# Five Questions

Michael A. Stay
Google and The University of Auckland
Email: stay@google.com

1. *Why were you initially drawn to the study of computation and randomness?*

   I've always been torn between studying physics and studying computer science, but it was the theoretical aspects of both that attracted me the most. There's clearly overlap in the computational physics arena, but I found that coming up with good numerical approximations to systems wasn't really to my taste. It was only when I began studying information theory that I came to understand that there was also an enormous overlap between the two theoretical sides of the fields. And like many others, I was intrigued with the possibility that somehow algorithmic randomness lies at the root of quantum randomness.

2. *What we have learned?*

   One of the strange things about quantum physics is that we can describe matter, which to our senses often feels hard and unyielding, by waves! Most people get their intuition about waves from music. When a bassist plucks his strings, it's hard to place the pitch precisely, because waves have a particularly interesting property: you can't tell both the time a note was played and its pitch with an accuracy greater than about a quarter cycle. The low range of a bass is only tens of vibrations per second, so you only hear a couple of complete cycles during a pluck. On the other hand, if the bassist uses his bow and plays the note long enough to get a good idea of the pitch, there's no unique instant during which the note was being played. Compare this to a violin's high range: its strings vibrate hundreds of times per second, so it's far easier to pick out a note (and far easier to cringe when it's not played precisely on pitch!)

   In quantum physics, we deal with matter waves, and the heavier the object, the higher the frequency. Our intuition about being able to measure the position and momentum of a particle at the same time is because we've spent our life listening to tightly strung violins and piccolos! When we get down to the size of molecules, which are more similar to the bass, we have to start considering their wave nature to describe them properly.

   This inability to measure both the pitch and time of a note, or the position and momentum of a particle, is called Heisenberg's uncertainty principle. Until that time, people had believed that it was possible, in principle, to predict the end of the world: simply measure the position and momentum of every particle and compute it! But Heisenberg showed that it was fundamentally impossible. Mathematically, we have

   $$\Delta x \cdot \Delta p \geq \frac{\hbar}{2},$$

   where $\Delta x$ means "the uncertainty in $x$," $x$ is the position, $p$ is the momentum, and $\hbar$ is Planck's constant. If we try to measure the position of a particle very carefully, making $\Delta x$

very small, we have to make $\Delta p$ large to satisfy the inequality and we lose all the information about its momentum. Similarly, in music, the click of a metronome occurs at a very precise time, but you can't assign a pitch to it.

Another analogy between music and quantum physics also holds: just as you can play notes simultaneously—or "superpose" them—to form a chord, you can superpose possibilities. When a photon hits a partially-reflective surface, like this page, the universe becomes a chord of two possibilities: one note has the photon being reflected, while another note has the photon being absorbed. But unlike sound waves, when we listen to the chord of the universe—that is, when we perform a measurement—we hear only a single note, and which note it is appears to be random! This has suggested to many people that we should be looking at algorithmic randomness for an answer to this mystery.

Chaitin [3] showed that there's a kind of number called an *Omega number* whose digits are "pure information:" to compute the first $n$ bits of an Omega number, you need $n - c$ bits of program, where $c$ is a number that only depends on your choice of programming language. One way to think about information is as a measure of surprise: if you can predict what's next—that is, you have a program that can compute it for you—then you're not surprised. For example, the first 2000 bits of the binary fraction for 1/3 can be produced with this short Python program:

```
print "01" * 1000;
```

But if the number is completely random, like an Omega number; if every bit is surprising and unpredictable, then the shortest program to print the number doesn't do any computation at all! It just looks something like

```
print "0100100011110100110010100101001010001011101101100110100 1010";
```

and if you want a lot of bits, your program will be very long.

When I saw this for the first time, it immediately struck me that this was the logarithm of an uncertainty principle! Here's how: first, pick a universal Turing machine $U$. Chaitin showed that if you pick a random program, the probability that it will actually run to completion (as opposed to going into an infinite loop or crashing) is one of these Omega numbers:

$$\Omega_U = \sum_{U(p) \text{ halts}} 2^{-|p|}.$$

Finally, suppose that there are two strings $s$ and $x$ such that together they form a program that prints out the first $n$ bits of $\Omega_U$—but also suppose that you don't know $s$.

What is the uncertainty $\Delta\Omega_U$? Well, $\Omega_U$ is a probability, so it's got to be between 0 and 1; this means it's a number that starts with "zero-dot," like 0.5 or 0.141592653... We also have the first $n$ bits after the dot, so the uncertainty must be $2^{-n}$.

What is the uncertainty $\Delta x$? There's no canonical way to give the distance between two strings, but there's a nice isomorphism between binary strings and natural numbers: stick a 1 on the front! Given this map, we can ask what the uncertainty is in the natural number $1x$. Well, if $s$ is already the shortest program outputting the first $n$ bits of $\Omega_U$, then we don't need to add anything else; that means $x$ can be the empty string, and $1x = 1$. On the other hand, $s$ might be empty, in which case $x$ forms the whole program. Since $\Omega_U$ is one of those special numbers Chaitin described, $x$ needs to be at least $n-c$ bits long; therefore $1x \geq 2^{n-c}$. The difference is the uncertainty: $\Delta x \geq 2^{n-c} - 1$.

The product of these two is an uncertainty principle formally equivalent to Heisenberg's uncertainty principle:

$$\Delta x \cdot \Delta \Omega_U \geq k,$$

where $k$ is a real number depending only on the choice of programming language. Starting from this principle, Cristian Calude and I were able to show [4] that it implies Chaitin's information-theoretic incompleteness, and we also designed a quantum computer for which the halting probability and the program were complementary observables, just like position and momentum. Now, the quantum computer wasn't a universal one, and we didn't show that quantum randomness has roots in algorithmic randomness. But there's something going on there that deserves some attention.

There are other connections between physics and algorithmic information theory. Szilard showed in 1929 [7] that information and energy can be exchanged; Landauer [5] later showed that *temperature* is the exchange rate between the currencies of energy and information: the energy required to erase $B$ arbitrary bits is

$$E \geq B \ln(2) kT.$$

Bennett, Gács, Li, Vitányi, and Zurek produced that wonderful paper on information distance [2] that refined Landauer's principle and connected it with computation: the energy required to turn a string $x$ into a string $y$ is

$$E \geq \mathrm{KR}(x|y) \ln(2) kT,$$

where $\mathrm{KR}(x|y)$ is the length in bits of the shortest reversible program that given $x$ produces $y$ (and since it's reversible, is also effectively a program for producing $x$ given $y$).

Statistical mechanics is the science of predicting what happens when you have a bazillion copies of a tiny system, like a balloon with a bazillion molecules inside. The *partition function* says that if your balloon is in equilibrium at temperature $T$, then the relative probability that any particular molecule has energy $E$ is $\exp(-E/kT)$—that is, as you increase the energy you're looking for, the probability you'll find a particle with that energy decreases exponentially.

Tadaki [8] generalized the halting probability by adding a weighting factor to the program length:

$$\Omega_U(s) = \sum_{U(p) \text{ halts}} 2^{-s|p|}.$$

I pointed out that this has the same form as the partition function in statistical mechanics.

We can think of a universal machine as being a balloon with a bazillion programs inside. By letting the "energy" of a program be its length in bits times $\ln(2)$, Tadaki's formula above turns into the partition function:

$$\Omega_U(1/kT) = \sum_{U(p) \text{ halts}} \exp(-\ln(2)|p| \;/\; kT).$$

Again, the temperature is playing the role of an exchange rate; if $T$ is computable, then the resulting real in this summation is *partially random*, and the temperature is a measure of how well its bits can be compressed. So it's clear there's a deep connection between algorithmic information theory and physics.

3. *What we don't know (yet)?*

   The category of data types with $\alpha$-$\beta$-$\eta$−equivalence classes of linear lambda terms between them is a symmetric monoidal closed category [1]; the same is true of the category of particle worldlines with Feynman diagrams between them and the category of Hilbert spaces and linear transformations. So programs are, in a precise way, analogous to Feynman diagrams, and a sum over programs is related to a path integral. There is something very deep to this analogy, and no one has plumbed it yet.

   The zeta function of a fractal string encodes information about its dimension in its zeros [6]; the generalized halting probability is the zeta function of "the halting fractal." This brings up the question of the value of the generalized halting probability at complex inputs. What information is encoded in its zeros? Can the zeta function be analytically continued around the pole at 1? If so, how compressible is the analytical continuation of the zeta function at computable values in $\mathbb{C}$?

4. *What are the most important open problems?*

   I don't know much about the *importance* of particular problems, but I think that the most *interesting* open problems for me are the ones dealing with what kind of algorithmic information is *actually accessible* to us: are there any finite physical systems that provably produce random bits? What computations are actually occurring when particles interact? Can we exploit them to solve problems that Turing machines can't?

5. *What are the prospects for progress?*

   Very good—we're just beginning to explore, and there's a lot of low-hanging fruit waiting to be picked and appreciated.

## References

[1] John Baez and Michael Stay, "Physics, Topology, Logic, and Computation: A Rosetta Stone." *New Structures in Physics*, a volume in *Lecture Notes in Physics*, to appear.

[2] Charles H. Bennett, Peter Gács, Ming Li, Paul M. B. Vitányi, and Woiciech Zurek, "Thermodynamics of computation and information distance," *IEEE Transactions on Information Theory* 44 (1998), no. 4, 14071423, Preliminary version appeared in the 1993 ACM Symp. on Th. of Comp.

[3] G.J. Chaitin. "A theory of program size formally identical to information theory," *J. Assoc. Comput. Mach.* 22 (1975), 329340. (Received April 1974)

[4] C.S. Calude, M.A. Stay, "From Heisenberg to Gödel via Chaitin." *International Journal of Theoretical Physics*, 44 (7). 1053–1065.

[5] Rolf Landauer, "Irreversibility and heat generation in the computing process," *IBM Journal of Research and Development*, vol. 5, pp. 183-191, 1961.

[6] Erin Pearse, *Complex dimensions of self-similar systems*, Ph.D. thesis.

[7] Szilard, L. (1929). *Zeitschrift für Physik*, 53, 840-856 [Translated and reprinted in Quantum Theory and Measurement, J. A. Wheeler and W. H. Zurek, eds., Princeton University Press, Princeton, New Jersey (1983)].

[8] Kohtaro Tadaki, "A Generalization of Chaitin's Halting Probability $\Omega$ and Halting Self-Similar Sets," Hokkaido Mathematical Journal, Vol. 31, No. 1, February 2002, 219-253

[9] W.K. Wootters and W.H. Zurek, "A Single Quantum Cannot be Cloned", *Nature* 299 (1982), pp. 802-803.