Algorithmic versus Quantum Randomness

Cristian S. Calude

Vienna, November 2007

In non-scientific parlance, the word random is used to express lack of order, purpose, cause, or predictability.

In non-scientific parlance, the word random is used to express lack of order, purpose, cause, or predictability.

Ramsey theory shows there is no such thing as "true randomness".

In non-scientific parlance, the word random is used to express lack of order, purpose, cause, or predictability.

Ramsey theory shows there is no such thing as "true randomness".

Algorithmic randomness, probably the best model of 'randomness', was intensively studied in the last 40 years by algorithmic information theory.

In non-scientific parlance, the word random is used to express lack of order, purpose, cause, or predictability.

Ramsey theory shows there is no such thing as "true randomness".

Algorithmic randomness, probably the best model of 'randomness', was intensively studied in the last 40 years by algorithmic information theory.

Quantum randomness is the only 'real' source of genuine randomness.

We will present new results and open questions related to the following fundamental problems:

We will present new results and open questions related to the following fundamental problems:

Is quantum randomness algorithmic random?

We will present new results and open questions related to the following fundamental problems:

Is quantum randomness algorithmic random?

How powerful is a Turing machine supplied with a finite, but unbounded source of quantum random bits?

1. Randomness
2. Algorithmic randomness
3. Quantum randomness
4. Is quantum randomness algorithmically random?

- Weird. "The system's been behaving pretty randomly".

- Weird. "The system's been behaving pretty randomly".
- Undistinguished. "Who was at the conference?" "Just a bunch of random business types".

- Weird. "The system's been behaving pretty randomly".
- Undistinguished. "Who was at the conference?" "Just a bunch of random business types".
- Unproductive. "He's just a random loser".

- Weird. "The system's been behaving pretty randomly".
- Undistinguished. "Who was at the conference?" "Just a bunch of random business types".
- Unproductive. "He's just a random loser".
- Incoherent. "That's a random name for that function".

- Weird. "The system's been behaving pretty randomly".
- Undistinguished. "Who was at the conference?" "Just a bunch of random business types".
- Unproductive. "He's just a random loser".
- Incoherent. "That's a random name for that function".
- In no particular order. "The I/O channels are in a pool, and when a file is opened one is chosen randomly".

- Weird. "The system's been behaving pretty randomly".
- Undistinguished. "Who was at the conference?" "Just a bunch of random business types".
- Unproductive. "He's just a random loser".
- Incoherent. "That's a random name for that function".
- In no particular order. "The I/O channels are in a pool, and when a file is opened one is chosen randomly".
- Arbitrary. "It generates a random name for the scratch file".

- Weird. "The system's been behaving pretty randomly".
- Undistinguished. "Who was at the conference?" "Just a bunch of random business types".
- Unproductive. "He's just a random loser".
- Incoherent. "That's a random name for that function".
- In no particular order. "The I/O channels are in a pool, and when a file is opened one is chosen randomly".
- Arbitrary. "It generates a random name for the scratch file".
- A random hacker. Teenager soaking up computer time and generally getting in the way.

- Weird. "The system's been behaving pretty randomly".
- Undistinguished. "Who was at the conference?" "Just a bunch of random business types".
- Unproductive. "He's just a random loser".
- Incoherent. "That's a random name for that function".
- In no particular order. "The I/O channels are in a pool, and when a file is opened one is chosen randomly".
- Arbitrary. "It generates a random name for the scratch file".
- A random hacker. Teenager soaking up computer time and generally getting in the way.
- (MIT usage) One who lives at Random Hall.

- Weird. "The system's been behaving pretty randomly".
- Undistinguished. "Who was at the conference?" "Just a bunch of random business types".
- Unproductive. "He's just a random loser".
- Incoherent. "That's a random name for that function".
- In no particular order. "The I/O channels are in a pool, and when a file is opened one is chosen randomly".
- Arbitrary. "It generates a random name for the scratch file".
- A random hacker. Teenager soaking up computer time and generally getting in the way.
- (MIT usage) One who lives at Random Hall.
- ...

The entire "philosophy" of AIT is to equate "randomness" with incompressibility.

The entire "philosophy" of AIT is to equate "randomness" with incompressibility.

A bit-string is (algorithmic) c–random if a universal Turing machine cannot compress it by more than c bits. The constant c should be less than the logarithm of the length of the string.

The entire "philosophy" of AIT is to equate "randomness" with incompressibility.

A bit-string is (algorithmic) c–random if a universal Turing machine cannot compress it by more than c bits. The constant c should be less than the logarithm of the length of the string.

Convention: we will drop the adjective "algorithmic".

Suppose someone claims to have tossed a fair coin 64 times and the result is:

$$0101010101010101010101010101010101$$
$$0101010101010101010101010101010101$$

Suppose someone claims to have tossed a fair coin 64 times and the result is:

$$0101010101010101010101010101010101$$
$$0101010101010101010101010101010101$$

Then, the experiment is repeated and the result is:

$$1010101010010010101001011011000 1$$
$$1000110011110011001111100001 1011$$

Suppose someone claims to have tossed a fair coin 64 times and the result is:

$$0101010101010101010101010101010101$$
$$0101010101010101010101010101010101$$

Then, the experiment is repeated and the result is:

$$1010101010010010101001011011000 1$$
$$1000110011110011001111100001 1011$$

We would be surprised/suspicious about the first result, but less/not about the second one.

Suppose someone claims to have tossed a fair coin 64 times and the result is:

$$0101010101010101010101010101010101$$
$$0101010101010101010101010101010101$$

Then, the experiment is repeated and the result is:

$$1010101010010010101001011011000 1$$
$$1000110011110011001111100001 1011$$

We would be surprised/suspicious about the first result, but less/not about the second one. Why?

Suppose someone claims to have tossed a fair coin 64 times and the result is:

$$0101010101010101010101010101010101$$
$$0101010101010101010101010101010101$$

Then, the experiment is repeated and the result is:

$$1010101010010010101001011011000 1$$
$$1000110011110011001111100001 1011$$

We would be surprised/suspicious about the first result, but less/not about the second one. Why?

A flawed explanation: the probability of obtaining the first result is 2^{-64}, so it is unreasonable to believe that it has been produced by a real experiment!

The difference between the above results is not probabilistic, but structural: while the first string is ordered, there is no apparent pattern in the second.

Finite random bit-strings have the following expected properties:

- Abundance: almost every bit-string is random (density one).

Finite random bit-strings have the following expected properties:

- Abundance: almost every bit-string is random (density one).
- Unpredictability: no Turing machine can compute the ith bit of a random string based on its first $i - 1$ bits.

Finite random bit-strings have the following expected properties:

- Abundance: almost every bit-string is random (density one).
- Unpredictability: no Turing machine can compute the ith bit of a random string based on its first $i-1$ bits.
- Statistically unbiased: they satisfy all computable enumerable statistical properties of randomness.

Finite random bit-strings have the following expected properties:

- Abundance: almost every bit-string is random (density one).
- Unpredictability: no Turing machine can compute the ith bit of a random string based on its first $i-1$ bits.
- Statistically unbiased: they satisfy all computable enumerable statistical properties of randomness.
- Uncomputability: no Turing machine can check whether a string is random or not.

Finite random bit-strings have the following expected properties:

- Abundance: almost every bit-string is random (density one).
- Unpredictability: no Turing machine can compute the ith bit of a random string based on its first $i-1$ bits.
- Statistically unbiased: they satisfy all computable enumerable statistical properties of randomness.
- Uncomputability: no Turing machine can check whether a string is random or not.
- Strong uncomputability: no Turing machine can generate infinitely many random strings.

Finite random bit-strings have the following expected properties:

- Abundance: almost every bit-string is random (density one).
- Unpredictability: no Turing machine can compute the ith bit of a random string based on its first $i-1$ bits.
- Statistically unbiased: they satisfy all computable enumerable statistical properties of randomness.
- Uncomputability: no Turing machine can check whether a string is random or not.
- Strong uncomputability: no Turing machine can generate infinitely many random strings.

Finite random bit-strings have the following unexpected properties:

- Halting times are not random: no Turing machine that stops on an input of n bits after a time larger than 2^{2n+3} stops at a random time.

Finite random bit-strings have the following unexpected properties:

- Halting times are not random: no Turing machine that stops on an input of n bits after a time larger than 2^{2n+3} stops at a random time.
- Every "decent" Monte Carlo simulation algorithm (like Rabin's primality test) powered with algorithmic randomness produces the correct result not only with high probability, but rigourously correct.

A sequence $x_1 x_2 \cdots$ is random if there is a $c > 0$ such that each of its prefixes is c–random.

A sequence $x_1 x_2 \cdots$ is random if there is a $c > 0$ such that each of its prefixes is c–random.

Random sequences

- satisfy all computable enumerable statistical properties of randomness,

A sequence $x_1 x_2 \cdots$ is random if there is a $c > 0$ such that each of its prefixes is c–random.

Random sequences

- satisfy all computable enumerable statistical properties of randomness,
- are unpredictable,

A sequence $x_1 x_2 \cdots$ is random if there is a $c > 0$ such that each of its prefixes is c–random.

Random sequences

- satisfy all computable enumerable statistical properties of randomness,
- are unpredictable,
- are uncomputable,

A sequence $x_1 x_2 \cdots$ is random if there is a $c > 0$ such that each of its prefixes is c–random.

Random sequences

- satisfy all computable enumerable statistical properties of randomness,
- are unpredictable,
- are uncomputable,
- abound (have measure one).

Example of an infinite random sequences

The infinite binary expansion of the halting probability of a prefix-free universal Turing machine U, Chaitin's Omega number,

$$\Omega_U = \sum_{U(x) \text{ stops}} 2^{-|x|}$$

is random.

Example of an infinite random sequences

The infinite binary expansion of the halting probability of a prefix-free universal Turing machine U, Chaitin's Omega number,

$$\Omega_U = \sum_{U(x) \text{ stops}} 2^{-|x|}$$

is random.

With the first n bits of Ω_U one can solve the halting problem for U and all programs of less than n bits. ▸ Examples

Quantum randomness

Quantum randomness

- It is postulated in the standard model of quantum mechanics.

Quantum randomness

- It is postulated in the standard model of quantum mechanics.
- It has been confirmed by theoretical and experimental research.

Quantum randomness

- It is postulated in the standard model of quantum mechanics.
- It has been confirmed by theoretical and experimental research.
- It passes all reasonable statistical properties of randomness.

Quantum randomness

- It is postulated in the standard model of quantum mechanics.
- It has been confirmed by theoretical and experimental research.
- It passes all reasonable statistical properties of randomness.
- Can be easily and reliably produced: A photon generated by a source beamed to a semitransparent mirror is reflected or transmitted with 50 per cent chance, and these measurements can be translated into a string of quantum random bits. ▸ Quantis

Quantum randomness is not computable

Quantum randomness is not computable

- Computer generated randomness is computable (John von Neumann: "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin".).

Quantum randomness is not computable

- Computer generated randomness is computable (John von Neumann: "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin".).
- Quantum generated randomness is uncomputable.

Quantum randomness is not computable

- Computer generated randomness is computable (John von Neumann: "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin".).
- Quantum generated randomness is uncomputable.
- Uncomputability is a strong property, but it does not necessarily imply algorithmical incompressibility.

Quantum randomness is not computable

- Computer generated randomness is computable (John von Neumann: "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin".).
- Quantum generated randomness is uncomputable.
- Uncomputability is a strong property, but it does not necessarily imply algorithmical incompressibility.

Is quantum randomness algorithmically random?

How 'random' quantum randomness?

Can finite tests distinguish between computable and uncomputable randomness?

Can finite tests distinguish between computable and uncomputable randomness?

Various tests

- standard: DieHarder, TESTU01 Crush and BigCrush tests,
- AIT–based

have been done with six 2^{32}-bit strings produced by Quantis (2), Mathematica (2), C and π.

How 'random' quantum randomness?

Can finite tests distinguish between computable and uncomputable randomness?

Various tests

- standard: DieHarder, TESTU01 Crush and BigCrush tests,
- AIT–based

have been done with six 2^{32}-bit strings produced by Quantis (2), Mathematica (2), C and π.

To date Quantis and Mathematica generated randomness are very close. More testing is in progress.

A PC powered with quantum random bits

Plug quantum random bits into a PC and we can, in theory at least, leapfrog Turing's barrier.

A PC powered with quantum random bits

Plug quantum random bits into a PC and we can, in theory at least, leapfrog Turing's barrier.

Is this interesting?

A PC powered with quantum random bits

Plug quantum random bits into a PC and we can, in theory at least, leapfrog Turing's barrier.

Is this interesting?

How much more powerful a Turing machine working with "an oracle of quantum random bits" can be?

A PC powered with quantum random bits

Plug quantum random bits into a PC and we can, in theory at least, leapfrog Turing's barrier.

Is this interesting?

How much more powerful a Turing machine working with "an oracle of quantum random bits" can be?

This machine could, at any time, ask the "oracle of quantum random bits" to supply an arbitrarily long (but finite) string. It won't have access to an infinite sequence, but to an unbounded finite set of strings of quantum random bits.

References

- C. S. Calude. Algorithmic randomness, quantum physics, and incompleteness, in M. Margenstern (ed.). *Proc. Conf. "Machines, Computations and Universality"*, Lectures Notes in Comput. Sci. 3354, Springer, Berlin, 2005, 1–17.
- C. S. Calude, J. Casti. The jumble cruncher, *New Scientist*, 25 September 2004, 36–37.
- C. S. Calude, H. Jürgensen. Is complexity a source of incompleteness? *Advances in Applied Mathematics* 35 (2005), 1–15.
- C. S. Calude, M. A. Stay. Most programs stop quickly or never halt, *Adv. Appl. Math.*, (2007), doi:10.1016/j.amm.2007.01.001.
- C. S. Calude, K. Svozil. Quantum randomness and value indefiniteness, *Advanced Science Letters*, accepted.
- K. Svozil. The quantum coin toss-testing microphysical undecidability, *Phys. Lett. A* 143 (1990), 433–437.

- With 3,484 bits one can solve Goldbach's Conjecture.
- With 7,780 bits one can solve the Riemann Hypothesis.

▸ Quantum randomness

Quantis

Quantis: 16Mbits/sec quantum mechanical random number generator produced and sold by *id Quantique* of the University of Geneva

▸ Quantum randomness