Metaphysics, Metamathematics and Metabiology

Gregory Chaitin[*]

Abstract

In this essay we present an information-theoretic perspective on epistemology using software models. We shall use the notion of algorithmic information to discuss what is a physical law, to determine the limits of the axiomatic method, and to analyze Darwin's theory of evolution.

Weyl, Leibniz, complexity and the principle of sufficient reason

The best way to understand the deep concept of conceptual complexity and algorithmic information, which is our basic tool, is to see how it evolved, to know its long history. Let's start with Hermann Weyl and the great philosopher/mathematician G. W. Leibniz. That everything that is true is true for a reason is rationalist Leibniz's famous *principle of sufficient reason.* The bits of Ω seem to refute this fundamental principle and also the idea that everything can be proved starting from self-evident facts.

What is a scientific theory?

The starting point of algorithmic information theory, which is the subject of this essay, is this toy model of the scientific method:

theory/program/010 $\rightarrow$ **Computer** $\rightarrow$ experimental data/output/110100101.

A scientific theory is a computer program for producing exactly the experimental data, and both theory and data are a finite sequence of bits, a bit string. Then we can define the complexity of a theory to be its size in bits, and we can compare the size in bits of a theory with the size in bits of the experimental data that it accounts for.

That the simplest theory is best, means that we should pick the smallest program that explains a given set of data. Furthermore, if the theory is the same size as the data, then it is useless, because there is always a theory that is

[*]Chaitin is emeritus researcher at the IBM Watson Research Center, an honorary professor at the University of Buenos Aires, and a member of the Académie Internationale de Philosophie des Sciences. His email address is `gjchaitin@gmail.com`.

the same size as the data that it explains. In other words, a theory must be a compression of the data, and the greater the compression, the better the theory. Explanations are compressions, comprehension is compression!

Furthermore, if a bit string has absolutely no structure, if it is completely random, then there will be no theory for it that is smaller than it is. Most bit strings of a given size are incompressible and therefore incomprehensible, simply because there are not enough smaller theories to go around.

This software model of science is not new. It can be traced back via Hermann Weyl (1932) to G. W. Leibniz (1686)! Let's start with Weyl. In his little book on philosophy *The Open World: Three Lectures on the Metaphysical Implications of Science*, Weyl points out that if arbitrarily complex laws are allowed, then the concept of law becomes vacuous, because there is always a law! In his view, this implies that the concept of a physical law and of complexity are inseparable; for there can be no concept of law without a corresponding complexity concept. Unfortunately he also points out that in spite of its importance, the concept of complexity is a slippery one and hard to define mathematically in a convincing and rigorous fashion.

Furthermore, Weyl attributes these ideas to Leibniz, to the 1686 *Discours de métaphysique.* What does Leibniz have to say about complexity in his *Discours*? The material on complexity is in Sections V and VI of the *Discours.*

In Section V, Leibniz explains why science is possible, why the world is comprehensible, lawful. It is, he says, because God has created the best possible, the most perfect world, in that the greatest possible diversity of phenomena are governed by the smallest possible set of ideas. God simultaneously maximizes the richness and diversity of the world and minimizes the complexity of the ideas, of the mathematical laws, that determine this world. That is why science is possible!

A modern restatement of this idea is that science is possible because the world seems very complex but is actually governed by a small set of laws having low conceptual complexity.

And in Section VI of the *Discours*, Leibniz touches on randomness. He points out that any finite set of points on a piece of graph paper always seems to follow a law, because there is always a mathematical equation passing through those very points. But there is a law only if the equation is simple, not if it is very complicated. This is the idea that impressed Weyl, and it becomes the definition of randomness in algorithmic information theory.[1]

Finding elegant programs

So the best theory for something is the smallest program that calculates it. How can we be sure that we have the best theory? Let's forget about theories and just call a program *elegant* if it is the smallest program that produces the output

[1] *Historical Note:* Algorithmic information theory was first proposed in the 1960s by R. Solomonoff, A. N. Kolmogorov, and G. J. Chaitin. Solomonoff and Chaitin considered this toy model of the scientific method, and Kolmogorov and Chaitin proposed defining randomness as algorithmic incompressibility.

that it does. More precisely, a program is elegant if no smaller program written in the same language produces the same output.

So can we be sure that a program is elegant, that it is the best theory for its output? Amazingly enough, we can't: It turns out that any formal axiomatic theory A can prove that at most finitely many programs are elegant, in spite of the fact that there are infinitely many elegant programs. More precisely, it takes an N-bit theory A, one having N bits of axioms, having complexity N, to be able to prove that an individual N-bit program is elegant. And we don't need to know much about the formal axiomatic theory A in order to be able to prove that it has this limitation.

What is a formal axiomatic theory?

All we need to know about the axiomatic theory A, is the crucial requirement emphasized by David Hilbert that there should be a proof-checking algorithm, a mechanical procedure for deciding if a proof is correct or not. It follows that we can systematically run through all possible proofs, all possible strings of characters in the alphabet of the theory A, in size order, checking which ones are valid proofs, and thus discover all the theorems, all the provable assertions in the theory A.[2]

That's all we need to know about a formal axiomatic theory A, that there is an algorithm for generating all the theorems of the theory. This is the software model of the axiomatic method studied in algorithmic information theory. If the software for producing all the theorems is N bits in size, then the complexity of our theory A is defined to be N bits, and we can limit A's power in terms of its complexity $H(A) = N$. Here's how:

Why can't you prove that a program is elegant?

Suppose that we have an N-bit theory A, that is, that $H(A) = N$, and that it is always possible to prove that individual elegant programs are in fact elegant, and that it is never possible to prove that inelegant programs are elegant. Consider the following paradoxical program P:

> *P runs through all possible proofs in the formal axiomatic theory A, searching for the first proof in A that an individual program Q is elegant for which it is also the case that the size of Q in bits is larger than the size of P in bits. And what does P do when it finds Q? It runs Q and then P produces as its output the output of Q.*

In other words, the output of P is the same as the output of the first provably elegant program Q that is larger than P. But this contradicts the definition of elegance! P is too small to be able to calculate the output of an elegant program Q that is larger than P. We seem to have arrived at a contradiction!

[2] *Historical Note:* The idea of running through all possible proofs, of creativity by mechanically trying all possible combinations, can be traced back through Leibniz to Ramon Llull in the 1200s.

But do not worry; there is no contradiction. What we have actually proved is that P can never find Q. In other words, there is no proof in the formal axiomatic theory A that an individual program Q is elegant, not if Q is larger than P. And how large is P? Well, just a fixed number of bits c larger than N, the complexity $H(A)$ of the formal axiomatic theory A. P consists of a small, fixed main program c bits in size, followed by a large subroutine $H(A)$ bits in size for generating all the theorems of A.

The only tricky thing about this proof is that it requires P to be able to know its own size in bits. And how well we are able to do this depends on the details of the particular programming language that we are using for the proof. So to get a neat result and to be able to carry out this simple, elegant proof, we have to be sure to use an appropriate programming language. This is one of the key issues in algorithmic information theory, which programming language to use.[3]

Farewell to reason: The halting probability Ω[4]

So there are infinitely many elegant programs, but there are only finitely many provably elegant programs in any formal axiomatic theory A. The proof of this is rather straightforward and short. Nevertheless, this is a fundamental information-theoretic incompleteness theorem that is rather different in style from the classical incompleteness results of Gödel, Turing and others.

An even more important incompleteness result in algorithmic information theory has to do with the halting probability Ω, the numerical value of the probability that a program p whose successive bits are generated by independent tosses of a fair coin will eventually halt:

$$\Omega = \sum_{p \text{ halts}} 2^{-(\text{size in bits of } p)}.$$

To be able to define this probability Ω, it is also very important how you chose your programming language. If you are not careful, this sum will diverge instead of being ≤ 1 like a well-behaved probability should.

Turing's fundamental result is that the halting problem is unsolvable. In algorithmic information theory the fundamental result is that the halting probability Ω is algorithmically irreducible or random. It follows that the bits of Ω cannot be compressed into a theory less complicated than they are. They are irreducibly complex. It takes N bits of axioms to be able to determine N bits of the numerical value

$$\Omega = .1101011\ldots$$

of the halting probability. If your formal axiomatic theory A has $H(A) = N$, then you can determine the values and positions of at most $N + c$ bits of Ω.

[3]See the chapter on "The Search for the Perfect Language" in Chaitin, *Mathematics, Complexity and Philosophy*, in press.

[4]*Farewell to Reason* is the title of a book by Paul Feyerabend, a wonderfully provocative philosopher. We borrow his title here for dramatic effect, but he does not discuss Ω in this book or any of his other works.

In other words, the bits of Ω are logically irreducible, they cannot be proved from anything simpler than they are. Essentially the only way to determine what are the bits of Ω is to add these bits to your theory A as new axioms. But you can prove anything by adding it as a new axiom. That's not using reasoning!

So the bits of Ω refute Leibniz's principle of sufficient reason: they are true for no reason. More precisely, they are not true for any reason simpler than themselves. This is a place where mathematical truth has absolutely no structure, no pattern, for which there is no theory!

Adding new axioms: Quasi-empirical mathematics[5]

So incompleteness follows immediately from fundamental information-theoretic limitations. What to do about incompleteness? Well, just add new axioms, increase the complexity $H(A)$ of your theory A! That is the only way to get around incompleteness.

In other words, do mathematics more like physics, add new axioms not because they are self-evident, but for pragmatic reasons, because they help mathematicians to organize their mathematical experience just like physical theories help physicists to organize their physical experience. After all, Maxwell's equations and the Schrödinger equation are not at all self-evident, but they work! And this is just what mathematicians have done in theoretical computer science with the hypothesis that $P \neq NP$, in mathematical cryptography with the hypothesis that factoring is hard, and in abstract axiomatic set theory with the new axiom of projective determinacy.[6]

Mathematics, biology and metabiology

We've discussed physical and mathematical theories; now let's turn to biology, the most exciting field of science at this time, but one where mathematics is not very helpful. Biology is very different from physics. There is no simple equation for your spouse. Biology is the domain of the complex. There are not many universal rules. There are always exceptions. Math is very important in theoretical physics, but there is no fundamental mathematical theoretical biology.

This is unacceptable. The honor of mathematics requires us to come up with a mathematical theory of evolution and either prove that Darwin was wrong or right! We want a general, abstract theory of evolution, not an immensely

[5]The term *quasi-empirical* is due to the philosopher Imre Lakatos, a friend of Feyerabend. For more on this school, including the original article by Lakatos, see the collection of quasi-empirical philosophy of math papers edited by Thomas Tymoczko, *New Directions in the Philosophy of Mathematics.*

[6]See the article on "The Brave New World of Bodacious Assumptions in Cryptography" in the March 2010 issue of the *AMS Notices*, and the article by W. Hugh Woodin on "The Continuum Hypothesis" in the June/July 2001 issue of the *AMS Notices.*

complicated theory of actual biological evolution. And we want proofs, not computer simulations! So we've got to keep our model very, very simple.

That's why this proposed new field is *metabiology*, not biology.

What kind of math can we use to build such a theory? Well, it's certainly not going to be differential equations. Don't expect to find the secret of life in a differential equation; that's the wrong kind of mathematics for a fundamental theory of biology.

In fact a universal Turing machine has much more to do with biology than a differential equation does. A universal Turing machine is a very complicated new kind of object compared to what came previously, compared with the simple, elegant ideas in classical mathematics like analysis. And there are self-reproducing computer programs, which is an encouraging sign.

There are in fact three areas in our current mathematics that do have some fundamental connection with biology, that show promise for math to continue moving in a biological direction:

Computation, Information, Complexity.

DNA is essentially a programming language that computes the organism and its functioning; hence the relevance of the theory of computation for biology.

Furthermore, DNA contains biological information. Hence the relevance of information theory. There are in fact at least four different theories of information:

- Boltzmann statistical mechanics and Boltzmann entropy,
- Shannon communication theory and coding theory,
- algorithmic information theory (Solomonoff, Kolmogorov, Chaitin), which is the subject of this essay, and
- quantum information theory and qubits.

Of the four, AIT (algorithmic information theory) is closest in spirit to biology. AIT studies the size in bits of the smallest program to compute something. And the complexity of a living organism can be roughly (very roughly) measured by the number of bases in its DNA, in the biological computer program for calculating it.

Finally, let's talk about complexity. Complexity is in fact the most distinguishing feature of biological as opposed to physical science and mathematics. There are many computational definitions of complexity, usually concerned with computation times, but again AIT, which concentrates on program size or conceptual complexity, is closest in spirit to biology.

Let's emphasize what we are not interested in doing. We are certainly not trying to do systems biology: large, complex realistic simulations of biological systems. And we are not interested in anything that is at all like Fisher-Wright population genetics that uses differential equations to study the shift of gene frequencies in response to selective pressures.

We want to use a sufficiently rich mathematical space to model the space of all possible designs for biological organisms, to model biological creativity. And the only space that is sufficiently rich to do that is a software space, the space of all possible algorithms in a fixed programming language. Otherwise we have limited ourselves to a fixed set of possible genes as in population genetics, and it is hopeless to expect to model the major transitions in biological evolution such as from single-celled to multicellular organisms, which is a bit like taking a main program and making it into a subroutine that is called many times.

Recall the cover of Stephen Gould's *Wonderful Life* on the Burgess shale and the Cambrian explosion? Around 250 primitive organisms with wildly differing body plans, looking very much like the combinatorial exploration of a software space. Note that there are no intermediate forms; small changes in software produce vast changes in output.

So to simplify matters and concentrate on the essentials, let's throw away the organism and just keep the DNA. Here is our proposal:

> *Metabiology: a field parallel to biology that studies the random evolution of artificial software (computer programs) rather than natural software (DNA), and that is sufficiently simple to permit rigorous proofs or at least heuristic arguments as convincing as those that are employed in theoretical physics.*

This analogy may seem a bit far-fetched. But recall that Darwin himself was inspired by the analogy between artificial selection by plant and animal breeders and natural section imposed by malthusian limitations.

Furthermore, there are many tantalizing analogies between DNA and large, old pieces of software. Remember *bricolage*, that Nature is a cobbler, a tinkerer? In fact, a human being is just a very large piece of software, one that is 3×10^9 bases $= 6 \times 10^9$ bits $\approx$ one gigabyte of software that has been patched and modified for more than a billion years: a tremendous mess, in fact, with bits and pieces of fish and amphibian design mixed in with that for a mammal.[7] For example, at one point in gestation the human embryo has gills. As time goes by, large human software projects also turn into a tremendous mess with many old bits and pieces.

The key point is that you can't start over, you've got to make do with what you have as best you can. If we could design a human being from scratch we could do a much better job. But we can't start over. Evolution only makes small changes, incremental patches, to adapt the existing code to new environments.

So how do we model this? Well, the key ideas are:

Evolution of mutating software,

and:

Random walks in software space.

[7]See Neil Shubin, *Your Inner Fish: A Journey into the 3.5-Billion-Year History of the Human Body.*

That's the general idea. And here are the specifics of our current model, which is quite tentative.

We take an organism, a single organism, and perform random mutations on it until we get a fitter organism. That replaces the original organism, and then we continue as before. The result is a random walk in software space with increasing fitness, a hill-climbing algorithm in fact.[8]

Finally, a key element in our proposed model is the definition of fitness. For evolution to work, it is important to keep our organisms from stagnating. It is important to give them something challenging to do.

The simplest possible challenge to force our organisms to evolve is what is called the Busy Beaver problem, which is the problem of providing concise names for extremely large integers. Each of our organisms produces a single positive integer. The larger the integer, the fitter the organism.[9]

The Busy Beaver function of N, BB(N), that is used in AIT is defined to be the largest positive integer that is produced by a program that is less than or equal to N bits in size. BB(N) grows faster than any computable function of N and is closely related to Turing's famous halting problem, because if BB(N) were computable, the halting problem would be solvable.[10]

Doing well on the Busy Beaver problem can utilize an unlimited amount of mathematical creativity. For example, we can start with addition, then invent multiplication, then exponentiation, then hyper-exponentials, and use this to concisely name large integers:

$$N + N \ \to\ N \times N \ \to\ N^N \ \to\ N^{N^N} \ \to\ \ldots$$

There are many possible choices for such an evolving software model: You can vary the computer programming language and therefore the software space, you can change the mutation model, and eventually you could also change the fitness measure. For a particular choice of language and probability distribution of mutations, and keeping the current fitness function, it is possible to show that in time of the order of 2^N the fitness will grow as BB(N), which grows faster than any computable function of N and shows that genuine creativity is taking place, for mechanically changing the organism can only yield fitness that grows as a computable function.[11]

[8]In order to avoid getting stuck on a local maximum, in order to keep evolution from stopping, we stipulate that there is a non-zero probability to go from any organism to any other organism, and $-\log_2$ of the probability of mutating from A to B defines an important concept, the *mutation distance*, which is measured in bits.

[9]*Alternative formulations:* The organism calculates a total function $f(n)$ of a single non-negative integer n and $f(n)$ is fitter than $g(n)$ if $f(n)/g(n) \to \infty$ as $n \to \infty$. Or the organism calculates a (constructive) Cantor ordinal number and the larger the ordinal, the fitter the organism.

[10]Consider BB$'$(N) defined to be the maximum run-time of any program that halts that is less than or equal to N bits in size.

[11]Note that to actually simulate our model an oracle for the halting problem would have to be employed to avoid organisms that have no fitness because they never calculate a positive integer. This also explains how the fitness can grow faster than any computable function. In our evolution model, implicit use is being made of an oracle for the halting problem, which answers questions whose answers cannot be computed by any algorithmic process.

So with random mutations and just a single organism we actually do get evolution, unbounded evolution, which was precisely the goal of metabiology!

This theorem may seem encouraging, but it actually has a serious problem. The times involved are so large that our search process is essentially *ergodic*, which means that we are doing an exhaustive search. Real evolution is not at all ergodic, since the space of all possible designs is much too immense for exhaustive search.

It turns out that with this same model there is actually a much quicker *ideal evolutionary pathway* that achieves fitness BB(N) in time of the order of N. This path is however unstable under random mutations, plus it is much too good: Each organism adds only a single bit to the preceding organism, and immediately achieves near optimal fitness for an organism of its size, which doesn't seem to at all reflect the haphazard, frozen-accident nature of what actually happens in biological evolution.[12]

So that is the current state of metabiology: a field with some promise, but not much actual content at the present time. The particular details of our current model are not too important. Some kind of mutating software model should work, should exhibit some kind of basic biological features. The challenge is to identify such a model, to characterize its behavior statistically,[13] and *to prove* that it does what is required.

References

[1] G. J. Chaitin, *Thinking about Gödel and Turing: Essays on Complexity, 1970–2007*, World Scientific (2007).

[2] G. J. Chaitin, *Mathematics, Complexity and Philosophy*, Midas, in press. (Draft at `http://www.cs.umaine.edu/~chaitin/midas.html`.)

[3] S. Gould, *Wonderful Life*, Norton (1989).

[4] N. Koblitz and A. Menezes, "The brave new world of bodacious assumptions in cryptography," *AMS Notices* **57**, 357–365 (2010).

[5] G. W. Leibniz, *Discours de métaphysique, suivi de Monadologie*, Gallimard (1995).

[6] N. Shubin, *Your Inner Fish*, Pantheon (2008).

[7] T. Tymoczko, *New Directions in the Philosophy of Mathematics*, Princeton University Press (1998).

[8] H. Weyl, *The Open World*, Yale University Press (1932).

[12] The Nth organism in this ideal evolutionary pathway is essentially just the first N bits of the numerical value of the halting probability Ω. Can you figure out how to compute BB(N) from this?

[13] For instance, will some kind of hierarchical structure emerge? Large human software projects are always written that way.

[9] W. H. Woodin, "The continuum hypothesis, Part I," *AMS Notices* **48**, 567–576 (2001).

Note added in proof: The mathematical structure of metabiology is starting to emerge. Please see my paper "To a mathematical theory of evolution and biological creativity" at `http://www.cs.auckland.ac.nz/CDMTCS//researchreports/391greg.pdf`.