A mathematician sees room for miracles. July 2009 (minor typos fixed 2/2010)

Edward Ordman edward @ ordman.net Comments requested

This is an -extremely- rough draft. I'd like to write it up more formally for an interdisciplinary or science/religion journal, perhaps such as **Soundings** or **Zygon**

Mathematics is the loom upon which God weaves the fabric of the universe. Clifford A. Pickover, The Loom of God: Mathematical Tapestries at the Edge of Time, Plenum Press, New York 1997, p 16, quoted in American mathematical Monthly 155(2008) p. 614.

Leopold Kronecker (December 7, 1823 – December 29, 1891) was a German mathematician and logician who argued that arithmetic and analysis must be founded on "whole numbers", saying, "God made the integers; all else is the work of man" (Bell 1986, p. 477). [wikipedia]

Those of you who disliked mathematics in school may take some comfort from this. Speaking somewhat facetiously, he was saying that one, two, three, and so on, come pretty naturally: if you found fractions, decimals, and trigonometry more daunting, that is because those human inventions really are less natural.

In 1960, the physicist Eugene Wigner published an article titled "The Unreasonable Effectiveness of Mathematics in the Natural Sciences". In it, he (1) observed that the mathematical structure of a physics theory often points the way to further advances in that theory and even to empirical predictions, and (2) argued that this is not just a coincidence and therefore must reflect some larger and deeper truth about both mathematics and physics. [wikipedia]Eugene Wigner, 1960, "The Unreasonable Effectiveness of Mathematics in the Natural Sciences," Communications on Pure and Applied Mathematics 13(1): 1–14.
Reprinted at http://www.dartmouth.edu/~matc/MathDrama/reading/Wigner.html
Inspired many others, e.g.
The Unreasonable Effectiveness of Mathematics
by R. W. HAMMING
Reprinted From: The American Mathematical Monthly Volume 87 Number 2 February 1980
reprinted at http://www.dartmouth.edu/~matc/MathDrama/reading/Hamming.html

God created a universe with rational enough rules that Man can make progress understanding it. He created a complex enough universe to guarantee that our children and grandchildren, as long a humanity lasts, will still have new things to discover and understand.

It is entirely rational to believe in the existence of things that we cannot see or point to examples of.

I want to make mathematical sense out of some of these things. I am, myself, a rather traditional believer, but among my friends I have many believers in "religious naturalism" or other approaches that want to gain their inspiration from the natural world. One question that arises, both among theists and naturalists, is, "how much of the world can we understand?" There are a number of arguments made that we cannot completely understand the world, cannot predict the future. There are arguments from quantum physics, from Heisenberg uncertainty: we cannot measure accurately enough. There are arguments from chaos theory, or from the so-called

"butterfly effect": the tiny movement of air caused by a butterfly flapping its wings in Japan can eventually change the weather in America: at least, the present state of the systems of differential equations we use to extrapolate air movements at least admits this possibility.

I find those "advanced science" arguments a bit too abstract, or maybe a bit far over my head. I somehow have this sense that some of the quantum or Heisenberg problems may go away with better understanding of the physics of tiny times and spaces, or that better measurement or better equations may lead us to be accurately able to predict consequences of that butterfly.

But I see things in the parts of mathematics I do understand, and in objects that to me seem "everyday", that justify some of my earlier statements about the complexity of the world, and the existence of "unseen things". I want to try to at least sketch some of that mathematics, by way of letting people share some of a mathematician's or computer scientists view of the "ultimately unknowable."

Pick a number - a very large number - expressed as a string of digits - at random. What makes you think it is random? Let me be a bit more precise. The sequence 010101....01, "01" repeated 500 times, does not appear to be random. 141592.....0 may appear to be random, unless you happen to know that it is the decimal part of "pi" to some large number of places. Let me give you a definition of "random", and a couple of arguments due to the researcher Gregory Chaitan of IBM.

Let's take a digital computer. There are mathematical theorems to tell us that if a problem can be solved on one computer, it can be solved on any other (if there is enough memory), so I don't have to tell you which computer: the one on your desk will work for anything I'm going to say, as long as you allow it to tell you when necessary "please go to the store, buy me a bigger external disk, and plug it in." (For those who want real technicalities, I've just given you a computer called a "Turing machine" with an unlimited tape to use as memory.)

A computer program, in the machine, is just a string of bits - zeros and ones - so just a string of numbers. By the "length" of a program I mean the length of the string of zeros and ones that represent the program. By a "big number" I also mean a string of binary digits, with bigness measured simply by its length. For example, a string of zeros and ones ten thousand digits long ought to be big enough for any argument I want to make.

With those understandings: A number is "random" if the shortest program to print it out is at least as long as the number. Now we can see easily that "01" repeated 5000 times is not random: the program 'Print "10" 5000 times' is a quite short program, even when translated into formal computerese. Informally: a number is random if you can't describe it in any shorter way than just telling the whole number. Let's allow, somewhat arbitrarily, 24 binary digits (bits) for the program "print this." So for the program to print the number to be no longer than the number, the description or formula contained in the program has to be at least 24 bits shorter than the number itself.

Now we can easily prove: there are a great many random numbers. Why? Well, how many ten thousand digit numbers are there? All the numbers from 000....000 to 111...111, which is 2^10000 numbers. And how many computer programs are there shorter than that? Once we allow 24 bits for "Print this", what is left are at most 2^(10000-24) programs. That's a lot less. In fact, on this basis, only about one number in 65 million of those every large numbers fails to be random. (I got that "65 million" by multiplying 2 by itself 24 times.)

If there are so many random numbers, what is an example of one? Well, that presents a problem. Suppose I give you a number, "12482407310056...765" and say, "this is random". You are entitled to say, "Prove it." How do I prove it? One way would be to argue that no program of the right length has it as its output. Perhaps, just perhaps, I could list all of the possible programs and determine the output of each of them (although after a later section, you may agree that that isn't very feasible either.) In any case, the work would be too great. Instead I'm going to show you the opposite - there is no proof that that number is random. In fact, there is no proof that any particular 10,000 digit number is random. How can I do that?

This is a variant of a famous paradox studied by Bertrand Russell. Suppose there is a number that can be proven to be random. If so, there must be a smallest such number (in the ordinary numeric sense that 000100..000 is smaller than 000100..001). Consider "the smallest number of 10000 digits that can be proven to be random," and call it M. M, we are told, can be proven to be random. But "the smallest number that can be proven to be random" is a description of M that is much shorter than 10,000 digits, so, by our definition, M is not random.

That description in words of M, you may rationally object, is not a computer program. Here is where the work of Chaitan comes in. Every "proof" is a string of symbols. A string of symbols can be converted, by giving an appropriate set of translation rules, in to a sequence of zeros and ones. And a computer program can be written which tests such a string of symbols, to see if it represents a valid proof. The proof may be very long; the test may be very long. But it is possible to write a program ***P*** which generates strings of bits, and tests them, and looks to see if they are a valid proof of a statement of the form "M is not the output of a program shorter than 10000 bits." If it finds such a proof, it prints M.

Now we can rephrase our argument a bit more carefully. Suppose there is a 10000-bit number M which is provably not the output of a program shorter than 10000 bits. Then there is a first such number that can be discovered by P and printed out. But, it turns out, the program P can be written - and it takes less than 10000 bits to write it. And that in turn means it can never find such a proof - since if it did, it would be itself a program shorter than 10000 bits that prints M, even though it found a proof that no such program exists.

Conclusion: Almost all very large numbers are random. But no particular large number can be proven to be random. And if I go to the effort of writing down, digit by digit, a number that I *believe* to be random, I can never prove to you that it is random. (Note that if I write a reasonably sized computer program, called a "random number generator," the output will *look* random but by

this definition it is provably *not* random since it was produced by a relatively short computer program.)

Theological leap: I believe in the existence of a lot of things that I never see a particular example of. In particular, I am free to believe in miracles (by some definition) even if I never see an incident that I can *prove* to you is a miracle.

SECTION
There are things we do not know. And there always will be.

Sometimes theologians describe a rather naive form of belief called "The God of the Gaps". Wikipedia May 15 2009:

The term goes back to Henry Drummond, a 19th century evangelical lecturer, from his Lowell Lectures on the Ascent of Man. He chastises those Christians who point to the things that science can not yet explain — "gaps which they will fill up with God" — and urges them to embrace all nature as God's, as the work of "... an immanent God, which is the God of Evolution, is infinitely grander than the occasional wonder-worker, who is the God of an old theology."[1]

As the scientific gaps fill, the theory goes, the need for God and power of God diminishes, and will (in a highly pro-science view) eventually vanish.

I'm not a "God of the Gaps" person; I give God a much greater role than that. But my appreciation of God stems in part from the complexity of the universe: the amazing number of things that we can learn and understand, and the fact that we are guaranteed that there will always be more. In "Gap" language: The gaps are infinitely large, and always will be. There will always be questions that we cannot answer.

My belief here stems from a theorem called *Godel's Incompleteness Theorem*, which loosely says: whatever axioms we start from, so long as they are complex enough to allow arithmetic, there will always be true statements that we cannot prove. Now, by the same sort of reasoning we have seen already, I'm not going to be able to give you a very concrete example. I can't give you a specific statement S about the usual universe and say (1) S is true, but (2) there is no proof of S. Because if (2) is true, how can I convince you of (1)? Well, there are some fairly esoteric techniques - Godel used them to prove the theorem, and others have come up with other arguments - but I have a favorite of my own, based on the work of the British mathematician Alan Turing (and exposited by Chaitan.)

Let's try an approach as follows. Let's consider a simple computer of some sort, and suppose that a typical *program* has a single *input* and a single *output*. Since we are expressing everything as a string of bits, each of these - the program, the input, and the output, can be thought of as a *number*. (Don't be put off by the format the input and output take. Alan Turing considered them to be a sequence of marks on a tape. We might consider the "input" to be the entire content of the machine's disk(s) when the program starts, and the "output" to be the content when it ends.)

Now, given a program P and an input N, let's start the program running. Two things can happen:

(a) the program eventually stops, with an output M, or (b) the program never stops. Note that a program that "crashes", to use common language, stops with some output (possibly zero.) A program may fail to stop, for example, by going into an infinite loop, repeating itself, or by running off to "infinity" (e.g. adding one to itself over and over. It may take a few thousand years until it runs out of disk space to hold the giant number it is up to, and you can then buy it another disk so it can keep on running....).

So let's ask the question: Given program P and input N, will it ever stop? If the program is well written, typically, it will. And we can check that by running it until it stops. But what if we enter a program, and it runs for a few hours - or a few days - and we begin to wonder if it will ever stop. In some cases (that program that keeps adding one) we can study and see that it won't stop. We may be lucky enough to analyze it and say "yes, it will stop" or "no, it won't". But it may be hard to analyze.

I think every beginning computer programming student has the same idea - why not write a program to check the first program, to see if it will stop. This program, let's call it "T", will have the following property: Run T with inputs P and N. If P with input N stops, have T output "1". If P with input N does not stop, have T output "0".

[The careful reader may want to protest. I defined a computer program to have one input before, but I want T to have two inputs. There is an easy cure. Remembering that P and N are both numbers, use (2^P)*(3^N) as the single input to T. That is, multiply P 2's and N 3's together and use that as an input. T can easily recover P and N, by seeing how many times 2 and 3 divide the input.]

Can we write such a program as T? No, there is no such program. If you want to believe me, you can skip the coming theorem/proof. If you want to try some details, here they are.

Theorem: There is no "checking" program like the one just described that tests all pairs (P, N.).

Proof. Suppose there were such a program T. Define a program U as follows:
Given N, if N(N) stops, that is, T(N,N) = 1, then U goes into an infinite loop and never stops.
If N(N) does not stop, that is, T(N,N) =0, then U stops with output 1.
Now what is T(U,U)? If T(U,U) = 1, U(U) stops, so T(U,U) = 0
If T(U,U) =0, then U(U) does not stop, so T(U,U) = 1.
In each case we have a contradiction. That is, the test program T cannot successfully test the program U on input U. So T cannot test all programs.

Very well, we cannot "mechanically" test all programs for stopping. Maybe there is some way of checking each one, individually? Let's make the following hypothesis.

Hypothesis A. For every pair P, N there is a either a proof that P on input N eventually stops, or a proof that P on N never stops.

That hypothesis is false. For suppose there was such a proof for each pair P, N. We could then write a program that in turn generated every possible sequence of symbols, and checked each sequence to see if it was a correct proof either that "P on N stops" or "P on N does not stop." Such a program might run very very slowly - it might take many years to check a case - but if for the pair (P, N) there was a proof, one way or the other, our program would find it. But if that were the case, our proof-finding program would work as an automatic checking program T - and we have seen that there is no such program.

Therefore Hypothesis A is *false*. There is some program P and some number N such that there is no proof that P on input N stops, and no proof that P on input N never stops.

However, stopping can be checked. If P on input N does eventually stop, we can in fact find that out - by running it until it stops. It may take years, but it will stop. And if it eventually stops, then we could build a proof that it will stop - just by enumerating each step until it does. (This amounts to saying: Theorem: If P on input N stops, there is a proof that P on input N stops.)

Conclusion: There is a program P and an input N such that program P given input N will never stop, but for which there is no proof that it will never stop.

Of course, it there is one such program, there are many of them. A more sophisticated argument - which I won't try here - would show that even if we added more rules (axioms) to help determine which programs run forever, and no matter how many such rules we added, there would still be programs that ran forever but with no proof that they did.

Let's call a program and number pair that runs forever, but for which there is no proof of that it runs forever, an *unpredictable* pair. We can now see that *unpredictable pairs* have a strong similarity to the large random numbers we looked at earlier - there are a great many of them, but we can never point at one and be sure that we have found one. For if we could prove we had one - that is, if we could prove that P, N was an unpredictable pair - we would have a proof that it runs forever, so it would no longer be unpredictable.

I'm now going to make an assertion that goes beyond mathematics - in something like the sense we have been talking about, *the world* is unpredictable. I believe that - I cannot prove it - because I believe that the universe is infinite, and that nothing in principle prevents us from building a computer, setting it running, and adding disk drives as needed when it needs more - so that we can run any programs we want. Some of those programs will, necessarily, be unpredictable. Our failure to be able to predict the future is not due to fine points of quantum mechanics, or because our measurements are too coarse to compute how air moves when the butterfly flaps its wings - our failure is due to the very nature of mathematics. The natural numbers, the counting numbers "created by God", the ones that occur naturally before we invented fractions, decimals, and solid geometry - already are sufficient to guarantee that there are unpredictable things in the world.

Now, I may be wrong, as a matter of physics. Perhaps the universe is finite in extent (but very

large). Perhaps the number of particles is finite, the number of positions they can occupy is finite, and there are a smallest and largest interval of time so that the total number of time intervals available is finite. If there are only finitely many whole numbers that can ever be physically realized, my arguments break down - although in a theoretical rather than a practical sense. Knowing that all computer programs must stop or repeat themselves, because the entire universe is doomed to either stop or repeat itself, is not a very helpful solution to the problems I have in mind.

If I believe, as the mathematical jest goes, "God created the natural numbers", then I must believe that we have an infinite number of problems in front of us, an infinite number of unpredictable situations to be faced. My grandchildren are not going to run out of interesting problems to be solved. And they are going to see a great number of events that still will appear to be, in some sense, random - events in which some people will no doubt sometimes see miracles, even if they are unable to prove that that's what they are.

Polkinghorne Zygon Dec 2006 Zygon 981-982 and also 978. Uncertainty from quantum or small measurement. Can God know the future? The block model.