Extracting information is hard

Computability and effective dimension

Joseph S. Miller

2008 Logic Colloquium
Bern, Switzerland
July 6, 2008

Extracting ~~information~~ is hard

Computability and effective dimension

Joseph S. Miller

2008 Logic Colloquium
Bern, Switzerland
July 6, 2008

Extracting randomness is hard

Computability and effective dimension

Joseph S. Miller

2008 Logic Colloquium
Bern, Switzerland
July 6, 2008

Informal Question (Reimann, Terwijn)

Given a semi-random sequence, can we effectively produce a random sequence?

Informal Question (Reimann, Terwijn)

Given a semi-random sequence, can we effectively produce a random sequence? Or at least a sequence that is closer to random (i.e., one with higher information density)?

Informal Question (Reimann, Terwijn)

Given a semi-random sequence, can we effectively produce a random sequence? Or at least a sequence that is closer to random (i.e., one with higher information density)?

- We will use *information* and *randomness* somewhat interchangeably (more on this later).

Informal Question (Reimann, Terwijn)

Given a semi-random sequence, can we effectively produce a random sequence? Or at least a sequence that is closer to random (i.e., one with higher information density)?

- We will use *information* and *randomness* somewhat interchangeably (more on this later).
- By sequence we mean infinite binary sequence.

A few examples

Example 0: Produce a sequence by flipping a fair coin: heads is 1 and tails is 0.

A few examples

Example 0: Produce a sequence by flipping a fair coin: heads is 1 and tails is 0.

- Random (with probability 1)

A few examples

Example 0: Produce a sequence by flipping a fair coin: heads is 1 and tails is 0.

- Random (with probability 1)
- I.e., you can't compress the initial segments (by much) and can't win (a lot of) money betting on the sequence.

A few examples

Example 0: Produce a sequence by flipping a fair coin: heads is 1 and tails is 0.

- Random (with probability 1)
- I.e., you can't compress the initial segments (by much) and can't win (a lot of) money betting on the sequence.

Example 1: Use a coin to determine the odd bits and make every even bit 0.

A few examples

Example 0: Produce a sequence by flipping a fair coin: heads is 1 and tails is 0.

- Random (with probability 1)
- I.e., you can't compress the initial segments (by much) and can't win (a lot of) money betting on the sequence.

Example 1: Use a coin to determine the odd bits and make every even bit 0.

- Half-random (every two bits contains one bit of information)

A few examples

Example 0: Produce a sequence by flipping a fair coin: heads is 1 and tails is 0.

- Random (with probability 1)
- I.e., you can't compress the initial segments (by much) and can't win (a lot of) money betting on the sequence.

Example 1: Use a coin to determine the odd bits and make every even bit 0.

- Half-random (every two bits contains one bit of information)
- Easy to extract a random sequence

A few examples

Example 2: Produce a sequence by flipping a biased coin.

A few examples

Example 2: Produce a sequence by flipping a biased coin.

- The right bias will make it half-random.

A few examples

Example 2: Produce a sequence by flipping a biased coin.

- The right bias will make it half-random. (About 89% heads will make the Shannon entropy of each bit $1/2$.)

A few examples

Example 2: Produce a sequence by flipping a biased coin.

- The right bias will make it half-random. (About 89% heads will make the Shannon entropy of each bit $1/2$.)
- Again, we can extract a random sequence

A few examples

Example 2: Produce a sequence by flipping a biased coin.

- The right bias will make it half-random. (About 89% heads will make the Shannon entropy of each bit $1/2$.)
- Again, we can extract a random sequence

Extracting randomness from a biased coin (von Neumann, 1951)

- Consider pairs of coin flips:

A few examples

Example 2: Produce a sequence by flipping a biased coin.

- The right bias will make it half-random. (About 89% heads will make the Shannon entropy of each bit $1/2$.)
- Again, we can extract a random sequence

Extracting randomness from a biased coin (von Neumann, 1951)

- Consider pairs of coin flips:
- Output 1 if you see HT and 0 if you see TH

A few examples

Example 2: Produce a sequence by flipping a biased coin.

- The right bias will make it half-random. (About 89% heads will make the Shannon entropy of each bit $1/2$.)
- Again, we can extract a random sequence

Extracting randomness from a biased coin (von Neumann, 1951)

- Consider pairs of coin flips:
- Output 1 if you see HT and 0 if you see TH
- Produce no output for HH or TT

A few examples

Example 2: Produce a sequence by flipping a biased coin.

- The right bias will make it half-random. (About 89% heads will make the Shannon entropy of each bit $1/2$.)
- Again, we can extract a random sequence

Extracting randomness from a biased coin (von Neumann, 1951)

- Consider pairs of coin flips:
- Output 1 if you see HT and 0 if you see TH
- Produce no output for HH or TT
- The resulting sequence is random

Algorithmic randomness

To move beyond simple examples, we need to make the informal question precise.

Algorithmic randomness

To move beyond simple examples, we need to make the informal question precise.

We use algorithmic randomness

Algorithmic randomness

To move beyond simple examples, we need to make the informal question precise.

We use algorithmic randomness

Algorithmic randomness gives answers to questions like:

- Given a finite binary sequence (a *string*) $\sigma \in 2^{<\omega}$, how much *information* does it contain? I.e., how *complex* is it?

Algorithmic randomness

To move beyond simple examples, we need to make the informal question precise.

We use algorithmic randomness

Algorithmic randomness gives answers to questions like:

- Given a finite binary sequence (a *string*) $\sigma \in 2^{<\omega}$, how much *information* does it contain? I.e., how *complex* is it?
- What does it mean for a sequence $A \in 2^{\omega}$ to be *random*?

To move beyond simple examples, we need to make the informal question precise.

We use algorithmic randomness

Algorithmic randomness gives answers to questions like:

- Given a finite binary sequence (a *string*) $\sigma \in 2^{<\omega}$, how much *information* does it contain? I.e., how *complex* is it?
- What does it mean for a sequence $A \in 2^{\omega}$ to be *random*?
- What does it mean for a sequence to be *half-random* (i.e., how do we measure information density)?

The complexity of strings

Solomonoff (1964) and Kolmogorov (1965) independently defined *Kolmogorov complexity* to measure the information content of finite binary strings.

The complexity of strings

Solomonoff (1964) and Kolmogorov (1965) independently defined *Kolmogorov complexity* to measure the information content of finite binary strings.

Levin (1973) and Chaitin (1975) modified Kolmogorov complexity to characterize random sequences. (We will use this modified version.)

The complexity of strings

Solomonoff (1964) and Kolmogorov (1965) independently defined *Kolmogorov complexity* to measure the information content of finite binary strings.

Levin (1973) and Chaitin (1975) modified Kolmogorov complexity to characterize random sequences. (We will use this modified version.)

Idea

The complexity of a string is the length of its shortest binary *description*.

The complexity of strings

Solomonoff (1964) and Kolmogorov (1965) independently defined *Kolmogorov complexity* to measure the information content of finite binary strings.

Levin (1973) and Chaitin (1975) modified Kolmogorov complexity to characterize random sequences. (We will use this modified version.)

Idea

The complexity of a string is the length of its shortest binary *description*.

What do we mean by description?

Examples of strings

In a reasonable "system of descriptions":

Examples of strings

In a reasonable "system of descriptions":

(a) $\overbrace{\mathsf{101010}\cdots\mathsf{10}}^{10^{100}\text{ bits}}$ would have a short description.

Examples of strings

In a reasonable "system of descriptions":

(a) $\overbrace{\mathsf{101010}\cdots\mathsf{10}}^{\mathsf{10}^{\mathsf{100}}\text{ bits}}$ would have a short description.

(b) The first $\mathsf{10}^{\mathsf{100}}$ digits in the binary expansion of π would have a fairly short description (much shorter than $\mathsf{10}^{\mathsf{100}}$ bits).

Examples of strings

In a reasonable "system of descriptions":

(a) $\overbrace{101010\cdots10}^{10^{100}\text{ bits}}$ would have a short description.

(b) The first 10^{100} digits in the binary expansion of π would have a fairly short description (much shorter than 10^{100} bits).

In *any* "system of descriptions":

(c) The result of flipping a coin 10^{100} times (heads=1, tails=0) will almost certainly not have a description much shorter than 10^{100}.

Examples of strings

In a reasonable "system of descriptions":

(a) $\overbrace{101010\cdots10}^{10^{100}\text{ bits}}$ would have a short description.

(b) The first 10^{100} digits in the binary expansion of π would have a fairly short description (much shorter than 10^{100} bits).

In *any* "system of descriptions":

(c) The result of flipping a coin 10^{100} times (heads=1, tails=0) will almost certainly not have a description much shorter than 10^{100}.

There are not enough short descriptions to go around.

What is a description?

Question. What sort of descriptions should we use?

What is a description?

Question. What sort of descriptions should we use?

Answer.

1. Want to be able to figure out what a description describes.

What is a description?

Question. What sort of descriptions should we use?

Answer.

1. Want to be able to figure out what a description describes.
2. No proper prefix of a description should be a description.

What is a description?

Question. What sort of descriptions should we use?

Answer.

1. Want to be able to figure out what a description describes.
2. No proper prefix of a description should be a description.

In other words, we want our descriptions to be “decoded” by a partial computable function with *prefix-free* domain:

$$M\colon 2^{<\omega} \to 2^{<\omega}$$

What is a description?

Question. What sort of descriptions should we use?

Answer.

1. Want to be able to figure out what a description describes.
2. No proper prefix of a description should be a description.

In other words, we want our descriptions to be "decoded" by a partial computable function with *prefix-free* domain:

$$M\colon 2^{<\omega} \to 2^{<\omega}$$

Think of M *as a decompression algorithm.*

Why prefix-free?

If a description could be a proper prefix of another description, then we need to be told when a description ends.

Why prefix-free?

If a description could be a proper prefix of another description, then we need to be told when a description ends.

This is extra information

Why prefix-free?

If a description could be a proper prefix of another description, then we need to be told when a description ends.

This is extra information

So, a description of length n could contain more than n bits of information.

Why prefix-free?

If a description could be a proper prefix of another description, then we need to be told when a description ends.

This is extra information

So, a description of length n could contain more than n bits of information. Description length can underestimate complexity.

Why prefix-free?

If a description could be a proper prefix of another description, then we need to be told when a description ends.

This is extra information

So, a description of length n could contain more than n bits of information. Description length can underestimate complexity.

Intuition

In a prefix-free system, descriptions code their own lengths.

Why prefix-free?

If a description could be a proper prefix of another description, then we need to be told when a description ends.

This is extra information

So, a description of length n could contain more than n bits of information. Description length can underestimate complexity.

Intuition

In a prefix-free system, descriptions code their own lengths.

Hence, the length of a description is not extra information.

Decompression functions

Let $M\colon 2^{<\omega} \to 2^{<\omega}$ be a partial computable function with prefix-free domain (a *machine*).

Decompression functions

Let $M\colon 2^{<\omega} \to 2^{<\omega}$ be a partial computable function with prefix-free domain (a *machine*).

Definition (Kolmogorov complexity with respect to M)

$$K_M(\sigma) = \min\{|\tau| \colon M(\tau) = \sigma\}.$$

Decompression functions

Let $M\colon 2^{<\omega} \to 2^{<\omega}$ be a partial computable function with prefix-free domain (a *machine*).

Definition (Kolmogorov complexity with respect to M)

$$K_M(\sigma) = \min\{|\tau| \colon M(\tau) = \sigma\}.$$

Question. But how should we choose M?

Decompression functions

Let $M\colon 2^{<\omega} \to 2^{<\omega}$ be a partial computable function with prefix-free domain (a *machine*).

Definition (Kolmogorov complexity with respect to M)

$$K_M(\sigma) = \min\{|\tau| : M(\tau) = \sigma\}.$$

Question. But how should we choose M?

Answer. There is an (essentially) optimal choice.

Prefix-free (Kolmogorov) complexity

There is a partial computable prefix-free $U\colon 2^{<\omega} \to 2^{<\omega}$ such that if $M\colon 2^{<\omega} \to 2^{<\omega}$ is any other partial computable prefix-free function, then

$$K_U(\sigma) \leqslant K_M(\sigma) + O(1).$$

Prefix-free (Kolmogorov) complexity

There is a partial computable prefix-free $U\colon 2^{<\omega} \to 2^{<\omega}$ such that if $M\colon 2^{<\omega} \to 2^{<\omega}$ is any other partial computable prefix-free function, then

$$K_U(\sigma) \leqslant K_M(\sigma) + O(1).$$

Note. The constant depends on M (but not on σ).

Prefix-free (Kolmogorov) complexity

There is a partial computable prefix-free $U\colon 2^{<\omega} \to 2^{<\omega}$ such that if $M\colon 2^{<\omega} \to 2^{<\omega}$ is any other partial computable prefix-free function, then

$$K_U(\sigma) \leqslant K_M(\sigma) + O(1).$$

Note. The constant depends on M (but not on σ).

I.e., the *universal* prefix-free machine U (de-)compresses as well as any other prefix-free machine M.

Prefix-free (Kolmogorov) complexity

There is a partial computable prefix-free $U\colon 2^{<\omega} \to 2^{<\omega}$ such that if $M\colon 2^{<\omega} \to 2^{<\omega}$ is any other partial computable prefix-free function, then

$$K_U(\sigma) \leqslant K_M(\sigma) + O(1).$$

Note. The constant depends on M (but not on σ).

I.e., the *universal* prefix-free machine U (de-)compresses as well as any other prefix-free machine M.

Prefix-free (Kolmogorov) complexity

$$K(\sigma) = K_U(\sigma).$$

Martin-Löf randomness

Idea. Random sequences should be *incompressible*.

Idea. Random sequences should be *incompressible*.

Definition

$A \in 2^\omega$ is *Martin-Löf random* (1-*random*) iff

$$K(A \upharpoonright n) \geqslant n - O(1).$$

Martin-Löf randomness

Idea. Random sequences should be *incompressible*.

Definition

$A \in 2^\omega$ is *Martin-Löf random* (1-*random*) iff

$$K(A \upharpoonright n) \geqslant n - O(1).$$

Martin-Löf random sequences can also be characterized as being:

- *Unremarkable*: miss all "effective" measure zero sets (Martin-Löf, 1966)

Martin-Löf randomness

Idea. Random sequences should be *incompressible*.

Definition

$A \in 2^\omega$ is *Martin-Löf random* (1-*random*) iff

$$K(A \upharpoonright n) \geqslant n - O(1).$$

Martin-Löf random sequences can also be characterized as being:

- *Unremarkable*: miss all "effective" measure zero sets (Martin-Löf, 1966)
- *Unpredictable*: (semi-)effective betting strategies cannot win against then (Schnorr, 1971)

Martin-Löf randomness

Idea. Random sequences should be *incompressible*.

Definition

$A \in 2^\omega$ is *Martin-Löf random* (1-*random*) iff
$$K(A \upharpoonright n) \geqslant n - O(1).$$

Martin-Löf random sequences can also be characterized as being:

- *Unremarkable*: miss all "effective" measure zero sets (Martin-Löf, 1966)
- *Unpredictable*: (semi-)effective betting strategies cannot win against then (Schnorr, 1971)

Fact. Almost all sequences are Martin-Löf random.

Initial segment complexity

The initial segment complexity of a sequence tells us more than whether it is random.

Initial segment complexity

The initial segment complexity of a sequence tells us more than whether it is random.

For example, it can tell us how random it is.

Initial segment complexity

The initial segment complexity of a sequence tells us more than whether it is random.

For example, it can tell us how random it is.

Theorem

$K(A \upharpoonright n)$ is infinitely often essentially maximal ($n + K(n) + O(1)$) iff A is 2-random (random relative to $\emptyset'$, the halting problem).

Initial segment complexity

The initial segment complexity of a sequence tells us more than whether it is random.

For example, it can tell us how random it is.

Theorem

$K(A \upharpoonright n)$ is infinitely often essentially maximal ($n + K(n) + O(1)$) iff A is 2-random (random relative to $\emptyset'$, the halting problem).

Theorem (M,Yu)

Let Z be Martin-Löf random. If $K(A \upharpoonright n) \leqslant K(B \upharpoonright n) + O(1)$ and A is Martin-Löf random relative to Z, then B is Martin-Löf random relative to Z.

Effective Hausdorff dimension

We can use Kolmogorov complexity to measure the *information density* of a sequence.

Effective Hausdorff dimension

We can use Kolmogorov complexity to measure the *information density* of a sequence.

Definition

$A \in 2^\omega$ has *effective (Hausdorff) dimension*

$$\dim(A) = \liminf_{n\to\infty} \frac{K(A \upharpoonright n)}{n}.$$

Effective Hausdorff dimension

We can use Kolmogorov complexity to measure the *information density* of a sequence.

Definition

$A \in 2^\omega$ has *effective (Hausdorff) dimension*

$$\dim(A) = \liminf_{n\to\infty} \frac{K(A \upharpoonright n)}{n}.$$

So, a sequence of effective dimension $1/2$ is guaranteed to have (almost) $n/2$ bits of information in the first n bits, for all n.

Effective Hausdorff dimension

We can use Kolmogorov complexity to measure the *information density* of a sequence.

Definition

$A \in 2^\omega$ has *effective (Hausdorff) dimension*

$$\dim(A) = \liminf_{n\to\infty} \frac{K(A \upharpoonright n)}{n}.$$

So, a sequence of effective dimension $1/2$ is guaranteed to have (almost) $n/2$ bits of information in the first n bits, for all n.

But it can have much more for some n.

Observations on effective dimension

Observations on effective dimension

Originally defined by Lutz (2000) as the effective Hausdorff dimension of $\{A\}$.

Observations on effective dimension

Originally defined by Lutz (2000) as the effective Hausdorff dimension of $\{A\}$. Although the Hausdorff dimension of a singleton is always zero, the effective Hausdorff dimension may not be.

Observations on effective dimension

Originally defined by Lutz (2000) as the effective Hausdorff dimension of $\{A\}$. Although the Hausdorff dimension of a singleton is always zero, the effective Hausdorff dimension may not be.

The equivalence was proved by Mayordomo in 2002 and is essentially implicit in Ryabko (1984).

Observations on effective dimension

Originally defined by Lutz (2000) as the effective Hausdorff dimension of $\{A\}$. Although the Hausdorff dimension of a singleton is always zero, the effective Hausdorff dimension may not be.

The equivalence was proved by Mayordomo in 2002 and is essentially implicit in Ryabko (1984).

- Clearly, all Martin-Löf random sequences have effective dimension 1.

Observations on effective dimension

Originally defined by Lutz (2000) as the effective Hausdorff dimension of $\{A\}$. Although the Hausdorff dimension of a singleton is always zero, the effective Hausdorff dimension may not be.

The equivalence was proved by Mayordomo in 2002 and is essentially implicit in Ryabko (1984).

- Clearly, all Martin-Löf random sequences have effective dimension 1.
- It is easy to construct a counterexample to the converse.

Observations on effective dimension

Originally defined by Lutz (2000) as the effective Hausdorff dimension of $\{A\}$. Although the Hausdorff dimension of a singleton is always zero, the effective Hausdorff dimension may not be.

The equivalence was proved by Mayordomo in 2002 and is essentially implicit in Ryabko (1984).

- Clearly, all Martin-Löf random sequences have effective dimension 1.
- It is easy to construct a counterexample to the converse.
- The sequences in Examples 1 and 2 (with the right bias) have effective dimension $1/2$.

Formalizing the main question

Formalizing the main question

Question (Reimann, Terwijn, ~2000)

- If $0 < \mathsf{dim}(A) < 1$, does A compute a sequence of higher effective dimension?

Formalizing the main question

Question (Reimann, Terwijn, ~2000)

- If $0 < \mathsf{dim}(A) < 1$, does A compute a sequence of higher effective dimension?
- If $\mathsf{dim}(A) = 1$, does A compute a Martin-Löf random?

Formalizing the main question

Question (Reimann, Terwijn, ~2000)

- If $0 < \mathsf{dim}(A) < 1$, does A compute a sequence of higher effective dimension?
- If $\mathsf{dim}(A) = 1$, does A compute a Martin-Löf random?

The answer to both will be no.

Formalizing the main question

Question (Reimann, Terwijn, ~2000)

- If $0 < \mathsf{dim}(A) < 1$, does A compute a sequence of higher effective dimension?
- If $\mathsf{dim}(A) = 1$, does A compute a Martin-Löf random?

The answer to both will be no.

Question. What is special about the sequences we saw in Examples 1 and 2?

Formalizing the main question

Question (Reimann, Terwijn, ~2000)

- If $0 < \mathrm{dim}(A) < 1$, does A compute a sequence of higher effective dimension?
- If $\mathrm{dim}(A) = 1$, does A compute a Martin-Löf random?

The answer to both will be no.

Question. What is special about the sequences we saw in Examples 1 and 2?

Partial answer. The information they contain is spread out fairly regularly.

Effective strong dimension

Definition

$A \in 2^\omega$ has *effective strong dimension*

$$\mathrm{Dim}(A) = \limsup_{n \to \infty} \frac{K(A \upharpoonright n)}{n}.$$

Effective strong dimension

Definition

$A \in 2^\omega$ has *effective strong dimension*

$$\mathrm{Dim}(A) = \limsup_{n\to\infty} \frac{K(A \upharpoonright n)}{n}.$$

- $\mathrm{Dim}(A) \geqslant \dim(A)$.

Effective strong dimension

Definition

$A \in 2^\omega$ has *effective strong dimension*
$$\mathrm{Dim}(A) = \limsup_{n\to\infty} \frac{K(A \upharpoonright n)}{n}.$$

- $\mathrm{Dim}(A) \geqslant \dim(A)$.
- Effective strong dimension is the effective analogue of packing dimension (Athreya, Hitchcock, Lutz and Mayordomo, 2004).

Effective strong dimension

Definition

$A \in 2^\omega$ has *effective strong dimension*

$$\mathrm{Dim}(A) = \limsup_{n\to\infty} \frac{K(A \upharpoonright n)}{n}.$$

- $\mathrm{Dim}(A) \geqslant \dim(A)$.
- Effective strong dimension is the effective analogue of packing dimension (Athreya, Hitchcock, Lutz and Mayordomo, 2004).
- If $A \in 2^\omega$ is from Examples 1 or 2 (i.e., obtained through dilution or from a biased coin), then $\dim(A) = \mathrm{Dim}(A)$.

A partial (positive) result

Theorem (Bienvenu, Doty and Stephan, 2007)

If $\epsilon > 0$ and $\mathsf{Dim}(A) > 0$, then A computes a set B such that $\mathsf{dim}(B) \geqslant \mathsf{dim}(A)/\mathsf{Dim}(A) - \epsilon$.

A partial (positive) result

Theorem (Bienvenu, Doty and Stephan, 2007)

If $\epsilon > 0$ and $\mathsf{Dim}(A) > 0$, then A computes a set B such that $\mathsf{dim}(B) \geqslant \mathsf{dim}(A)/\mathsf{Dim}(A) - \epsilon$.

For example, if $\mathsf{dim}(A) = \mathsf{Dim}(A) = 1/2$, then A computes sequences with effective dimension arbitrarily close to 1.

A partial (positive) result

Theorem (Bienvenu, Doty and Stephan, 2007)

If $\epsilon > 0$ and $\mathsf{Dim}(A) > 0$, then A computes a set B such that $\mathsf{dim}(B) \geqslant \mathsf{dim}(A)/\mathsf{Dim}(A) - \epsilon$.

For example, if $\mathsf{dim}(A) = \mathsf{Dim}(A) = 1/2$, then A computes sequences with effective dimension arbitrarily close to 1.

Open Question

Is there a sequence $A \in 2^\omega$ such that $\mathsf{dim}(A) = \mathsf{Dim}(A) = 1/2$ but A does not compute a sequence of dimension 1?

A partial (positive) result

Theorem (Bienvenu, Doty and Stephan, 2007)

If $\epsilon > 0$ and $\mathsf{Dim}(A) > 0$, then A computes a set B such that $\mathsf{dim}(B) \geqslant \mathsf{dim}(A)/\mathsf{Dim}(A) - \epsilon$.

For example, if $\mathsf{dim}(A) = \mathsf{Dim}(A) = 1/2$, then A computes sequences with effective dimension arbitrarily close to 1.

Open Question

Is there a sequence $A \in 2^\omega$ such that $\mathsf{dim}(A) = \mathsf{Dim}(A) = 1/2$ but A does not compute a sequence of dimension 1?

It follows from one of our results that A need not compute a Martin-Löf random sequence.

More partial results

More partial results

Another positive one:

Theorem (Zimond, 2007)

If $A, B \in 2^\omega$ have positive effective dimension and are *sufficiently independent*, then together they compute a sequence of effective dimension 1.

More partial results

Another positive one:

Theorem (Zimond, 2007)

If $A, B \in 2^\omega$ have positive effective dimension and are *sufficiently independent*, then together they compute a sequence of effective dimension 1.

Related to a result from the theory of randomness extractors.

More partial results

Another positive one:

Theorem (Zimond, 2007)

If $A, B \in 2^\omega$ have positive effective dimension and are *sufficiently independent*, then together they compute a sequence of effective dimension 1.

Related to a result from the theory of randomness extractors.

On the negative side:

Theorem (Nies, Reimann; Bienvenu, Doty and Stephan, 2007)

There is no single algorithm that, given a sequence of effective dimension $1/2$, extracts a sequence of higher dimension.

More partial results

Another positive one:

Theorem (Zimond, 2007)

If $A, B \in 2^\omega$ have positive effective dimension and are *sufficiently independent*, then together they compute a sequence of effective dimension 1.

Related to a result from the theory of randomness extractors.

On the negative side:

Theorem (Nies, Reimann; Bienvenu, Doty and Stephan, 2007)

There is no single algorithm that, given a sequence of effective dimension $1/2$, extracts a sequence of higher dimension.

Perhaps the algorithm simply needs extra information, such as the strong dimension.

Theorem

There is an $A \in 2^\omega$ such that $\mathsf{dim}(A) = 1/2$ and A does not compute a sequence of higher effective dimension.

Extracting randomness is hard

Theorem

There is an $A \in 2^\omega$ such that $\dim(A) = 1/2$ and A does not compute a sequence of higher effective dimension.

This is proved using a novel forcing partial order.

Extracting randomness is hard

Theorem

There is an $A \in 2^\omega$ such that $\mathrm{dim}(A) = 1/2$ and A does not compute a sequence of higher effective dimension.

This is proved using a novel forcing partial order.

The forcing conditions—in other words, the approximations to A used in the construction—are (certain special) Π^0_1 classes whose measures have effective dimension $1/2$.

Extracting randomness is hard

Theorem

There is an $A \in 2^\omega$ such that $\mathsf{dim}(A) = 1/2$ and A does not compute a sequence of higher effective dimension.

This is proved using a novel forcing partial order.

The forcing conditions—in other words, the approximations to A used in the construction—are (certain special) Π^0_1 classes whose measures have effective dimension $1/2$.

Note. By the Bienvenu, Doty, Stephan result, $\mathsf{Dim}(A) = 1$.

Optimal covers

Definition

Let $S \subseteq 2^{<\omega}$. The *direct weight* of S is

$$\mathsf{DW}(S) = \sum_{\sigma \in S} 2^{-|\sigma|/2}.$$

Optimal covers

Definition

Let $S \subseteq 2^{<\omega}$. The *direct weight* of S is

$$\mathsf{DW}(S) = \sum_{\sigma \in S} 2^{-|\sigma|/2}.$$

The *weight* of S is

$$\mathsf{W}(S) = \inf\{\mathsf{DW}(V) \colon [S] \subseteq [V]\}.$$

Optimal covers

Definition

Let $S \subseteq 2^{<\omega}$. The *direct weight* of S is

$$\mathsf{DW}(S) = \sum_{\sigma \in S} 2^{-|\sigma|/2}.$$

The *weight* of S is

$$\mathsf{W}(S) = \inf\{\mathsf{DW}(V) \colon [S] \subseteq [V]\}.$$

$S^{oc} \subseteq 2^{<\omega}$ is the *optimal cover* of $S \subseteq 2^{<\omega}$ if $[S] \subseteq [S^{oc}]$ and $\mathsf{DW}(S^{oc}) = \mathsf{W}(S)$.

Optimal covers

Definition

Let $S \subseteq 2^{<\omega}$. The *direct weight* of S is

$$\mathsf{DW}(S) = \sum_{\sigma \in S} 2^{-|\sigma|/2}.$$

The *weight* of S is

$$\mathsf{W}(S) = \inf\{\mathsf{DW}(V) \colon [S] \subseteq [V]\}.$$

$S^{oc} \subseteq 2^{<\omega}$ is the *optimal cover* of $S \subseteq 2^{<\omega}$ if $[S] \subseteq [S^{oc}]$ and $\mathsf{DW}(S^{oc}) = \mathsf{W}(S)$.

For the sake of uniqueness, we also require $[S^{oc}]$ to have the minimum measure among all possible contenders.

Optimal covers

Definition

Let $S \subseteq 2^{<\omega}$. The *direct weight* of S is

$$\mathsf{DW}(S) = \sum_{\sigma \in S} 2^{-|\sigma|/2}.$$

The *weight* of S is

$$\mathsf{W}(S) = \inf\{\mathsf{DW}(V) \colon [S] \subseteq [V]\}.$$

$S^{oc} \subseteq 2^{<\omega}$ is the *optimal cover* of $S \subseteq 2^{<\omega}$ if $[S] \subseteq [S^{oc}]$ and $\mathsf{DW}(S^{oc}) = \mathsf{W}(S)$.

For the sake of uniqueness, we also require $[S^{oc}]$ to have the minimum measure among all possible contenders.

If S is c.e., then S^{oc} is clearly Δ^0_2.

Optimal covers

Definition

Let $S \subseteq 2^{<\omega}$. The *direct weight* of S is

$$\mathsf{DW}(S) = \sum_{\sigma \in S} 2^{-|\sigma|/2}.$$

The *weight* of S is

$$\mathsf{W}(S) = \inf\{\mathsf{DW}(V)\colon [S] \subseteq [V]\}.$$

$S^{oc} \subseteq 2^{<\omega}$ is the *optimal cover* of $S \subseteq 2^{<\omega}$ if $[S] \subseteq [S^{oc}]$ and $\mathsf{DW}(S^{oc}) = \mathsf{W}(S)$.

For the sake of uniqueness, we also require $[S^{oc}]$ to have the minimum measure among all possible contenders.

If S is c.e., then S^{oc} is clearly Δ^0_2.

More importantly, $[S^{oc}]$ is a Σ^0_1 class.

Our conditions are pairs $\langle \sigma, S \rangle$ such that

- $\sigma \in 2^{<\omega}$,
- $S \subseteq [\sigma]^{<\omega}$ is a c.e. set, and
- $\sigma \notin S^{oc}$.

The forcing conditions

Our conditions are pairs $\langle \sigma, S \rangle$ such that

- $\sigma \in 2^{<\omega}$,
- $S \subseteq [\sigma]^{<\omega}$ is a c.e. set, and
- $\sigma \notin S^{oc}$.

The set of all sequences consistent with a condition $\langle \sigma, S \rangle$ is the Π^0_1 class

$$P_{\langle \sigma, S \rangle} = [\sigma] \setminus [S^{oc}].$$

The forcing conditions

Our conditions are pairs $\langle \sigma, S \rangle$ such that

- $\sigma \in 2^{<\omega}$,
- $S \subseteq [\sigma]^{<\omega}$ is a c.e. set, and
- $\sigma \notin S^{oc}$.

The set of all sequences consistent with a condition $\langle \sigma, S \rangle$ is the Π^0_1 class

$$P_{\langle \sigma, S \rangle} = [\sigma] \smallsetminus [S^{oc}].$$

The most important property conditions have:

Lemma

Let $\langle \sigma, S \rangle$ be a condition. Then $\dim(\mu(P_{\langle \sigma, S \rangle})) \leqslant 1/2$.

Effective dimension 1

The proof of the previous result can be modified to show:

Theorem

There is an $A \in 2^\omega$ such that $\dim(A) = 1$ and A does not compute a Martin-Löf random.

Effective dimension 1

The proof of the previous result can be modified to show:

Theorem

There is an $A \in 2^\omega$ such that $\dim(A) = 1$ and A does not compute a Martin-Löf random.

A different technique produces a much stronger result.

Effective dimension 1

The proof of the previous result can be modified to show:

Theorem

There is an $A \in 2^\omega$ such that $\dim(A) = 1$ and A does not compute a Martin-Löf random.

A different technique produces a much stronger result.

Definition

$A \in 2^\omega$ has *minimal (Turing) degree* if, for every sequence B computable from A, either

Effective dimension 1

The proof of the previous result can be modified to show:

Theorem

There is an $A \in 2^\omega$ such that $\dim(A) = 1$ and A does not compute a Martin-Löf random.

A different technique produces a much stronger result.

Definition

$A \in 2^\omega$ has *minimal (Turing) degree* if, for every sequence B computable from A, either

- B is computable, or

Effective dimension 1

The proof of the previous result can be modified to show:

Theorem

There is an $A \in 2^\omega$ such that $\dim(A) = 1$ and A does not compute a Martin-Löf random.

A different technique produces a much stronger result.

Definition

$A \in 2^\omega$ has *minimal (Turing) degree* if, for every sequence B computable from A, either

- B is computable, or
- B computes A.

Effective dimension 1

Theorem (Greenberg, M)

There is an $A \in 2^\omega$ such that $\mathsf{dim}(A) = 1$ and A has minimal Turing degree.

Effective dimension 1

Theorem (Greenberg, M)

There is an $A \in 2^\omega$ such that $\mathsf{dim}(A) = 1$ and A has minimal Turing degree.

It is well known that a Martin-Löf random cannot have minimal degree (nor can it be computable).

Effective dimension 1

Theorem (Greenberg, M)

There is an $A \in 2^\omega$ such that $\dim(A) = 1$ and A has minimal Turing degree.

It is well known that a Martin-Löf random cannot have minimal degree (nor can it be computable). So A does not compute a Martin-Löf random.

Effective dimension 1

Theorem (Greenberg, M)

There is an $A \in 2^\omega$ such that $\dim(A) = 1$ and A has minimal Turing degree.

It is well known that a Martin-Löf random cannot have minimal degree (nor can it be computable). So A does not compute a Martin-Löf random.

We reduced the problem to (the proof of) the following result:

Theorem (Kumabe, 1996; Kumabe, Lewis)

There is a DNC function of minimal Turing degree.

The computational strength of Pravda

The computational strength of Pravda

Definition (Diagonally non-computable)

A function $\mathsf{f}\colon \omega \to \omega$ is *DNC* if $\mathsf{f}(e)$ is not the output of the eth program (on input e), for all e.

The computational strength of Pravda

Definition (Diagonally non-computable)

A function $\mathsf{f}\colon \omega \to \omega$ is *DNC* if $\mathsf{f}(e)$ is <u>not</u> the output of the eth program (on input e), for all e.

So, f cannot tell you whether the eth program halts;

The computational strength of Pravda

Definition (Diagonally non-computable)

A function $f\colon \omega \to \omega$ is *DNC* if $f(e)$ is <u>not</u> the output of the eth program (on input e), for all e.

So, f cannot tell you whether the eth program halts; it just gives you one number guaranteed not to be its output if it does.

Definition (Diagonally non-computable)

A function $f\colon \omega \to \omega$ is *DNC* if $f(e)$ is not the output of the eth program (on input e), for all e.

So, f cannot tell you whether the eth program halts; it just gives you one number guaranteed not to be its output if it does.

Note

If f is DNC, then f is not computable.

The computational strength of Pravda

Definition (Diagonally non-computable)

A function $\mathsf{f}\colon \omega \to \omega$ is *DNC* if $\mathsf{f}(e)$ is not the output of the eth program (on input e), for all e.

So, f cannot tell you whether the eth program halts; it just gives you one number guaranteed not to be its output if it does.

Note

If f is DNC, then f is not computable.

But what can you do with a DNC function?

You can do a little…for example:

Theorem (M, Nies)

If f is DNC, then either

The computational strength of Pravda

You can do a little…for example:

Theorem (M, Nies)

If f is DNC, then either

- f computes a total function that is not dominated by any computable function (i.e., f has *hyperimmune degree*), or

The computational strength of Pravda

You can do a little…for example:

Theorem (M, Nies)

If f is DNC, then either

- f computes a total function that is not dominated by any computable function (i.e., f has *hyperimmune degree*), or
- f enumerates a set that is not computable using both f and $\emptyset'$, the halting problem, (i.e., f is not *generalized low*).

The computational strength of Pravda

You can do a little…for example:

Theorem (M, Nies)

If f is DNC, then either

- f computes a total function that is not dominated by any computable function (i.e., f has *hyperimmune degree*), or
- f enumerates a set that is not computable using both f and $\emptyset'$, the halting problem, (i.e., f is not *generalized low*).

Theorem (Greenberg, M; Kjos-Hanssen)

If f is DNC, then it computes an infinite subset of some Martin-Löf random set.

You can do a little…for example:

Theorem (M, Nies)

If f is DNC, then either

- f computes a total function that is not dominated by any computable function (i.e., f has *hyperimmune degree*), or
- f enumerates a set that is not computable using both f and $\emptyset'$, the halting problem, (i.e., f is not *generalized low*).

Theorem (Greenberg, M; Kjos-Hanssen)

If f is DNC, then it computes an infinite subset of some Martin-Löf random set.

…but not much. Might not compute a sequence with positive dimension (Ambos-Spies, Kjos-Hanssen, Lempp and Slaman).

Bounded DNC

Definition

Let $h\colon \omega \to \omega \setminus \{0, 1\}$. We say that $f\colon \omega \to \omega$ is h-*DNC* if f is DNC and $f(e) < h(e)$, for all e.

Bounded DNC

Definition

Let $h\colon \omega \to \omega \setminus \{0, 1\}$. We say that $f\colon \omega \to \omega$ is h-*DNC* if f is DNC and $f(e) < h(e)$, for all e.

A constant bound guarantees computational strength:

Theorem (Various)

If f is 2-DNC, then it computes:

Bounded DNC

Definition

Let $h\colon \omega \to \omega \setminus \{0, 1\}$. We say that $f\colon \omega \to \omega$ is h-*DNC* if f is DNC and $f(e) < h(e)$, for all e.

A constant bound guarantees computational strength:

Theorem (Various)

If f is 2-DNC, then it computes:

- A prime ideal in every computable (countable) commutative ring.
- A fixed point for every computable $g\colon [0, 1]^2 \to [0, 1]^2$.

Bounded DNC

Definition

Let $h\colon \omega \to \omega \setminus \{0,1\}$. We say that $f\colon \omega \to \omega$ is h-*DNC* if f is DNC and $f(e) < h(e)$, for all e.

A constant bound guarantees computational strength:

Theorem (Various)

If f is 2-DNC, then it computes:

- A prime ideal in every computable (countable) commutative ring.
- A fixed point for every computable $g\colon [0,1]^2 \to [0,1]^2$.

Such functions also compute Martin-Löf random sequences, so they cannot have minimal Turing degree.

h-Bounded DNC

The Kumabe and Lewis proof actually shows:

h-Bounded DNC

The Kumabe and Lewis proof actually shows:

Theorem (Kumabe, Lewis)

For any computable, unbounded, nondecreasing $h \colon \omega \to \omega \setminus \{0, 1\}$, there is an h-DNC function of minimal Turing degree.

h-Bounded DNC

The Kumabe and Lewis proof actually shows:

Theorem (Kumabe, Lewis)

For any computable, unbounded, nondecreasing $h\colon \omega \to \omega \setminus \{0, 1\}$, there is an h-DNC function of minimal Turing degree.

So, all we had to prove was:

Theorem (Greenberg, M)

There is a computable, unbounded, nondecreasing $h\colon \omega \to \omega \setminus \{0, 1\}$ such that every h-DNC function computes a sequence with dimension 1.

h-Bounded DNC

The Kumabe and Lewis proof actually shows:

Theorem (Kumabe, Lewis)

For any computable, unbounded, nondecreasing $h\colon \omega \to \omega \setminus \{0,1\}$, there is an h-DNC function of minimal Turing degree.

So, all we had to prove was:

Theorem (Greenberg, M)

There is a computable, unbounded, nondecreasing $h\colon \omega \to \omega \setminus \{0,1\}$ such that every h-DNC function computes a sequence with dimension 1.

…to get a sequence with dimension 1 and minimal degree.

h-Bounded DNC and uniform computation

What can be computed from an h-DNC function, for a sufficiently slow growing h, is related to what can be *uniformly* computed from a n-DNC function for all n.

h-Bounded DNC and uniform computation

What can be computed from an h-DNC function, for a sufficiently slow growing h, is related to what can be *uniformly* computed from a n-DNC function for all n.

Theorem (Jockusch, 1989)

An n-DNC function computes a 2-DNC function.

h-Bounded DNC and uniform computation

What can be computed from an h-DNC function, for a sufficiently slow growing h, is related to what can be *uniformly* computed from a n-DNC function for all n.

Theorem (Jockusch, 1989)

An n-DNC function computes a 2-DNC function. *But this can not be done uniformly for* $n > 2$.

h-Bounded DNC and uniform computation

What can be computed from an h-DNC function, for a sufficiently slow growing h, is related to what can be *uniformly* computed from a n-DNC function for all n.

Theorem (Jockusch, 1989)

An n-DNC function computes a 2-DNC function. *But this can not be done uniformly for* $n > 2$.

Theorem (Greenberg, M)

There is a *uniform* procedure to compute a sequence of dimension 1 from an n-DNC function.

h-Bounded DNC and uniform computation

What can be computed from an h-DNC function, for a sufficiently slow growing h, is related to what can be *uniformly* computed from a n-DNC function for all n.

Theorem (Jockusch, 1989)

An n-DNC function computes a 2-DNC function. *But this can not be done uniformly for* $n > 2$.

Theorem (Greenberg, M)

There is a *uniform* procedure to compute a sequence of dimension 1 from an n-DNC function.

It is open whether 3-DNC functions compute Martin-Löf random sequences *uniformly*.

An application

Bjørn Kjos-Hanssen used (a relativization of) the existence of dimension 1 sequence of minimal Turing degree to show:

An application

Bjørn Kjos-Hanssen used (a relativization of) the existence of dimension 1 sequence of minimal Turing degree to show:

Theorem (Kjos-Hanssen)

The Hausdorff dimension of the class of minimal Turing degrees is 1.

An application

Bjørn Kjos-Hanssen used (a relativization of) the existence of dimension 1 sequence of minimal Turing degree to show:

Theorem (Kjos-Hanssen)

The Hausdorff dimension of the class of minimal Turing degrees is 1.

Note: Kurtz proved that (even the upward closure of) this class has measure 0.

Another open question

Can the two main results be combined?

Another open question

Can the two main results be combined? I.e., is there a minimal degree of fractional dimension?

Another open question

Can the two main results be combined? I.e., is there a minimal degree of fractional dimension?

Open Question

Is there a sequence $A \in 2^\omega$ of minimal degree such that $0 < \dim(A) < 1$ and A does not compute a sequence of higher effective dimension (or at least, does not compute sequences of dimension arbitrarily close to 1)?

Thank You