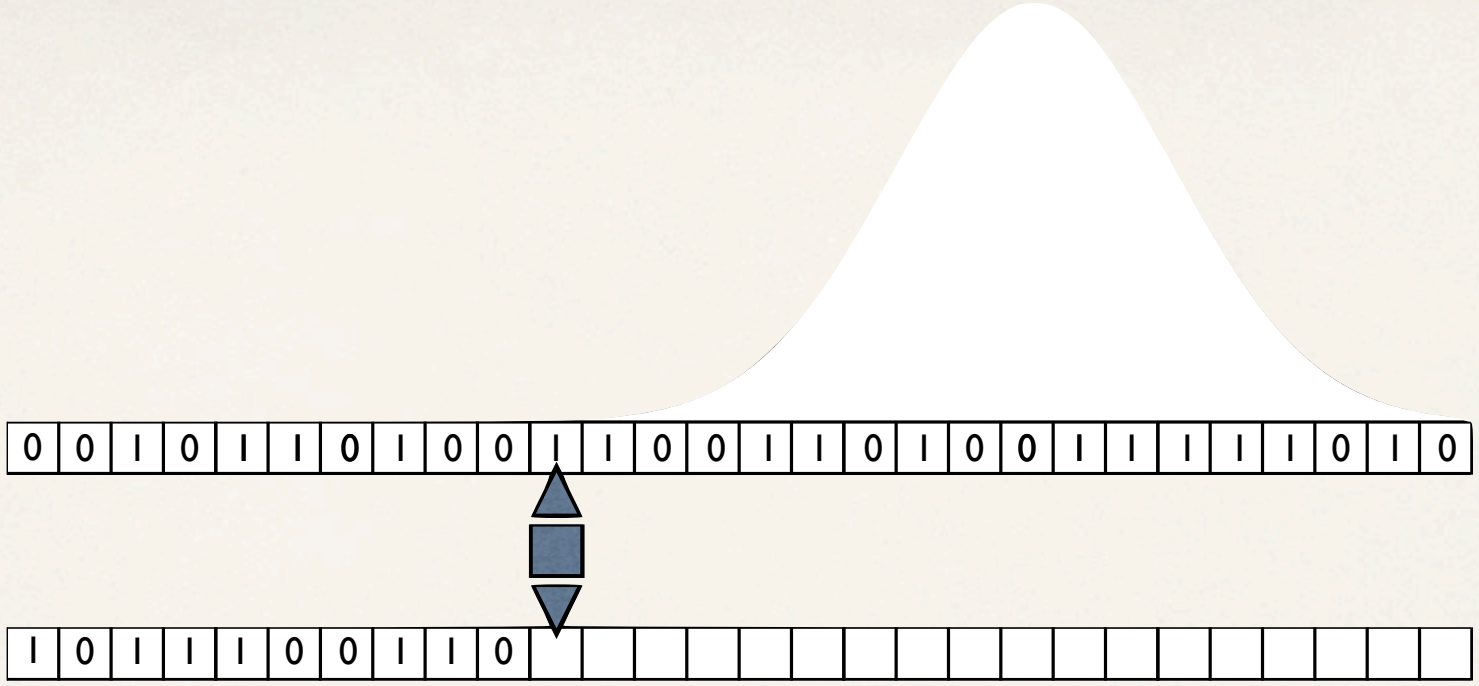



# Algorithmic Randomness and Probabilistic Computation

Christopher Porter, LIAFA

*Séminaire Automates*

*June 14, 2013*

# Motivation

In computability theory, much research has focused on determining which problems are solvable by Turing machines and which ones are not.

- Example: For a given set of natural numbers $S$, is there an effective procedure for determining membership in $S$?

If a given problem is shown to be effectively unsolvable, we can further ask: Just how unsolvable is it?

In fact, there are a number of hierarchies for classifying the difficulty of solving various problems.

# An alternative approach

In these investigations of the unsolvability of problems, the computations are carried out by Turing machines (often equipped with an oracle).

What would happen if we were to work with some model of a probabilistic Turing machine?

In this talk, I will discuss recent work with Laurent Bienvenu and Antoine Taveneaux on the picture that emerges when we consider a specific model of probabilistic computation that draws upon the theory of algorithmic randomness.

# Probabilistic computation

As computability theorists, we don't want to stray too far from Turing's original model of oracle computation.

One model that we could use is given by a Turing machine with an oracle full of randomly generated bits (for instance, produced by the repeated tosses of a fair coin).

Since with probability one we will produce an algorithmically random sequence by repeatedly tossing a fair coin, we can assume that the oracle tape of our machine contains an algorithmically random sequence.

# Algorithmic randomness

There are a number of non-equivalent definition of algorithmic randomness (several of which will be discussed shortly), and thus a question naturally arises:

Which definition of algorithmic
randomness should the sequence
on our oracle tape satisfy?

In some cases, it won't matter which definition of algorithmic randomness we work with, but in other cases, we will be very sensitive to the notion of algorithmic randomness that is used as our oracle.

# Our main questions

We will focus on two sorts of questions:

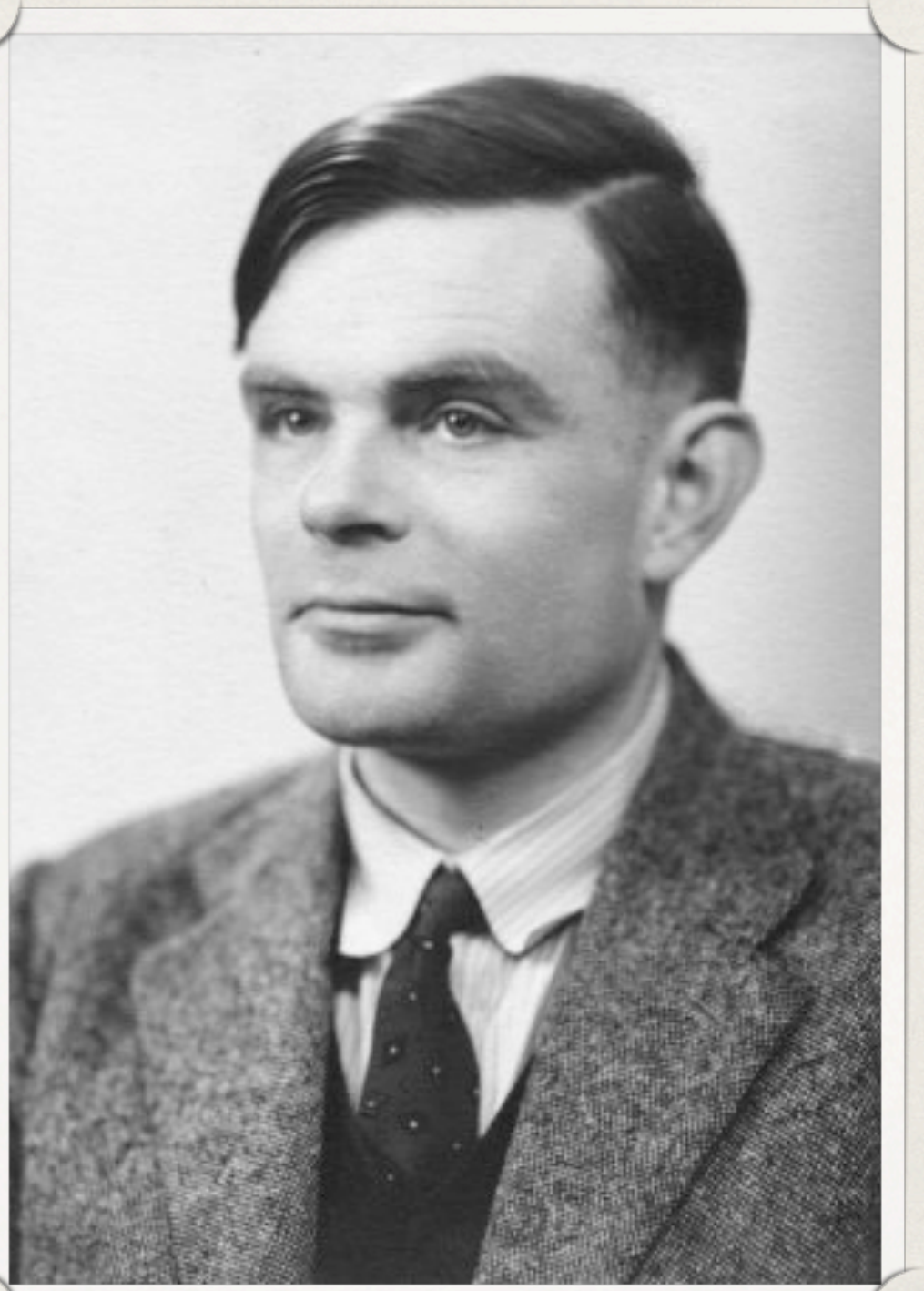
First, we will consider the **power** of computing with random oracles:

(1) What level of randomness is necessary to guarantee a solution to a given problem?

Second, we will consider the **limitations** of computing with random oracles:

(2) Which problems are not solvable with positive probability by any Turing machine equipped with a sufficiently random oracle?

# A Bit of Computability Theory



# Fixing some notation, I

| | |
|---|---|
| $2^{<\omega}$ | the collection of finite binary strings |
| $\sigma, \tau, \ldots$ | members of $2^{<\omega}$ |
| $\sigma \preceq \tau$ | $\sigma$ is an initial segment of $\tau$ (or $\sigma = \tau$) |
| $2^{\omega}$ | the collection of infinite binary sequences |
| $X, Y, \ldots$ | members of $2^{\omega}$ |
| $\sigma \prec X$ | $\sigma$ is an initial segment of $X$ |

# Fixing some notation, II

The standard topology on $2^\omega$ is given by basic open sets of the form

$$[\![\sigma]\!] = \{X \in 2^\omega : \sigma \prec X\}$$

for $\sigma \in 2^{<\omega}$.

The Lebesgue measure on $2^\omega$, denoted $\lambda$, is defined by

$$\lambda([\![\sigma]\!]) = 2^{-|\sigma|}$$

for $\sigma \in 2^{<\omega}$ (where $|\sigma|$ is the length of $\sigma$), and then we extend $\lambda$ to all Borel sets in the usual manner.

# Computable functions

Let $\{\phi_i\}_{i\in\omega}$ be the collection of partial computable functions.

Recall that a set $S$ is computable if the characteristic function of $S$, $\chi_S$, can be computed by a total computable function.

For $A \in 2^\omega$, if we allow our computations to access $A$ as an oracle, this yields the collection of partial computable functions relative to $A$, $\{\phi_i^A\}_{i\in\omega}$.

# Turing reducibility

Let $A, B \in 2^\omega$.

If there is a $B$-partial computable function $\phi^B$ such that $\phi^B = \chi_A$, we say that $A$ is **Turing reducible** to $B$ (or $B$ computes $A$), denoted

$$A \leq_T B.$$

Further, if $A \leq_T B$ and $B \leq_T A$, then we say that $A$ is **Turing equivalent** to $B$, denoted

$$A \equiv_T B.$$

The **Turing degree** of $A$ is defined to be

$$\deg_T(A) = \{B \in 2^\omega : A \equiv_T B\}.$$

# The halting problem

The halting problem is the set $K = \{e : \phi_e(e)\downarrow\}$.

The Turing degree of the halting problem is called $\emptyset'$.

As we will see, this Turing degree is particularly important for our discussion.

# $\Pi^0_1$ Classes

We will also make use of what are known as $\Pi^0_1$ classes.

Let $T$ be a computable tree (a subset of $2^{<\omega}$ that is closed downwards under $\preceq$). Then the set of infinite paths through $T$ is denoted $[T]$.

A collection $\mathcal{P} \subseteq 2^\omega$ is a $\Pi^0_1$ class if there is some computable tree $T$ such that $\mathcal{P} = [T]$.

Note: $\Pi^0_1$ classes are the effectively closed subclasses of $2^\omega$.

# Completions of Peano arithmetic

A very nice and useful example of a $\Pi^0_1$ class is given by the collection of consistent completions of Peano arithmetic.

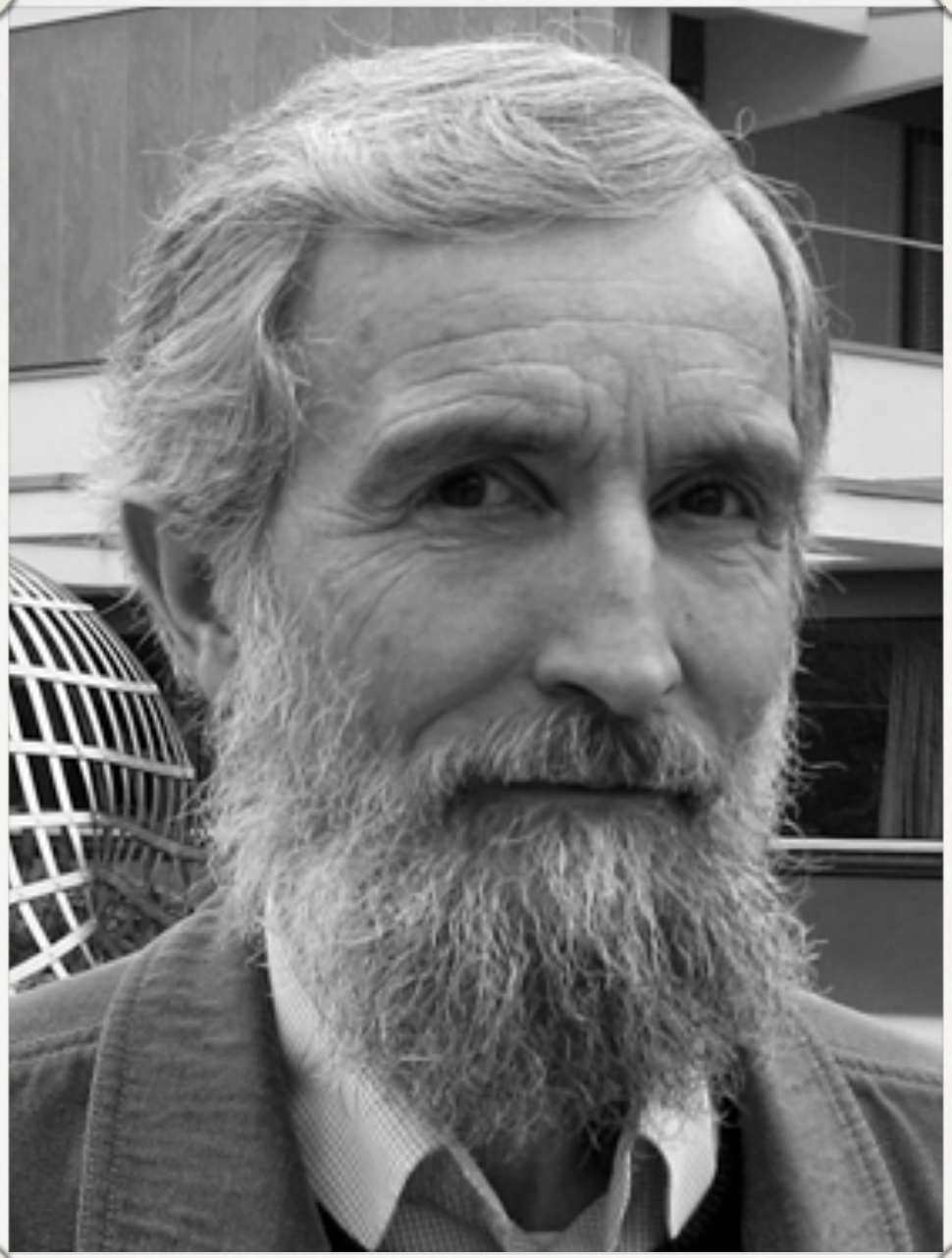
By Gödel's incompleteness theorem, there is an infinite collection of sentences not decidable by PA.

As we move through the tree determined by Gödel numbers of formulas of arithmetic (where the branching at the $n$th level of the tree corresponds to the $n$th formula and its negation), when we arrive at a level corresponding to an undecidable sentence $G$, we can consistently add $G$ or its negation.

# Sets of PA degree

Let us say of $X \in 2^\omega$ that it has **PA degree** if there is some completion of Peano arithmetic $A \in 2^\omega$ such that $A \leq_T X$.

Fact: Every sequence that computes the halting problem computes a completion of PA, but there are many sequences of PA degree that fail to compute the halting problem.



# A Bit of Algorithmic Randomness

# The main idea

There are a number of ways to motivate the idea of algorithmic randomness. For instance:

- A sequence is algorithmically random if it does not have any effectively detectable regularities.
- A sequence algorithmically random if it satisfies every effectively specifiable law of probability.

These two ideas can be made precise in such a way that the resulting definitions of algorithmic randomness are equivalent.

# Martin-Löf tests

A **Martin-Löf test** is a uniform sequence $(\mathcal{U}_i)_{i\in\omega}$ of $\Sigma^0_1$ (effectively open) subsets of $2^\omega$ such that for each $i \in \omega$,

$$\lambda(\mathcal{U}_i) \leq 2^{-i}.$$

A sequence $X \in 2^\omega$ **passes a Martin-Löf test** $(\mathcal{U}_i)_{i\in\omega}$ if

$$X \notin \bigcap_{i\in\omega} \mathcal{U}_i.$$

An important fact is that there is a **universal** Martin-Löf test; that is, there is a Martin-Löf test $(\widehat{\mathcal{U}}_i)_{i\in\omega}$ such that for every Martin-Löf test $(\mathcal{U}_i)_{i\in\omega}$,

$$\bigcap_{i\in\omega} \mathcal{U}_i \subseteq \bigcap_{i\in\omega} \widehat{\mathcal{U}}_i.$$

# Martin-Löf randomness

A sequence $X \in 2^\omega$ is **Martin-Löf random** if $X$ passes every Martin-Löf test, or equivalently, if $X$ passes the universal Martin-Löf test.

Martin-Löf random sequences satisfy all of the standard properties of "randomly generated sequences".

- the law of large numbers;
- the law of the iterated logarithm;
- any "law of probability" that can be effectively enumerated.

Despite these nice properties, there are Martin-Löf random sequences that are outliers (in some sense).

# "Ill-behaved" random sequences

**Theorem** (Kučera-Gács): For every $X \geq \emptyset'$, there is some Martin-Löf random sequence $Y$ such that $X \equiv_T Y$.

What's more, these are precisely the Martin-Löf random sequences of PA degree.

**Theorem** (Stephan): For every Martin-Löf random sequence $X$, $X \geq \emptyset'$ if and only if $X$ has PA degree.

# Relative randomness

The notion of Martin-Löf randomness can be relativized to an oracle.

For $A \in 2^\omega$, an ***A*-Martin-Löf test** is a uniform sequence $(\mathcal{U}_i^A)_{i\in\omega}$ of $\Sigma^0_1(A)$ (effectively open relative to $A$) subsets of $2^\omega$ such that for each $i \in \omega$,

$$\lambda(\mathcal{U}_i) \leq 2^{-i}.$$

$X \in 2^\omega$ is ***A*-Martin-Löf random** if $X$ passes every $A$-Martin-Löf test.

# 2-randomness

If we use $\emptyset'$ as an oracle while enumerating our tests, the resulting notion of randomness is known as **2-randomness**.

2-randomness is strictly stronger than Martin-Löf randomness.

For instance, no 2-random sequence can compute $\emptyset'$.

In general, 2-random sequences behave more like randomly generated sequences than Martin-Löf random sequences do.

# Difference randomness

One last definition of randomness that will be useful for us is known as difference randomness.

A **difference test** is a uniform sequence $((\mathcal{U}_i, \mathcal{V}_i)_{i\in\omega}$ of pairs of $\Sigma^0_1$ -classes such that for each $i \in \omega$,

$$\lambda(\mathcal{U}_i \setminus \mathcal{V}_i) \leq 2^{-i}.$$

$X \in 2^\omega$ is **difference random** if $X$ passes every difference test.

# A surprising result

**Theorem** (Franklin, Ng): A sequence $X$ is difference random if and only if $X$ is Martin-Löf random and $X \not\geq_T \emptyset'$ (if and only if $X$ is Martin-Löf random and does not have PA degree).

Thus, difference random sequences are slightly more well-behaved than Martin-Löf random sequences.

Still, difference randomness is weaker than 2-randomness: there are difference random sequences that are computable from $\emptyset'$, but no 2-random sequence has this property.


# The Limitations of Probabilistic Computation

# Computing individual sequences

Using our model of probabilistic computation, do we have additional power to compute individual sequences?

That is, are there non-computable sequences that are computable with positive probability by a Turing machine with a randomly generated oracle?

Interestingly, the answer is “No”.

# Sacks' Theorem

**Theorem** (Sacks): Suppose there is some $A \in 2^\omega$ and an oracle Turing machine $\phi$ such that

$$\lambda(\{X : \phi^X = \chi_A\} > 0.$$

Then $A$ is computable.

Proof sketch: By the Lebesgue density theorem, there is some $\sigma \in 2^{<\omega}$ such that more than $2/3$ of the sequences in the basic open set $[\![\sigma]\!]$ compute $A$. We can then compute the values of $A$ by majority vote.

# Computing members of $\Pi^0_1$ classes

Even though probabilistic computation does not allow us to compute any individual non-computable sequence with positive probability, we can probabilistically compute a member of certain $\Pi^0_1$ classes with positive probability. For example:

- any $\Pi^0_1$ class consisting entirely of Martin-Löf random sequences (which must have positive Lebesgue measure);

Are there $\Pi^0_1$ classes whose members are difficult to compute probabilistically?

# Negligible $\Pi^0_1$ classes

A $\Pi^0_1$ class $\mathcal{P} \subseteq 2^\omega$ is **negligible** if for every oracle Turing machine $\phi$, the probability of computing a member of $\mathcal{P}$ using $\phi$ equipped with a randomly generated oracle is 0. That is,

$$\sum_{i \in \omega} \lambda(\{X \in 2^\omega : \phi_i^X \in \mathcal{P}\}) = 0.$$

We've already encountered an example of a negligible $\Pi^0_1$ class, namely the collection of consistent completions of Peano arithmetic.

# Negligibility and randomness

**Proposition** (BPT): If $\mathcal{P} \subseteq 2^\omega$ is a negligible $\Pi^0_1$ class, then $\mathcal{P}$ does not contain any sequence that is random with respect to any computable probability measure.

If a $\Pi^0_1$ class $\mathcal{P}$ does not contain any sequence that is random with respect to any computable probability measure, does it follow that $\mathcal{P}$ is negligible?

**Proposition** (BPT): There is a non-negligible $\Pi^0_1$ class that contains no sequence that is random with respect to any computable probability measure.

# Deep $\Pi^0_1$ classes

Closely related to negligible $\Pi^0_1$ classes are what we refer to as "deep" $\Pi^0_1$ classes.

A $\Pi^0_1$ class $\mathcal{P} \subseteq 2^\omega$ is **deep** if there is a computable function $h$ such that for every $n$ the probability of computing some $\sigma \in 2^{<\omega}$ at the $n$th level of the tree $T$ such that $\mathcal{P} = [T]$ via any Turing machine equipped with a randomly generated oracle is bounded above by $2^{-h(n)}$.

# Some facts about deep classes

The collection of consistent completions of PA form a deep class (Levin).

We've identified a number of other deep classes that naturally occur in computability theory (shift-complex sequences, compression functions, certain sequences related to diagonal non-computability).

Further, we've established the level of randomness at which computing members of deep classes becomes impossible:

**Theorem** (BPT): Any Martin-Löf random sequence that computes a member of a deep $\Pi^0_1$ class is not difference random.

# The Power of Probabilistic Computation

# The main idea

We now shift gears to discuss the computational power of a Turing machine equipped with a random oracle.

We will consider computational power of a very specific kind:

(1) We consider properties that are satisfied by almost every Turing degree.

(2) For each such property, we can calibrate the level of randomness necessary for the property to hold.

# Two key examples

In the early 1980s, Kurtz identified a number of properties that are satisfied by almost every sequence. Let's consider two examples:

- almost every sequence computes a function that is not dominated by any computable function (such a sequence is said to have **hyperimmune degree**);
- almost every sequence computes a 1-generic sequence (an effective analogue of a Cohen generic sequence).

# Kautz's improvement

In the early 1990s, Kautz improved Kurtz's results by identifying a level of randomness that is sufficient to guarantee that these properties are satisfied:

**Theorem** (Kautz): (i) Every 2-random sequence computes a function not dominated by any computable function.
(ii) Every 2-random sequence computes a 1-generic sequence.

# Some details, I

Kurtz's original results can be recast in terms of a probabilistic algorithm that succeeds with positive probability.

For example, to show that almost every sequence computes a function that is not dominated by any computable function, the probabilistic algorithm defines a function $f$ in terms of the collection of partial computable functions $\{\phi_i\}_{i\in\omega}$.

For each partial computable function $\phi_i$, we do not know if it is total, so we will use our random oracle to either guess that $\phi_i$ is defined on some sufficiently large number $N$ or that it is undefined on $N$.

# Some details, II

If we correctly guess that $\phi_i$ is defined on $N$, we will eventually see $\phi_i(N)\downarrow$ and then we can define $f(N) = \phi_i(N) + 1$.

If we correctly guess that $\phi_i$ is undefined on $N$, this won't affect us, since then $\phi_i$ cannot be total.

If we incorrectly guess that $\phi_i$ is undefined on $N$, we will eventually see that our guess was incorrect, and we can make another guess.

If we incorrectly guess that $\phi_i$ is defined on $N$, then we are in trouble, as we will wait forever to see $\phi_i(N)\downarrow$, which will never happen.

# A key observation

Corresponding to the action taken for each of the partial computable functions is the difference of two $\Sigma^0_1$ classes. This yields a difference test $((\mathcal{U}_i, \mathcal{V}_i)_{i\in\omega}$.

The collection of sequences for which this algorithm fails are contained in infinitely many levels of the difference test.

Recall that $X$ passes a difference test if and only if

$$X \notin \bigcap_{i\in\omega} (\mathcal{U}_i \setminus \mathcal{V}_i).$$

This does not rule out the possibility that $X \in \mathcal{U}_i \setminus \mathcal{V}_i$ for infinitely many $i$.

# Strong difference randomness

If we define a sequence to be random if and only if it is not contained in *infinitely many levels* of a difference test, we get a new notion of randomness that we call **strong difference randomness**.

Every strongly difference random sequence is difference random, but not vice versa. Moreover, every 2-random sequence is strongly difference random, but not vice versa.

# The power of SDR

If a sequence is strongly difference random, then the probabilistic algorithm is guaranteed to succeed in producing a fast-growing function. Thus we have:

**Theorem** (BP): Every strongly difference random sequence computes a function not dominated by any computable function.

Moreover, one can show that there is a similar probabilistic algorithm for computing a 1-generic.

**Theorem** (BP): Every strongly difference random sequence computes a 1-generic.

# In conclusion

Algorithmic randomness provides a number of powerful tools for analyzing both the power and limitations of probabilistic computation.

We have only begun to scratch the surface in our investigations, and we hope to find further uses of probabilistic computation in computability theory.

Merci!