

Interview with Gregory Chaitin*

(IBM Research Division, Yorktown)

I consider myself to be a computer programmer who does mathematics as a hobby, a computer programming professional and amateur mathematician. I do not consider myself a philosopher. I have been forced to face some philosophical issues because of the kind of mathematics I do. And unlike a professional philosopher, I do not have an official position on every philosophical issue. I like to play with ideas: “On the one hand... On the other hand...,” that sort of thing, some kind of an internal debate, never completely resolved.

Why was I initially drawn to computational and informational issues? As a child in the 1950s I read every issue of *Scientific American* from cover to cover, and kept a complete collection of these magazines in my room. Shannon’s information theory had made quite a splash, and computer technology was progressing by leaps and bounds. I studied some of the earliest books on computer programming and computer technology, absolutely fascinated by these “thinking machines,” these “giant electronic brains,” as they were called then.

I also loved the clarity and sharpness of mathematical thought, which is the basis for physics, and read a lot of popularizations: W. W. Sawyer, G. Polya, Einstein and Infeld, E. T. Bell, G. H. Hardy, Tobias Dantzig, Courant and Robbins, endless piles of books from the New York Public Library.

And then I heard rumors that something was seriously amiss, that a crack had developed in the dike of mathematical thought, that there was a mysterious result questioning everything: Gödel’s incompleteness theorem! It looked simple at first. It was all explained, very clearly, in a little book by Nagel and Newman called *Gödel’s Proof*, which I read and re-read.

*Prepared for Floridi, *The Philosophy of Computing & Information: 5 Questions*.

However, the heart of the argument was a bizarre self-reference, “I am unprovable!,” which made me very unhappy. It was as if you had asked the Delphic oracle for the secret of the universe, and it had responded, “43!” Perhaps correct, but not very enlightening. Could **this** be the mystery at the heart of understanding what mathematics was all about, what reasoning could or could not achieve?!

I **had** to understand this. The quest was intoxicating; I kept discovering fascinating new ideas.

Right away I discovered there was an approach to incompleteness that satisfied me far more than Gödel’s proof. Turing’s way of deducing incompleteness as a corollary of uncomputability seemed much more satisfying, much more solid and concrete. Computers and computer programs were quite tangible for me. I began learning the craft of computer programming as a teenager: machine language, assembly language, FORTRAN, recursive procedures, list processing, LISP, all that fun stuff, the carpentry of pure thought of the 20th century.

The most straightforward version of Turing’s approach was in an article by S. C. Kleene in the *Encyclopaedia Britannica*: Let’s define a **number-theoretic function** to be a function of a single argument, which is an unsigned integer, and which produces as its value, if there is one, another unsigned integer. Let $f_k(n)$ be the number-theoretic function calculated by the k th computer program, including the possibility that $f_k(n)$ is undefined because the computation doesn’t terminate successfully. So sometimes these are partial, not total functions. Consider the number-theoretic function $F(n) = f_n(n) + 1$ if $f_n(n)$ is defined, and 0 otherwise:

$$F(n) = \begin{cases} f_n(n) + 1 & \text{if } f_n(n) \text{ is defined,} \\ 0 & \text{if } f_n(n) \text{ is not defined.} \end{cases}$$

Clearly F can’t be one of the f_k , therefore F is uncomputable. Hence there can be no algorithm to decide if $f_n(n)$ is defined, therefore, following Turing, no formal axiomatic theory that enables you to prove whether $f_n(n)$ is defined or not in each particular case. For otherwise you could systematically run through the tree of all possible formal proofs and automatically decide whether or not $f_n(n)$ is defined, which would enable you to calculate the uncomputable function F !

This I could understand, this I could add to my intellectual toolkit. I immediately found an incompleteness theorem of my own. Let’s imagine that we are using a formal axiomatic theory to prove that f_k , for specific,

individual values of k , is a total function or isn't, whichever the case may be. Why can't we always do this?

First, let's forget about f_k , which includes partial functions. We'll use a different list of functions, g_k , all of which are total.

Consider a specific formal axiomatic theory in which it is possible to prove that a computer program computes a total number-theoretic function. Consider all the computer programs that **provably** compute a total number-theoretic function. List these programs in the order that such proofs are discovered as we systematically work our way through the tree of all possible formal proofs in our theory. This gives us a list g_k of all the computable number-theoretic functions which we can prove to be total in our theory. Now define $G(n)$ to be 2 raised to the maximum value of $g_k(n)$ for all k from 1 to n :

$$G(n) = 2^{\lceil \max_{k=1}^n g_k(n) \rceil}.$$

$G(n)$ is computable and it's a total number-theoretic function that goes to infinity with n much faster than any of the g_k functions. But the g_k include **all** the provably total computable number-theoretic functions in our formal axiomatic theory. Hence this theory cannot prove that the program for G computes a total number-theoretic function, even though it does. *Ipsa facto*, we get incompleteness, and also a way to give an upper bound G on the power of a formal axiomatic theory using computational ideas.

Let's change topic. Fix the programming language for the entire discussion, and consider the **size** of a computer program, measured in bits of code. What if we want to show that we have the smallest computer program for a particular computational task, for example, the smallest computer program that computes a particular number-theoretic function? Let's call such a program **elegant**.

Can we do this?

Well, it's easy to see that any given formal axiomatic theory can only enable us to prove that finitely many specific, individual programs Π are elegant. And my proof of this gives us another way to measure, to bound, the power of a formal axiomatic theory, this time using the size in bits σ of the smallest computer program Θ that runs through all the proofs and generates all the theorems (a never-ending task). If we want to prove that a program is elegant which is substantially larger than σ , we will fail.

Why?

The proof, a *reductio ad absurdum*, is as follows. Consider the first prov-

ably elegant program Π that is larger than twice the size σ of Θ , the program for running through the proofs and finding all the theorems. *I.e.*, Π appears in the first proof discovered by Θ that a specific program with size $\geq 2\sigma$ is elegant. Π is at least twice the size of Θ , but instead of using Π directly, we can instead take Θ , run it until it produces Π , and then run Π . This indirect way of getting Π reduces Π to half the size! And with more attention to technical details, and provided the programming language has certain suitable features, we can sharpen this result as follows: If a program Π is larger in size than the program Θ for producing all the theorems in our formal axiomatic theory, then this theory cannot enable us to prove that Π is elegant.

In more colorful language, if a program is larger than the complexity of a formal theory, then there is no way to prove that it's an elegant program within that theory. To prove that an n -bit program is elegant, you need an n -bit theory. Throughout this discussion—indeed, this entire essay—I make the implicit assumption that the formal axiomatic theory in question proves only true theorems.

Our constructions of F and G via diagonal arguments do in fact contain a self-reference, although a more palatable one than in Gödel's proof, namely when we look at $f_n(n)$ or $g_n(n)$. The unprovability-of-elegance proof that I have just given is genuinely different. It is **not** a diagonal argument, not even a concealed one. It's an information-theoretic argument.

The old idea of diagonal constructions, inspired by Cantor's diagonal argument in the theory of infinite sets, or in my case, actually, by Paul du Bois-Reymond's argument about orders of infinity that I read in a little book by G. H. Hardy, is replaced by something radically different. This is the idea of measuring the complexity of something by the size in bits of the smallest program for calculating it.

Let me give a more sophisticated illustration of these ideas. It involves the halting probability Ω , which is my version of Turing's famous halting problem, converted into a single real number Ω between 0 and 1 that is in fact the most compact oracle for solving the halting problem. If you were told the first n bits of the base-two binary expansion of the numerical value of Ω , that would in principle enable you to solve the halting problem for all programs up to n bits in size.

But the most interesting thing about Ω is not that it can serve as an oracle for the halting problem, it's the fact that the base-two bits of Ω are **irreducibly complex**. In other words, a program that computes n bits of Ω must itself be at least n bits in size. Furthermore, a formal axiomatic theory

Θ that enables you to prove what are the first n bits of Ω (whether each bit is a 0 or is a 1) must itself have complexity n or more. In other words, the program for generating the theorems of Θ must be at least n bits long.

(*Caveat*: These statements depend on the appropriate choice of programming language for measuring program-size complexity, algorithmic information content or algorithmic complexity, whatever you may prefer to call it. The precise numerical value of Ω also depends on your choice of programming language. I shall not discuss any of this here.)

Right away, very concretely, the infinite stream of base-two bits of Ω shows that pure mathematics contains zones which are irreducibly complex. This example of irreducible complexity may seem a bit artificial, a bit contrived. However, the bits of Ω can be dressed up as statements about properties of diophantine equations or the word problem for semigroups, mathematical questions that on the face of it seem to have little to do with computation. *Immediate corollary*: Pure mathematics is infinitely complex and no finite set of axioms can fully exhaust it. In algebraic terms, the world of mathematical ideas has no finite basis.

As I learnt much later, the idea of irreducible complexity can be traced via Hermann Weyl back to Leibniz's 1686 *Discours de métaphysique*. And Vladimir Tasić points out that Emile Borel anticipated Ω , at least partially, with his 1927 know-it-all real number that can answer every possible yes/no question in French, by using the n th digit after the decimal point to answer the n th question.

Following the lead of Weyl and Tasić, I have studied the work of Leibniz and Borel and have done my best to advertise it and to make people appreciate how important it is. After all, Borel's 1927 number is extremely uncomputable, and this was a decade **before** Turing's famous paper "On computable numbers. . ."

And we are immediately confronted with fundamental philosophical questions. In the case of Borel, ontological ones: Does his know-it-all real number really exist? Does Ω ? Do real numbers exist? Can anything in the physical world be measured with infinite precision? Is physical reality continuous or discrete? Does any physical system contain an infinite number of bits of information? Does the entire universe? Zeno of Elea, all over again!

Even more important, through the connection with Leibniz, we are face to face with fundamental epistemological issues. What is a law of nature? Weyl expresses Leibniz's response to this question (*Discours*, Section VI) in a particularly dramatic fashion: If arbitrarily complex laws are permitted,

then the concept of law becomes vacuous because there is always a law!

Let's restate this in terms of algorithmic information: A scientific theory is only of value to the extent that it's a compression. The number of bits in the theory (considered as software) must be substantially smaller than the number of bits of empirical data that we are trying to understand/explain. Understanding is compression of information, a good explanation is a good compression. (I am **not** discussing prediction.)

Furthermore, the best theory is the smallest computer program that reproduces exactly the empirical data. In other words, the best theory is what I've called an elegant program for calculating the experimental data.

This permits us to quantify Leibniz's dictum (Section V of the *Discours*) that science is possible, that the world is comprehensible, precisely because God simultaneously minimizes the complexity of the laws of nature and maximizes the richness and diversity of the universe that these laws determine. In my toy model, both the laws of nature and the resulting universe are represented as a finite string of bits, and we merely compare their size. The laws are a program and the universe is its output.

What more can we do with these ideas, with this view of epistemology as information theory? It seems to me that viewed from this information-theoretic computational perspective, physical theories and mathematical theories are not that different. One is a compression of facts from the laboratory; the other compresses facts discovered with the aid of the computer, which is our laboratory when we do pure mathematics.

I have argued for many years that this provides support for what Imre Lakatos named a **quasi-empirical** view of mathematics, and that even if you are a firm believer in the reality of the Platonic world of mathematical ideas, you may sometimes have to add new, complicated, non-self-evident axioms to mathematics for pragmatic reasons, in order to be able to organize our mathematical experience better. Indeed, mathematicians, in working with the Riemann hypothesis, the hypothesis that $\mathbf{P} \neq \mathbf{NP}$, or the axiom of projective determinacy, are already behaving like physicists.

Most pure mathematicians, the true believers, the Platonists, will cry "heresy!" Materialist mathematicians may perhaps have some sympathy for my position, but they will deny the reality of Ω ! I attempt to balance delicately between these two positions, having learned from Leibniz, whom I greatly admire, that some ideas are not as irreconcilable as most people think.

Here is a completely different area where these ideas might apply. For

many years I have hoped that perhaps the kind of complexity that I have been talking about in this essay has some kind of connection with biological complexity, and might eventually lead to a mathematical discussion of whether biogenesis and evolution are likely or not. I am thinking of an abstract model of life as software (our DNA!) that avoids getting into the messy details of molecular biology and biochemistry.

Furthermore, if mathematics is not static, if it is not a particular universal formal axiomatic system as Hilbert had hoped, then what is it? I would very much like to see a mathematical model of how mathematics evolves, of creativity, of how new ideas arise. Such a theory might also tell us how new biological ideas, new species, are created.

(A Platonist would phrase this differently: The world of pure mathematics, mathematical reality, is static, eternal and perfect. It is our imperfect and limited **knowledge** of this perfect world that is constantly evolving, constantly changing, and quasi-empirical.)

Clearly, much remains to be done; that is as far as I've been able to take these ideas so far. For more detailed discussions, see my non-technical book *Meta Math!*, the collection of my philosophical papers, *Thinking about Gödel & Turing*, or the *festschrift* Calude, *Randomness & Complexity, from Leibniz to Chaitin*.

There is also a U.K. edition of *Meta Math!* with a slightly different title, *Meta Maths*, and various translations into foreign languages are in progress.

The work on the word problem for semigroups that I referred to above is quite recent. My paper on this subject, "An algebraic characterization of the halting probability," will be published in volume 79 of *Fundamenta Informaticae*.

May 16, 2007