on $[0,1]^{\mathbb{N}}$.

Diagonalizing against Computable Diagonalizability

Fact

If $S \in [0,1]^{\mathbb{N}}$ computes r, then there is a single Turing functional taking every name of S to a name of r.

Let $\{\psi_e\}_{e\in\mathbb{N}}$ be an effective listing of all partial computable functions from (names of) elements of $[0,1]^{\mathbb{N}}$ to (names of) reals in $[0,1]$.

We extend each ψ_e to a total *mutivalued* function $\widehat{\psi}_e$ such that

- If $\psi_e(\lambda) = r$, then $\widehat{\psi}_e(\lambda) = \{r\}$.
- Otherwise, $\widehat{\psi}_e(\lambda)$ is a closed interval.

We may assume that $\widehat{\psi}_e$ is coherent, in that its behavior on different names of the same sequence will be the same. So they can be viewed as functions on $[0,1]^{\mathbb{N}}$.

Define $\Psi\colon [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ by $\Psi(S) = \bigoplus_{e\in\mathbb{N}} \widehat{\psi}_e(S)$.

Diagonalizing against Computable Diagonalizability

Define $\Psi\colon [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ by $\Psi(S) = \bigoplus_{e\in\mathbb{N}} \widehat{\psi}_e(S)$.

Diagonalizing against Computable Diagonalizability

Define $\Psi\colon [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ by $\Psi(S) = \bigoplus_{e\in\mathbb{N}} \widehat{\psi}_e(S)$.

We need a generalization of Brouwer's fixed point theorem to multivalued functions on an infinite dimensional space.

Theorem (Eilenberg and Montgomery 1946)

Assume that $\Psi\colon [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ is a multivalued function with a closed graph such that $\Psi(S)$ is nonempty and convex for each $S \in [0,1]^{\mathbb{N}}$. Then there is a fixed point S of Ψ (i.e., $S \in \Psi(S)$).

Diagonalizing against Computable Diagonalizability

Define $\Psi\colon [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ by $\Psi(S) = \bigoplus_{e\in\mathbb{N}} \widehat{\psi}_e(S)$.

We need a generalization of Brouwer's fixed point theorem to multivalued functions on an infinite dimensional space.

Theorem (Eilenberg and Montgomery 1946)

Assume that $\Psi\colon [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ is a multivalued function with a closed graph such that $\Psi(S)$ is nonempty and convex for each $S \in [0,1]^{\mathbb{N}}$. Then there is a fixed point S of Ψ (i.e., $S \in \Psi(S)$).

Our Ψ satisfies the hypothesis.

Diagonalizing against Computable Diagonalizability

Define $\Psi\colon [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ by $\Psi(S) = \bigoplus_{e\in\mathbb{N}} \widehat{\psi}_e(S)$.

We need a generalization of Brouwer's fixed point theorem to multivalued functions on an infinite dimensional space.

Theorem (Eilenberg and Montgomery 1946)

Assume that $\Psi\colon [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ is a multivalued function with a closed graph such that $\Psi(S)$ is nonempty and convex for each $S \in [0,1]^{\mathbb{N}}$. Then there is a fixed point S of Ψ (i.e., $S \in \Psi(S)$).

Our Ψ satisfies the hypothesis.

Therefore, there is an $S \in [0,1]^{\mathbb{N}}$ such that $S \in \Psi(S)$.

Diagonalizing against Computable Diagonalizability

Define $\Psi\colon [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ by $\Psi(S) = \bigoplus_{e\in\mathbb{N}} \widehat{\psi}_e(S)$.

We need a generalization of Brouwer's fixed point theorem to multivalued functions on an infinite dimensional space.

Theorem (Eilenberg and Montgomery 1946)

Assume that $\Psi\colon [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ is a multivalued function with a closed graph such that $\Psi(S)$ is nonempty and convex for each $S \in [0,1]^{\mathbb{N}}$. Then there is a fixed point S of Ψ (i.e., $S \in \Psi(S)$).

Our Ψ satisfies the hypothesis.

Therefore, there is an $S \in [0,1]^{\mathbb{N}}$ such that $S \in \Psi(S)$. This means that $S(e) \in \widehat{\psi}_e(S)$, for all e.

Diagonalizing against Computable Diagonalizability

Define $\Psi\colon [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ by $\Psi(S) = \bigoplus_{e\in\mathbb{N}} \widehat{\psi}_e(S)$.

We need a generalization of Brouwer's fixed point theorem to multivalued functions on an infinite dimensional space.

Theorem (Eilenberg and Montgomery 1946)

Assume that $\Psi\colon [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ is a multivalued function with a closed graph such that $\Psi(S)$ is nonempty and convex for each $S \in [0,1]^{\mathbb{N}}$. Then there is a fixed point S of Ψ (i.e., $S \in \Psi(S)$).

Our Ψ satisfies the hypothesis.

Therefore, there is an $S \in [0,1]^{\mathbb{N}}$ such that $S \in \Psi(S)$. This means that $S(e) \in \widehat{\psi}_e(S)$, for all e. So if $\widehat{\psi}_e(S) = \{r\}$, then $S(e) = r$.

Diagonalizing against Computable Diagonalizability

Define $\Psi\colon [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ by $\Psi(S) = \bigoplus_{e\in\mathbb{N}} \widehat{\psi}_e(S)$.

We need a generalization of Brouwer's fixed point theorem to multivalued functions on an infinite dimensional space.

Theorem (Eilenberg and Montgomery 1946)

Assume that $\Psi\colon [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ is a multivalued function with a closed graph such that $\Psi(S)$ is nonempty and convex for each $S \in [0,1]^{\mathbb{N}}$. Then there is a fixed point S of Ψ (i.e., $S \in \Psi(S)$).

Our Ψ satisfies the hypothesis.

Therefore, there is an $S \in [0,1]^{\mathbb{N}}$ such that $S \in \Psi(S)$. This means that $S(e) \in \widehat{\psi}_e(S)$, for all e. So if $\widehat{\psi}_e(S) = \{r\}$, then $S(e) = r$.

Therefore,

S is (diagonally) non computably diagonalizable.

Computability of sequences of reals

We have shown that there is an $S \in [0,1]^{\mathbb{N}}$ that is not computably diagonalizable.

Computability of sequences of reals

We have shown that there is an $S \in [0,1]^{\mathbb{N}}$ that is not computably diagonalizable.

It follows that:

- S does not have Turing degree.
- S has no least Turing degree name.

Computability of sequences of reals

We have shown that there is an $S \in [0,1]^{\mathbb{N}}$ that is not computably diagonalizable.

It follows that:

- S does not have Turing degree.
- S has no least Turing degree name.

It can also be shown:

Theorem

- Every non-computable sequence $S \in [0,1]^{\mathbb{N}}$ computes a non-computable set (contains a non-computable real).
- Every computable function $f\colon [0,1] \to \mathbb{R}$ is computably equivalent to an element of $[0,1]^{\mathbb{N}}$, and visa versa.

Computability of sequences of reals

We have shown that there is an $S \in [0,1]^{\mathbb{N}}$ that is not computably diagonalizable.

It follows that:

- S does not have Turing degree.
- S has no least Turing degree name.

It can also be shown:

Theorem

- Every non-computable sequence $S \in [0,1]^{\mathbb{N}}$ computes a non-computable set (contains a non-computable real).
- Every computable function $f\colon [0,1] \to \mathbb{R}$ is computably equivalent to an element of $[0,1]^{\mathbb{N}}$, and visa versa.

The degrees of elements of $[0,1]^{\mathbb{N}}$ are called the *continuous degrees*.

A moment to reflect

A moment to reflect

The Turing degrees are not sufficient to measure the effective content of $[0,1]^{\mathbb{N}}$ (or continuous functions $f\colon [0,1] \to \mathbb{R}$).

A moment to reflect

The Turing degrees are not sufficient to measure the effective content of $[0,1]^{\mathbb{N}}$ (or continuous functions $f\colon [0,1] \to \mathbb{R}$).

But the elements of $[0,1]^{\mathbb{N}}$ that don't have Turing degree seem rare.

A moment to reflect

The Turing degrees are not sufficient to measure the effective content of $[0,1]^{\mathbb{N}}$ (or continuous functions $f\colon [0,1] \to \mathbb{R}$).

But the elements of $[0,1]^{\mathbb{N}}$ that don't have Turing degree seem rare. Any random or generic element has Turing degree.

A moment to reflect

The Turing degrees are not sufficient to measure the effective content of $[0,1]^{\mathbb{N}}$ (or continuous functions $f\colon [0,1] \to \mathbb{R}$).

But the elements of $[0,1]^{\mathbb{N}}$ that don't have Turing degree seem rare. Any random or generic element has Turing degree.

Even so:

Leonid Levin (essentially) constructed such an object in the seventies.

Martin-Löf randomness

In 1966, Martin-Löf gave a definition of randomness.

Martin-Löf randomness

In 1966, Martin-Löf gave a definition of randomness.

If $\sigma \in 2^{<\mathbb{N}}$, let $[\sigma] = \{X \in 2^{\mathbb{N}} : \sigma \text{ is a prefix of } X\}$.

Martin-Löf randomness

In 1966, Martin-Löf gave a definition of randomness.

If $\sigma \in 2^{<\mathbb{N}}$, let $[\sigma] = \{X \in 2^{\mathbb{N}} : \sigma \text{ is a prefix of } X\}$.
If $S \subseteq 2^{<\mathbb{N}}$, let $[S] = \bigcup_{\sigma \in S} [\sigma]$.

Martin-Löf randomness

In 1966, Martin-Löf gave a definition of randomness.

If $\sigma \in 2^{<\mathbb{N}}$, let $[\sigma] = \{X \in 2^{\mathbb{N}} : \sigma \text{ is a prefix of } X\}$.
If $S \subseteq 2^{<\mathbb{N}}$, let $[S] = \bigcup_{\sigma \in S} [\sigma]$.

If S is computably enumerable, then $[S]$ is a Σ^0_1 *class*. These are the "effective open sets".

Martin-Löf randomness

In 1966, Martin-Löf gave a definition of randomness.

If $\sigma \in 2^{<\mathbb{N}}$, let $[\sigma] = \{X \in 2^{\mathbb{N}} : \sigma \text{ is a prefix of } X\}$.
If $S \subseteq 2^{<\mathbb{N}}$, let $[S] = \bigcup_{\sigma \in S} [\sigma]$.

If S is computably enumerable, then $[S]$ is a Σ^0_1 *class*. These are the "effective open sets".

Definition

A *Martin-Löf test* is a computable sequence $\{V_n\}_{n \in \mathbb{N}}$ of Σ^0_1 classes such that $\mu(V_n) \leqslant 2^{-n}$.

Martin-Löf randomness

In 1966, Martin-Löf gave a definition of randomness.

If $\sigma \in 2^{<\mathbb{N}}$, let $[\sigma] = \{X \in 2^{\mathbb{N}} : \sigma \text{ is a prefix of } X\}$.
If $S \subseteq 2^{<\mathbb{N}}$, let $[S] = \bigcup_{\sigma \in S} [\sigma]$.

If S is computably enumerable, then $[S]$ is a Σ^0_1 *class*. These are the "effective open sets".

Definition

A *Martin-Löf test* is a computable sequence $\{V_n\}_{n \in \mathbb{N}}$ of Σ^0_1 classes such that $\mu(V_n) \leqslant 2^{-n}$.

$X \in 2^{\mathbb{N}}$ is *Martin-Löf random* if no Martin-Löf test covers it (i.e., $X \notin \bigcap_{n \in \mathbb{N}} V_n$).

Martin-Löf randomness

In 1966, Martin-Löf gave a definition of randomness.

If $\sigma \in 2^{<\mathbb{N}}$, let $[\sigma] = \{X \in 2^{\mathbb{N}} : \sigma \text{ is a prefix of } X\}$.
If $S \subseteq 2^{<\mathbb{N}}$, let $[S] = \bigcup_{\sigma \in S} [\sigma]$.

If S is computably enumerable, then $[S]$ is a Σ^0_1 *class*. These are the "effective open sets".

Definition

A *Martin-Löf test* is a computable sequence $\{V_n\}_{n \in \mathbb{N}}$ of Σ^0_1 classes such that $\mu(V_n) \leqslant 2^{-n}$.

$X \in 2^{\mathbb{N}}$ is *Martin-Löf random* if no Martin-Löf test covers it (i.e., $X \notin \bigcap_{n \in \mathbb{N}} V_n$).

There is an extensive literature on effective randomness; Martin-Löf randomness is one of the central notions.

Randomness for other measures

We have a notion of effective randomness for Lebesgue measure.

Randomness for other measures

We have a notion of effective randomness for Lebesgue measure.
What about other (possibly non-computable) measures?

Randomness for other measures

We have a notion of effective randomness for Lebesgue measure. What about other (possibly non-computable) measures?

Let ν be an arbitrary measure on $2^{\mathbb{N}}$.

Randomness for other measures

We have a notion of effective randomness for Lebesgue measure. What about other (possibly non-computable) measures?

Let ν be an arbitrary measure on $2^{\mathbb{N}}$. We identify ν with its values $\nu([\sigma])$, where $\sigma \in 2^{<\mathbb{N}}$.

Randomness for other measures

We have a notion of effective randomness for Lebesgue measure. What about other (possibly non-computable) measures?

Let ν be an arbitrary measure on $2^{\mathbb{N}}$. We identify ν with its values $\nu([\sigma])$, where $\sigma \in 2^{<\mathbb{N}}$.

In other words, we think of ν as a sequence in $[0,1]^{2^{<\mathbb{N}}}$.

Randomness for other measures

We have a notion of effective randomness for Lebesgue measure. What about other (possibly non-computable) measures?

Let ν be an arbitrary measure on $2^{\mathbb{N}}$. We identify ν with its values $\nu([\sigma])$, where $\sigma \in 2^{<\mathbb{N}}$.

In other words, we think of ν as a sequence in $[0,1]^{2^{<\mathbb{N}}}$.

Definition

Let λ be a name of ν.

Randomness for other measures

We have a notion of effective randomness for Lebesgue measure. What about other (possibly non-computable) measures?

Let ν be an arbitrary measure on $2^{\mathbb{N}}$. We identify ν with its values $\nu([\sigma])$, where $\sigma \in 2^{<\mathbb{N}}$.

In other words, we think of ν as a sequence in $[0,1]^{2^{<\mathbb{N}}}$.

Definition

Let λ be a name of ν. A ν-*Martin-Löf test relative to* λ is a λ-computable sequence $\{V_n\}_{n\in\mathbb{N}}$ of $\Sigma^0_1[\lambda]$ classes such that $\nu(V_n) \leqslant 2^{-n}$.

Randomness for other measures

We have a notion of effective randomness for Lebesgue measure. What about other (possibly non-computable) measures?

Let ν be an arbitrary measure on $2^{\mathbb{N}}$. We identify ν with its values $\nu([\sigma])$, where $\sigma \in 2^{<\mathbb{N}}$.

In other words, we think of ν as a sequence in $[0,1]^{2^{<\mathbb{N}}}$.

Definition

Let λ be a name of ν. A ν-*Martin-Löf test relative to* λ is a λ-computable sequence $\{V_n\}_{n\in\mathbb{N}}$ of $\Sigma^0_1[\lambda]$ classes such that $\nu(V_n) \leqslant 2^{-n}$.

$X \in 2^{\mathbb{N}}$ is ν-*random* if there is some name λ of ν such that no ν-Martin-Löf test relative to λ covers X.

Randomness for other measures

We have a notion of effective randomness for Lebesgue measure. What about other (possibly non-computable) measures?

Let ν be an arbitrary measure on $2^{\mathbb{N}}$. We identify ν with its values $\nu([\sigma])$, where $\sigma \in 2^{<\mathbb{N}}$.

In other words, we think of ν as a sequence in $[0,1]^{2^{<\mathbb{N}}}$.

Definition

Let λ be a name of ν. A ν-*Martin-Löf test relative to* λ is a λ-computable sequence $\{V_n\}_{n\in\mathbb{N}}$ of $\Sigma^0_1[\lambda]$ classes such that $\nu(V_n) \leqslant 2^{-n}$.

$X \in 2^{\mathbb{N}}$ is ν-*random* if there is some name λ of ν such that no ν-Martin-Löf test relative to λ covers X.

This definition is equivalent to ones of Levin 1976 and Reimann 2008.

Weakly neutral measures

Definition

ν is a *weakly neutral measure* if every $X \in 2^{\mathbb{N}}$ is ν-random.

Weakly neutral measures

Definition

ν is a *weakly neutral measure* if every $X \in 2^{\mathbb{N}}$ is ν-random.

In 1976, Levin introduce neutral measures (a slightly stronger notion) and proved that they exist.

Weakly neutral measures

Definition

ν is a *weakly neutral measure* if every $X \in 2^{\mathbb{N}}$ is ν-random.

In 1976, Levin introduce neutral measures (a slightly stronger notion) and proved that they exist.

Proposition (Day and M.)

If ν has Turing degree, then it is not weakly neutral.

Weakly neutral measures

Definition

ν is a *weakly neutral measure* if every $X \in 2^{\mathbb{N}}$ is ν-random.

In 1976, Levin introduce neutral measures (a slightly stronger notion) and proved that they exist.

Proposition (Day and M.)

If ν has Turing degree, then it is not weakly neutral.

Proof.

Every name of ν can compute a $X \in 2^{\mathbb{N}}$ that is not an atom of ν.

Weakly neutral measures

Definition

ν is a *weakly neutral measure* if every $X \in 2^{\mathbb{N}}$ is ν-random.

In 1976, Levin introduce neutral measures (a slightly stronger notion) and proved that they exist.

Proposition (Day and M.)

If ν has Turing degree, then it is not weakly neutral.

Proof.

Every name of ν can compute a $X \in 2^{\mathbb{N}}$ that is not an atom of ν. If ν has Turing degree, then it has a least Turing degree name λ.

Weakly neutral measures

Definition

ν is a *weakly neutral measure* if every $X \in 2^{\mathbb{N}}$ is ν-random.

In 1976, Levin introduce neutral measures (a slightly stronger notion) and proved that they exist.

Proposition (Day and M.)

If ν has Turing degree, then it is not weakly neutral.

Proof.

Every name of ν can compute a $X \in 2^{\mathbb{N}}$ that is not an atom of ν. If ν has Turing degree, then it has a least Turing degree name λ. Take $X \in 2^{\mathbb{N}}$ computable form λ and not an atom of ν. Then every name of ν computes a ν-Martin-Löf test covering X.

Weakly neutral measures

Definition

ν is a *weakly neutral measure* if every $X \in 2^{\mathbb{N}}$ is ν-random.

In 1976, Levin introduce neutral measures (a slightly stronger notion) and proved that they exist.

Proposition (Day and M.)

If ν has Turing degree, then it is not weakly neutral.

Proof.

Every name of ν can compute a $X \in 2^{\mathbb{N}}$ that is not an atom of ν. If ν has Turing degree, then it has a least Turing degree name λ. Take $X \in 2^{\mathbb{N}}$ computable form λ and not an atom of ν. Then every name of ν computes a ν-Martin-Löf test covering X. So X is not ν-random. □

Weakly neutral measures

Levin's neutral measures have non-Turing continuous degree.

Weakly neutral measures

Levin's neutral measures have non-Turing continuous degree.

Levin's construction used Sperner's Lemma, a combinatorial analogue of the Brouwer fixed point theorem.

Weakly neutral measures

Levin's neutral measures have non-Turing continuous degree.

Levin's construction used Sperner's Lemma, a combinatorial analogue of the Brouwer fixed point theorem. (Maybe the non-Turing continuous degrees really are a topological phenomenon.)

Weakly neutral measures

Levin's neutral measures have non-Turing continuous degree.

Levin's construction used Sperner's Lemma, a combinatorial analogue of the Brouwer fixed point theorem. (Maybe the non-Turing continuous degrees really are a topological phenomenon.)

We now have a better understanding of (weakly) neutral measures.

Weakly neutral measures

Levin's neutral measures have non-Turing continuous degree.

Levin's construction used Sperner's Lemma, a combinatorial analogue of the Brouwer fixed point theorem. (Maybe the non-Turing continuous degrees really are a topological phenomenon.)

We now have a better understanding of (weakly) neutral measures.

Definition

$X \in 2^{\mathbb{N}}$ has *PA degree* if it computes an infinite path through every infinite computable tree $T \subseteq 2^{<\mathbb{N}}$.

Weakly neutral measures

Levin's neutral measures have non-Turing continuous degree.

Levin's construction used Sperner's Lemma, a combinatorial analogue of the Brouwer fixed point theorem. (Maybe the non-Turing continuous degrees really are a topological phenomenon.)

We now have a better understanding of (weakly) neutral measures.

Definition

$X \in 2^{\mathbb{N}}$ has *PA degree* if it computes an infinite path through every infinite computable tree $T \subseteq 2^{<\mathbb{N}}$.

The PA degrees are so named because they are the Turing degrees of complete consistent extensions of Peano arithmetic.

Weakly neutral measures

Levin's neutral measures have non-Turing continuous degree.

Levin's construction used Sperner's Lemma, a combinatorial analogue of the Brouwer fixed point theorem. (Maybe the non-Turing continuous degrees really are a topological phenomenon.)

We now have a better understanding of (weakly) neutral measures.

Definition

$X \in 2^{\mathbb{N}}$ has *PA degree* if it computes an infinite path through every infinite computable tree $T \subseteq 2^{<\mathbb{N}}$.

The PA degrees are so named because they are the Turing degrees of complete consistent extensions of Peano arithmetic.

They are important in computability theory and reverse mathematics.

Weakly neutral measures

Theorem (Day and M.)

- Every PA degree computes a neutral measure.
- Every weakly neutral measure computes a PA degree.
- The atoms of a weakly neutral measure are exactly the $X \in 2^{\mathbb{N}}$ computable from the measure.
- Every weakly neutral measure has a Martin-Löf random atom.

Weakly neutral measures

Theorem (Day and M.)

- Every PA degree computes a neutral measure.
- Every weakly neutral measure computes a PA degree.
- The atoms of a weakly neutral measure are exactly the $X \in 2^{\mathbb{N}}$ computable from the measure.
- Every weakly neutral measure has a Martin-Löf random atom.

We end with a result about two well-studied classes of Turing degrees, first proved using neutral measures:

Theorem (Day and Reimann)

Let $A \subseteq \mathbb{N}$ be a computably enumerable set and let $X \not\geq_T A$ have PA degree. Then $A \oplus X$ computes the halting problem.

I have time for this slide? Yay!

In fact:

Theorem (Day and Reimann)

Assume that $A \subseteq \mathbb{N}$ is computably enumerable and $X \subseteq \mathbb{N}$ has PA degree. Then either $A \oplus X$ computes the halting problem or X has PA degree relative to A.

I have time for this slide? Yay!

In fact:

Theorem (Day and Reimann)

Assume that $A \subseteq \mathbb{N}$ is computably enumerable and $X \subseteq \mathbb{N}$ has PA degree. Then either $A \oplus X$ computes the halting problem or X has PA degree relative to A.

Theorem (Frank Steffan)

Assume that $A \subseteq \mathbb{N}$ is computably enumerable and $X \subseteq \mathbb{N}$ is Martin-Löf random. Then either $A \oplus X$ computes the halting problem or X is Martin-Löf random relative to A.

I have time for this slide? Yay!

In fact:

Theorem (Day and Reimann)

Assume that $A \subseteq \mathbb{N}$ is computably enumerable and $X \subseteq \mathbb{N}$ has PA degree. Then either $A \oplus X$ computes the halting problem or X has PA degree relative to A.

Theorem (Frank Steffan)

Assume that $A \subseteq \mathbb{N}$ is computably enumerable and $X \subseteq \mathbb{N}$ is Martin-Löf random. Then either $A \oplus X$ computes the halting problem or X is Martin-Löf random relative to A.

Theorem

Assume that $A \subseteq \mathbb{N}$ is computably enumerable and $X \subseteq \mathbb{N}$ has DNC (diagonally non-computable) degree. Then either $A \oplus X$ computes the halting problem or X has DNC degree relative to A.

Thank You!