Computers, Paradoxes and the Foundations of Mathematics

Some great thinkers of the 20th century have shown that even in the austere world of mathematics, incompleteness and randomness are rife

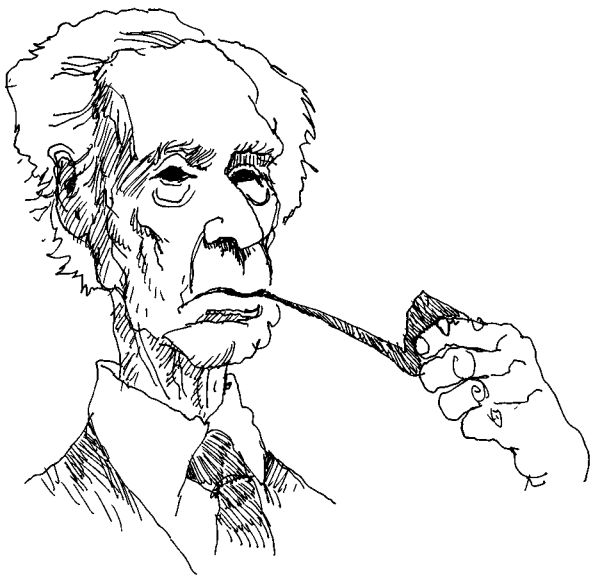
Gregory J. Chaitin

Everyone knows that the computer is a very practical thing. In fact, computers have become indispensable to running a modern society. But what even computer experts don't remember is that—I exaggerate only slightly—the computer was invented in order to help to clarify a philosophical question about the foundations of mathematics. Surprising? Yes indeed.

This amazing story begins with David Hilbert, a well-known German mathematician who at the start of the 20th century proposed formalizing completely all of mathematical reasoning. It turned out that you can't formalize mathematical reasoning, so in one sense his idea was a tremendous failure. Yet in another way, Hilbert's idea was a success, because formalism has been one of the biggest boons of the 20th century—not for mathematical reasoning or deduction, but for programming, for calculating, for computing. This is a forgotten piece of intellectual history.

I will relate that history here without delving into mathematical details. So it will be impossible to fully explain the work of the relevant contributors, which include Bertrand Russell, Kurt Gödel and Alan Turing. Still, a patient reader should be able to glean the essence of their arguments and see what inspired some of my own ideas about the randomness inherent in mathematics.

Gregory J. Chaitin is a mathematician at the IBM Watson Research Center in Yorktown Heights, New York. He is also a visiting professor at the University of Buenos Aires and at the University of Auckland. For the past 35 years, he has been the principal architect of algorithmic information theory, which he invented as a teenager. His latest advance has been to transform algorithmic information theory so that it offers predictions about the size of real computer programs. Chaitin has written seven books: Algorithmic Information Theory *(Cambridge University Press);* Information, Randomness & Incompleteness *and* Information-Theoretic Incompleteness *(both World Scientific); and* The Limits of Mathematics, The Unknowable, Exploring Randomness *and (most recently)* Conversations with a Mathematician *(all Springer-Verlag). This article is excerpted from a lecture that he delivered at the University of Massachusetts, Lowell, in 1999, which appears in full in* Conversations with a Mathematician. *This material is reproduced here with the permission of Springer-Verlag. Address: IBM Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598. Internet: chaitin@us.ibm.com*



Russell's Logical Paradoxes

Let me start with Bertrand Russell, a mathematician who later turned into a philosopher and finally into a humanist. Russell is key because he discovered some disturbing paradoxes in logic itself. That is, he found cases where reasoning that seems to be sound leads to contradictions. Russell was tremendously influential in spreading the idea that these contradictions constituted a serious crisis and had to be resolved somehow.

The paradoxes that Russell discovered attracted a great deal of attention in mathematical circles, but strangely enough only one of them ended up with his name on it. To understand the Russell paradox, consider the set of all sets

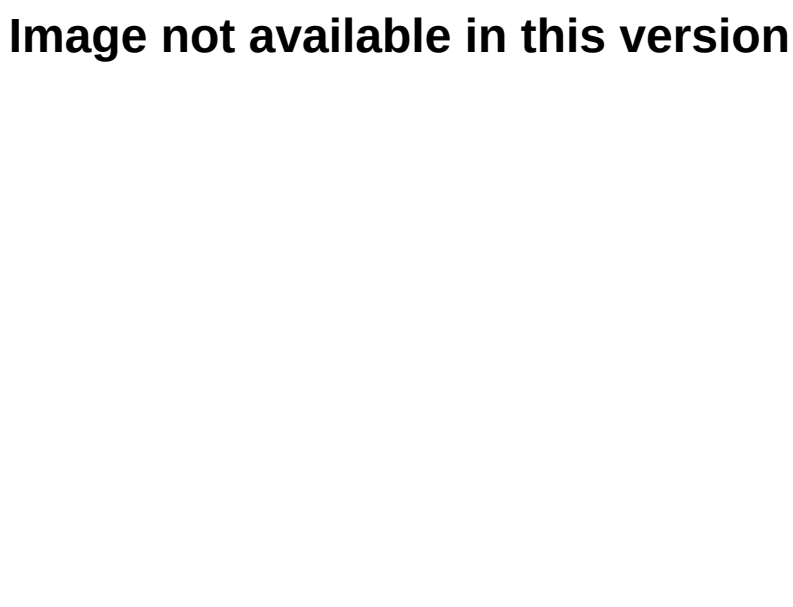


Figure 1. ***Drawing Hands*****, composed by M. C. Escher in 1948, provides a visual analogy for the so-called Russell paradox, named for the British mathematician Bertrand Russell, who brought this problem in logic to the attention of his contemporaries in the early years of the 20th century—eventually inspiring the work of Kurt Gödel, Alan Turing and the author on the limits of mathematics. One form the Russell paradox takes is the pair of statements: "The following sentence is false. The preceding sentence is true." Either assertion alone seems reasonable (that is, it may either be true or false), but together the sentences cannot be evaluated for their truth or falsehood. It is their combination that creates the paradox, just as with the two hands of Escher's drawing.**

that are not members of themselves. Then ask, "Is this set a member of itself?" If it is a member of itself, then it shouldn't be, and vice versa.

The set of all sets in the Russell paradox is like the barber in a small, remote town who shaves all the men who don't shave themselves. That description seems pretty reasonable, until you ask, "Does the barber shave himself?" He shaves himself if and only if he doesn't shave himself. Now you may say, "Who cares about this hypothetical barber? This is just silly word play!" But when you're dealing with the mathematical concept of a set, it's not so easy to dismiss a logical problem.

The Russell paradox is a set-theoretic echo of an earlier paradox, one that was known to the

ancient Greeks. It is often called the Epimenides paradox or the paradox of the liar. The essence of the problem is this: Epimenides was said to exclaim, "This statement is false!" Is it false? If his statement is false, that means that it must be true. But if it's true, it's false. So whatever you assume about his veracity, you're in trouble. A two-sentence version of the paradox goes like this: "The following statement is true. The preceding statement is false." Either statement alone is okay, but combined they make no sense. You might dismiss such conundrums as meaningless word games, but some of the great minds of the 20th century took them very seriously.

One of the reactions to the crisis of logic was Hilbert's attempt to escape into formalism. If one gets into trouble with reasoning that seems okay, the solution is to use symbolic logic to create an artificial language and be very careful to specify the rules so that contradictions don't crop up. After all, everyday language is ambiguous—you never know what a pronoun refers to.

Hilbert's Rescue Plan

Hilbert's idea was to create a perfect artificial language for reasoning, for doing mathematics, for deduction. Hence, he stressed the importance of the axiomatic method, whereby one works from a set of basic postulates (axioms) and well-defined rules of deduction to derive valid theorems. The notion of doing mathematics that way goes back to the ancient Greeks and particularly to Euclid and his geometry, which is a beautifully clear mathematical system.

In other words, Hilbert's intention was to be completely precise about the rules of the game—about the definitions, the elementary concepts, the grammar and the language—so that everyone could agree on how mathematics should be done. In practice it would be too much work to use such a formal axiomatic system for developing new mathematics, but it would be philosophically significant.

Hilbert's proposal seemed fairly straightforward. After all, he was just following the formal traditions in mathematics, drawing from a long history of work by Leibniz, Boole, Frege and Peano. But he wanted to go all the way to the very end and formalize *all* of mathematics. The big surprise is that it turned out that this could not be done. Hilbert was wrong—but wrong in a tremendously fruitful way, because he had asked a very good question. In fact, by asking this question he created an entirely new discipline called *metamathematics*, an introspective field of math in which you study what mathematics can or cannot achieve.

The basic concept is this: Once you entomb mathematics in an artificial language *à la* Hilbert, once you set up a completely formal axiomatic system, then you can forget that it has any meaning and just look at it as a game played with marks on paper that enable you to deduce theorems from axioms. Of course, the reason one does mathematics is because it has meaning. But if you want to be able to study mathematics using mathematical methods, you have to crystallize out the meaning and just examine an artificial language with completely precise rules.

What kind of questions might you ask? Well, one question is whether one can prove, say, that 0 = 1. (We can hope not.) Indeed, for any statement, call it A, you can ask if it's possible to prove either A or the opposite of A. A formal axiomatic system is considered complete if you either prove that A is true, or prove that it's false.

Hilbert envisioned creating rules so precise that any proof could always be submitted to an unbiased referee, a mechanical procedure that would say, "This proof obeys the rules," or perhaps, "On line 4 there's a misspelling" or, "This thing on line 4 that supposedly follows from line 3, actually doesn't." And that would be the end; no appeal.

His idea was not that mathematics should actually be done this way, but rather that if you could take mathematics and do it this way, you could then use mathematics to study the power of mathematics. And Hilbert thought that he was actually going to be able to accomplish that feat. So you can imagine just how very, very shocking it was in 1931 when an Austrian mathematician named Kurt Gödel showed that Hilbert's rescue plan wasn't at all reasonable. It could never be carried out, even in principle.

Gödel's Incompleteness

Gödel exploded Hilbert's vision in 1931 while on the faculty of the University of Vienna, although he originally came from what is now called the Czech Republic, from the city of Brno. (It was then part of the Austro-Hungarian empire.) Later Gödel was to join Einstein at the Institute for Advanced Study in Princeton.

Gödel's amazing discovery is that Hilbert was dead wrong: There is, in fact, no way to have a formal axiomatic system for all of mathematics in which it is crystal clear whether something is correct or not. More precisely, what Gödel discovered was that the plan fails even if you just try to deal with elementary arithmetic, with the numbers 0, 1, 2, 3,. . . and with addition and multiplication.

Any formal system that tries to contain the whole truth and nothing but the truth about addition, multiplication and the numbers 0, 1, 2, 3,. . . will have to be incomplete. Actually, it will either be inconsistent or incomplete. So if you assume that it only tells the truth, then it won't tell the whole truth. In particular, if you assume that the axioms and rules of deduction don't allow you to prove false theorems, then there will be true theorems that you cannot prove.

Gödel's incompleteness proof is very clever. It's very paradoxical. It almost looks crazy. Gödel starts in effect with the paradox of the liar: the statement "I'm false!," which is neither true nor false. Actually what Gödel does is to construct a statement that says of itself, "I'm unprovable!" Now if you can construct such a statement in elementary number theory, in arithmetic, a mathematical statement that describes itself, you've got to be very clever—but if you *can* do it, it's easy to see that you're in trouble. Why? Because if that statement is provable, it is by necessity false, and you're proving false results. If it's unprovable, as it says of itself, then it's true, and mathematics is incomplete.

Gödel's proof involves many complicated technical details. But if you look at his original paper, you find something that looks a lot like LISP programming in it. That is because Gödel's proof involves defining a great many functions recursively, functions dealing with lists—precisely what LISP is all about. So even though there were no computers or programming languages in 1931, with the benefit of hindsight you can clearly see a programming language at the core of Gödel's original paper.

Another famous mathematician of that era, John von Neumann (who, incidentally, was instrumental in encouraging the creation of computer technology in the United States) appreciated Gödel's insight immediately. It had never occurred to von Neumann that Hilbert's plan was unsound. So Gödel was not only tremendously clever, he had the courage to imagine that Hilbert might be wrong.

Many people saw Gödel's conclusion as absolutely devastating: All of traditional mathematical philosophy ended up in a heap on the floor. In 1931, however, there were also a few other problems to worry about in Europe. There was a major depression, and a war was brewing.

Turing's Machine

The next major step forward came five years later, in England, when Alan Turing discovered uncomputability. Recall that Hilbert had said that there should be a "mechanical procedure" to decide if a proof obeys the rules or not. Hilbert never clarified what he meant by a mechanical procedure. Turing essentially said, "What you really mean is a machine" (a machine of a kind that we now call a Turing machine).

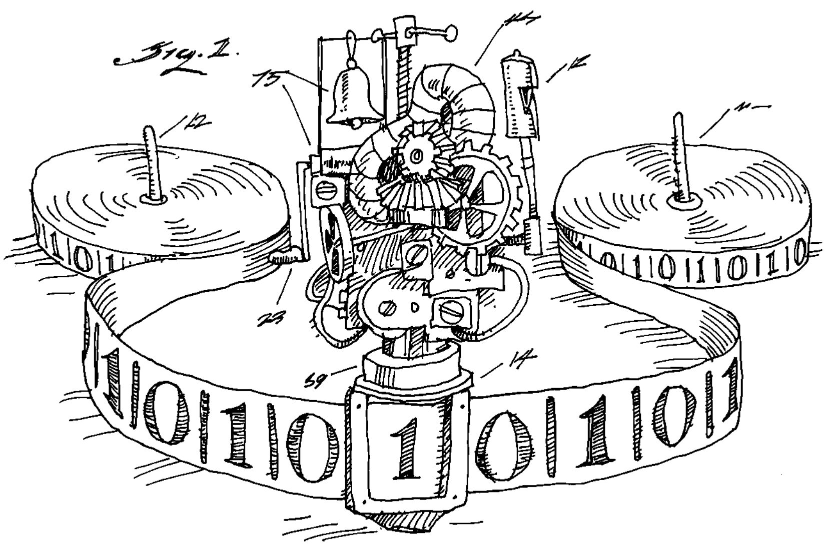


Figure 2. Alan Turing's seminal paper of 1936 introduced the notion of a machine that could perform operations, one cell at a time, on an infinitely long tape. His mental construction has since become known as a "Turing machine." This imaginary device can read what is written on one cell of the tape. Depending on the internal state of the machine, it leaves or modifies that cell and moves the tape one space to the left or right, then repeats the process. Turing showed that such an automaton could use this simple procedure to carry out any conceivable calculation, so long as the machine was given the appropriate set of basic instructions.

Turing's original paper contains a programming language, just as Gödel's paper does, or what we would now call a programming language. But these two programming languages are very different. Turing's isn't a high-level language like LISP; it's more like a machine language, the raw code of ones and zeros that are fed to a computer's central processor. Turing's invention of 1936 is, in fact, a horrible machine language, one that nobody would want to use today, because it's too rudimentary.

Although Turing's hypothetical computing machines are very simple and their machine language rather primitive, they're very flexible. In his 1936 paper, Turing claims that such a machine should be able to perform any computation that a human being can carry out.

Turing's train of thought now takes a very dramatic turn. What, he asks, is *impossible* for such a machine? What can't it do? And he immediately finds a problem that no Turing machine can solve: the halting problem. This is the problem of deciding in advance whether a Turing machine (or a computer program) will eventually find its desired solution and halt.

If you allow a time limit, it's very easy to solve this problem. Say that you want to know whether a program will halt within a year. Then you just run it for a year, and it either halts or doesn't. What Turing showed is that you get in terrible trouble if you impose no time limit, if you try to deduce whether a program will halt without just running it.

Let me outline Turing's reasoning: Suppose that you *could* write a computer program that checks whether any given computer program eventually halts. Call it a termination tester. In theory, you feed it a program, and it would spew out an answer: "Yes, this program will terminate," or, "No, it will go off spinning its wheels in some infinite loop and never come to a halt." Now create a second program that uses the termination tester to evaluate some program. If the program under investigation terminates, have your new program arranged so that it goes into an infinite loop. Here comes the subtle part: Feed your new program a copy of itself. What does it do?

Remember, you've written this new program so that it will go into an infinite loop if the program under test terminates. But here it is *itself* the program under test. So if it terminates, it goes off in an infinite loop, which means it doesn't terminate—a contradiction. Assuming the opposite outcome doesn't help: If it doesn't terminate, the termination tester will indicate this, and the program will not go into an infinite loop, thus terminating. The paradox led Turing to conclude that a general purpose termination tester couldn't be devised.

The interesting thing is that Turing immediately deduced a corollary: If there's no way to determine in advance by a calculation whether a program will halt, there also cannot be any way to decide it in advance using reasoning. No formal axiomatic system can enable you to deduce whether a program will eventually halt. Why? Because if you could use a formal axiomatic system in this way, that would give you the means to calculate in advance whether a program will halt or not. And that's impossible, because you get into a paradox like, "This statement is false!" You can create a program that halts if and only if it doesn't halt. The paradox is similar to what Gödel found in his investigation of number theory. (Recall he was looking at nothing more complicated than 0, 1, 2, 3,. . . and addition and multiplication.) Turing's coup is that he showed that *no* formal axiomatic system can be complete.

After World War II broke out, Turing began working on cryptography, von Neumann started working on how to calculate atom-bomb detonations, and people forgot about the incompleteness of formal axiomatic systems for a while.

Randomness in Mathematics

The generation of mathematicians who were concerned with these deep philosophical questions basically disappeared with World War II. Then I showed up on the scene.

In the late 1950s, when I was a youngster, I read an article on Gödel and incompleteness in *Scientific American*. Gödel's result fascinated me, but I couldn't really understand it; I thought there was something fishy. As for Turing's approach, I appreciated that it went much

deeper, but I still wasn't satisfied. This is when I got a funny idea about randomness.

When I was a kid, I also read a lot about another famous intellectual issue, not the foundations of mathematics but the foundations of physics—about relativity theory and cosmology and even more often about quantum mechanics. I learned that when things are very small the physical world behaves in a completely crazy way. In fact, things are random—intrinsically unpredictable. I was reading about all of this, and I started to wonder whether there was also randomness in pure mathematics. I began to suspect that maybe this was the real reason for incompleteness.

A case in point is elementary number theory, where there are some very difficult questions. Consider the prime numbers. Individual prime numbers behave in a very unpredictable way, if you're interested in their detailed structure. It's true that there are statistical patterns. There's a thing called the prime number theorem that predicts fairly accurately the overall average distribution of the primes. But as for the detailed distribution of individual prime numbers, that looks pretty random.

So I began to think that maybe the inherent randomness in mathematics provided a deeper reason for all this incompleteness. In the mid-60s I, and independently A. N. Kolmogorov in the U.S.S.R., came up with some new ideas, which I like to call algorithmic information theory. That name makes it sound very impressive, but the basic idea is simple: It's just a way to measure computational complexity.

One of the first places that I heard about the idea of computational complexity was from von Neumann. Turing considered the computer as a mathematical concept—a perfect computer, one that never makes mistakes, one that has as much time and space as it needs to do its work. After Turing came up with this idea, the next logical step for a mathematician was to study the time needed to do a calculation—a measure of its complexity. Around 1950 von Neumann highlighted the importance of the time complexity of computations, and that is now a well-developed field.

My idea was not to look at the time, even though from a practical point of view time is very important. My idea was to look at the *size* of computer programs, at the amount of information that you have to give a computer to get it to perform a given task. Why is that interesting? Because program-size complexity connects with the notion of entropy in physics.

Recall that entropy played a particularly crucial role in the work of the famous 19th-century physicist Ludwig Boltzmann, and it comes up in the fields of statistical mechanics and thermodynamics. Entropy measures the degree of disorder, chaos, randomness, in a physical system. A crystal has low entropy, and a gas (say, at room temperature) has high entropy.

Entropy is connected with a fundamental philosophical question: Why does time run in just one direction? In everyday life, there is, of course, a great difference between going backward and forward in time. Glasses break, but they don't reassemble spontaneously. Similarly, in Boltzmann's theory, entropy has to increase—the system has to get more and more

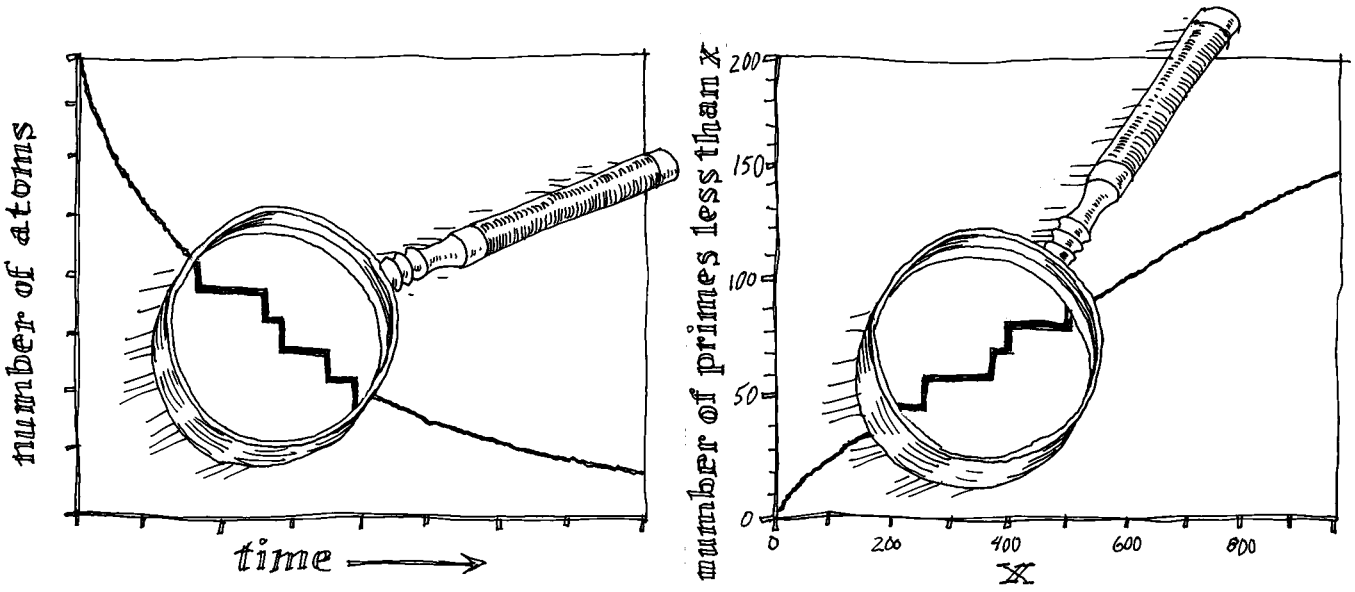


Figure 3. Quantum mechanics reflects the role of randomness in physics. The seemingly smooth, regular decay of a radioactive substance, for example, is in reality made up of a series of discrete steps, where the exact moment that the next atom decays cannot be predicted *(left)*. The author's work highlights a similar randomness in mathematics, which can be seen, for instance, in the distribution of prime numbers. Although the number of primes less than *x* follows a well-known trend, the curve is made up of a series of erratic steps, and the exact value of the next larger prime cannot be predicted precisely from any general theory *(right)*.

Figure 4. Kurt Gödel's work led to the modern understanding that randomness is inherent in mathematics just as it is in physics—a notion that Albert Einstein resisted. Still, these two men were close friends when they both lived in Princeton. (Courtesy of the archives of the Institute for Advanced Study.)

disordered. This is the well-known Second Law of Thermodynamics.

Boltzmann's contemporaries couldn't see how to deduce that result from Newtonian physics. After all, in a gas, where the atoms bounce around like billiard balls, each interaction is reversible. If you were somehow able to film a small portion of a gas for a brief time, you couldn't tell whether you were seeing the movie run forward or backward. But Boltzmann's gas theory says that there *is* an arrow of time—a system will start off in an ordered state and will end up in a very mixed up disordered state. There's even a scary expression for the final condition, "heat death."

The connection between my ideas and Boltzmann's theory comes about because the size of a computer program is analogous to the degree of disorder of a physical system. A gas might take a large program to say where all its atoms are located, whereas a crystal doesn't require as big a program at all, because of its regular structure. Entropy and program-size complexity are thus closely related.

This concept of program-size complexity is also connected with the philosophy of the scientific method. Ray Solomonoff (a computer scientist then working for the Zator Company in Cambridge, Massachusetts) proposed this idea at a conference in 1960, although I only learned of his work after I came up with some very similar ideas on my own a few years later. Just think of Occam's razor, the idea that the simplest theory is best. Well, what's a theory? It's a computer program for predicting observations. And the statement that the simplest theory is best translates into saying that a concise computer program constitutes the best theory.

What if there is no concise theory? What if the most concise program for reproducing a given body of experimental data is the same size as the data set? Then the theory is no good—it's cooked up—and the data are incomprehensible, random. A theory is good only to the extent that it compresses the data into a much smaller set of theoretical assumptions and rules for deduction.

So you can define randomness as something that cannot be compressed at all. The only way to describe a completely random object or number to someone else is to present it and say, "This is it." Because it has no structure or pattern, there is no more concise description. At the other extreme is an object or number that has a very regular pattern. Perhaps you can describe it by saying that it is a million repetitions of 01, for example. This is very big object with a very short description.

My idea was to use program-size complexity to define randomness. And when you start looking at the size of computer programs—when you begin to think about this notion of program-size or information complexity instead of run-time complexity—then an interesting thing happens: Everywhere you turn, you find incompleteness. Why? Because the very first question you ask in my theory gets you into trouble. You measure the complexity of something by the size of the smallest computer program for calculating it. But how can you be sure that what you have is the smallest computer program possible? The answer is that you can't. This task escapes the power of mathematical reasoning, amazingly enough.

Showing why this is so gets somewhat involved, so I'm just going to quote the actual result, which is one of my favorite statements of incompleteness: If you have n bits of axioms, you can never prove that a program is the smallest possible if it is more than n bits long. That is, you get into trouble with a program if it's larger than a computerized version of the axioms—or more precisely, if it's larger than the size of the proof-checking program for the axioms and the associated rules of deduction.

So it turns out that you cannot in general

calculate program-size complexity, because to determine the program-size complexity of something is to know the size of the most concise program that calculates it. You can't do that if the program is larger than the axioms. If there are *n* bits of axioms, you can never determine the program-size complexity of anything that has more than *n* bits of complexity, which means almost everything.

Let me explain why I claim that. The sets of axioms that mathematicians normally use are fairly concise, otherwise no one would believe in them. In practice, there's this vast world of mathematical truth out there—an infinite amount of information—but any given set of axioms only captures a tiny, finite amount of this information. That, in a nutshell, is why Gödel incompleteness is natural and inevitable rather than mysterious and complicated.

Where Do We Go from Here?

This conclusion is very dramatic. In only three steps one goes from Gödel, where it seems shocking that there are limits to reasoning, to Turing, where it looks much more reasonable, and then to a consideration of program-size complexity, where incompleteness, the limits of mathematics, just hits you in the face.

People often say to me, "Well, this is all very well and good. Algorithmic information theory is a nice theory, but give me an example of a specific result that you think escapes the power of mathematical reasoning." For years, one of my favorite answers was, "Perhaps Fermat's last theorem." But a funny thing happened: In 1993 Andrew Wiles came along with a proof. There was a misstep, but now everyone is convinced that the proof is correct. So there's the problem. Algorithmic information theory shows that there are lots of things that you can't prove, but it cannot reach a conclusion for individual mathematical questions.

How come, in spite of incompleteness, mathematicians are making so much progress? These incompleteness results certainly have a pessimistic feeling about them. If you take them at face value, it would seem that there's no way to progress, that mathematics is impossible. Fortunately for those of us who do mathematics, that doesn't seem to be the case. Perhaps some young metamathematician of the next generation will prove why this has to be so.

Bibliography

Casti, J. L., and W. DePauli. 2000. *Gödel: A Life of Logic*. Cambridge, Mass.: Perseus Publishing.

Chaitin, G. J. 1975. Randomness and mathematical proof. *Scientific American* 232(5):47–52.

Chaitin, G. J. 1988. Randomness in arithmetic. *Scientific American* 259(1):80–85.

Hofstadter, D. R. 1979. *Gödel, Escher, Bach: An Eternal Golden Braid*. New York: Basic Books.

Nagel, E., and J. R. Newman. 1956. Gödel's Proof. *Scientific American* 194(6):71–86.

Nagel, E., and J. R. Newman. 1958. *Gödel's Proof*. New York: New York University Press.

Links to Internet resources for further exploration of "Computers, Paradoxes and the Foundations of Mathematics" are available on the American Scientist Web site:

http://www.americanscientist.org/articles/02articles/chaitin.html