An Incompleteness Theorem for the Natural World[i]

by Rudy Rucker

rudy@rudyrucker.com

Emeritus Professor, Department of Computer Science, San Jose State University, San Jose, CA

Appeared in Hector Zenil (ed), *Irreducibility and Computational Equivalence. Emergence, Complexity and Computation: 10 Years After Wolfram's A New Kind of Science*. Springer. Berlin. 2013.

Introduction

The philosopher Gottfried Wilhelm von Leibniz is perhaps best known for the fierce controversy that arose between him and Sir Isaac Newton over the invention of calculus. The S-like integral sign that we use to this day is in fact a notation invented by Leibniz.

When Leibniz was a youth of nineteen, he wrote a paper called “De Arte Combinatorica”, in which he tried to formulate a universal algebra for reasoning, in the hope that human thought might some day be reducible to mathematical calculations, with symbols or characters standing for thoughts.

> But to return to the expression of thoughts by means of characters, I thus think that controversies can never be resolved, nor sectarian disputes be silenced, unless we renounce complicated chains of reasoning in favor of simple calculations, and vague terms of uncertain meaning in favor of determinate characters.
>
> In other words, it must be brought about that every fallacy becomes nothing other than a calculating error, and every sophism expressed in this new type of notation becomes in fact nothing other than a grammatical or linguistic error, easily proved to be such by the very laws of this philosophical grammar.
>
> Once this has been achieved, when controversies arise,

> there will be no more need for a disputation between two philosophers than there would be between two accountants. It would be enough for them to pick up their pens and sit at their abacuses, and say to each other (perhaps having summoned a mutual friend): "Let us calculate."[ii]

Let's refer to this notion as Leibniz's dream — the dream of finding a logical system to decide all of the things that people might ever disagree about. Could the dream ever work?

Even if the dream were theoretically possible (which it isn't), as a practical matter it wouldn't work anyway. If a universal algebra for reasoning had come into existence, would, for instance, Leibniz have been able to avoid his big arguments with Newton? Not likely. People don't actually care all that much about logic, not even Leibniz. We just pretend to like logic when it happens to be on our side — otherwise we very often abandon logic and turn to emotional appeals.

This said, there's a powerful attraction to Leibniz's dream. People like the idea of finding an ultimate set of rules to decide everything. Physicists, for instance, dream of a Theory of Everything. At a less exalted level, newspapers and TV are filled with miracle diets — simple rules for regulating your weight as easily as turning a knob on a radio. On the ethical front, each religion has its own compact set of central teachings. And books meant to help their readers lead happier lives offer a simple list of rules to follow.

But, as I hinted above, achieving Leibniz's dream is logically impossible.

In order to truly refute Leibniz's dream, we need to find a precise way to formulate it. As it happens, formal versions of Leibniz's dream were first developed early in the Twentieth century.

An early milestone occurred in 1910, when the philosophers Bertrand Russell and Alfred North Whitehead published their monumental *Principia Mathematica*, intended to provide a formal logical system that could account for all of mathematics. And, as we'll be discussing below, hand in hand with the notion of a formal system came an exact description of what is meant by a logical proof.

There were some problems with the Russell-Whitehead system, but by 1920, the mathematician David Hilbert was confident enough to propose what came to be known as *Hilbert's program*.

(1) We will discover a complete formal system, capable of deciding all the questions of mathematics.

(2) We will prove that this system is free of any possible contradiction.

As Hilbert put it, "The conviction of the solvability of every mathematical problem is a powerful incentive to the worker. We hear within us the perpetual call: There is the problem. Seek its solution. You can find it by pure reason, for in mathematics there is no *ignorabimus*."

For a decade, scientists could dream that Hilbert's program might come true. And meanwhile mathematics and much of physics were being recast as formal systems. Scientific theories could now be viewed as deterministic processes for determining the truth of theorems. Leibniz's dream was nearly at hand!

But, then, in 1931, the logician Kurt Gödel proved his celebrated Incompleteness Theorem.

Gödel's Incompleteness Theorem. If F is a consistent formal system as powerful as arithmetic, then there are infinitely many sentences which are undecidable for F.

This means there can never be formal system of mathematics of the kind sought by Hilbert's program. Every formal system F about mathematics is incomplete in the sense that there are sentences G such that F fails to prove G or ~G, where ~G is the negation of G.

Gödel's sentences G take the form of statements that certain algebraic formulas have no solutions in the natural numbers. Normally these sentences include at least one very large numerical parameter that in some sense codes up the entire theory F. Wolfram (2002, p. 790) has suggested that there might be some much simpler undecidable Gödelian sentences, and proposes that the following sentence might be undecidable: "For all m and n, $m^2 \neq n^5 + 6n + 3$."

Philosophers of science have wondered if there is something like an Incompleteness Theorem for theories about the natural world. One somewhat awkward approach might be to argue that if the natural world happens to be infinite, then we can in some sense represent the system of natural numbers as a list of objects within the world and then go on to claim that the usual undecidable Gödel statements about arithmetic are also statements about the natural world.

But, as I discuss in (Rucker, 1982, p. 290), this isn't a satisfying approach. If we wanted to have number theory be a subset of a theory W about the physical world, we'd need for W to single out an infinite set of objects to play the role of the numbers, and W would also need to define relations the correspond to numerical addition and multiplication.

What we really want is a proof—or at least a plausibility argument—for a Natural Incompleteness Theorem that asserts the existence of undecidable sentences that are about natural physical processes—as opposed to being about the natural numbers in disguise.

Wolfram's analysis of computation in *A New Kind of Science* opens a path. The first step is to accept the idea that natural processes can be thought of as computations. And the second step is to argue for some form of Wolfram's Principle of Computational Equivalence.

Wolfram's Principle of Computational Equivalence (PCE): Almost all processes that are not obviously simple can be viewed as computations of equivalent sophistication.

In this essay I'll show that, starting from Wolfram's two steps, we can prove a Natural Incompleteness Theorem. My method will be to make use of Alan Turing's 1936 work on what he called unsolvable halting problems. And rather than using the full strength of Wolfram's somewhat controversial Principle of Computational Equivalence, I'll base my argument on a weaker assumption, which I call the Halting Problem Hypothesis. And we'll end up with the following Natural Incompleteness Theorem.

Natural Incompleteness Theorem. For most naturally occurring complex processes and for any correct formal system for science, there will be sentences about the process which are undecidable by the given formal system.

This is, I believe, a clean statement of new result—and may be of real importance to the philosophy of science. Although Wolfram (2002, p. 1138) gives some specific examples of undecidable statements about natural processes, he fails to state the general Natural Incompleteness Theorem.

The Halting Problem Hypothesis

It's traditional to ask if a computation comes to an end, or if it halts. We can extend our language a bit and speak of a natural process as halting it happens to reach or to pass through some particular designated state. The established results about the narrow sense of halting apply to this generalized sense as well.

In many situations we value processes that halt in our more general sense. Suppose you feed a set of equations into some computer algebra software, and that you ask the software to solve the equations. What you want is for the resulting process to halt in the sense of displaying an answer on the screen. It doesn't halt in the more dramatic and narrow sense of going dead or freezing up the machine.

In many situations, we like to have computations or processes that *don't* halt. When we simulate, say, the life of some artificially alive creature, or the evolution of a species, we aren't aiming towards a specific kind of result, and still less do we want to see a fixed state or periodic behavior. In this situation we prefer a non-halting computation that continues to produce novel effects.

The distinction between halting and not halting leads to Turing's Theorem of 1936**.**

Definition. The computation P is said to have a *solvable halting problem* if and only if there is an algorithm for deciding in advance which inputs will cause P eventually to reach a halted target state, and which inputs will cause P to run endlessly without ever reaching a halted target state.

Definition. A computation is *universal* if it can emulate any other computation.

Emulating a particular computation C means that you can feed a certain code into your universal computation U that will cause U to produce the same input-output behavior as C.

As it happens, universal computations are in fact very common. Any personal computer, for instance, embodies a universal computation. Indeed, even as simple a computation as the one-dimensional cellular automaton with rule-code 110 is universal (Wolfram, 2002).

Putting all our new concepts together, we arrive at the following.

Turing's Theorem. If U is a universal computation, and then U has an unsolvable halting problem.

This means that if a computation is of a sufficiently rich and general nature, then there is no simple algorithm for predicting which inputs will make U run forever, and which inputs will make U end up in some desired target state, such as the state of coming to a halt.

Let's switch focus now, and discuss how the notion of halting problems can be used to formulate a weaker form of Wolfram's Principle of Computational Equivalence. For convenience, here is a statement of the PCE again.

Wolfram's Principle of Computational Equivalence (PCE): Almost all processes that are not obviously simple can be viewed as computations of equivalent sophistication.

I'll now ring the PCE through three changes, hit a snag, formulate an alternate form of the PCE, and then suggest a still-weaker hypothesis that I'll call the Halting Problem Hypothesis (HPH).

Suppose that we speak of computations rather than processes, and that we speak of computations that are "complex" rather than "not obviously simple." In this case the PCE becomes:

(1) *Almost all complex computations are of equivalent sophistication.*

What might Wolfram mean by saying that two computations are "of equivalent sophistication"? Suppose we take this to mean that the computations can emulate each other or that, more technically, they have the same degree of unsolvability. So now the PCE becomes:

(2) *Almost all complex computations can emulate each other.*

Now certainly Turing's universal computation is complex. So, given that a computation which emulates a universal computation is itself universal, the PCE becomes:

(3) *Almost all complex computations are universal.*

But mathematical logicians have proved:

(Snag) *There are very many complex computations which are not universal*.

The "almost all" in the PCE gives us some wiggle room.[iii] But at this point we'd do well to back off. Suppose we weaken the range of application of the PCE. Rather than saying it applies to "almost all" complex computations, suppose we say it applies to "Most naturally occurring" complex computations. And this gives us a weakened formulation of the PCE.

(4) Most naturally occurring complex computations are universal.

This statement may still be too strong. Rather than insisting upon it, let's consider what we plan to use the PCE *for*. As I mentioned in the introductory section, I plan to use something like the PCE as a stepping stone to a Natural Incompleteness Theorem. And for this, all I need is the following *Halting Problem Hypothesis (HPH)*.

(HPH) Halting Problem Hypothesis: Most naturally occurring complex computations have unsolvable halting problems relative to some simple notion of halting.

Think of a computation as an ongoing process, for example your life, or society, or a plant growing, or the weather. As I mentioned in the previous section, relative to a given computation we can formulate the notion of a *target state* as being some special status or behavior that the computation might eventually reach. The *halting problem* in this context is the problem of deciding whether a given input will eventually send your computation into one of the target states. And, once again, a halting problem is *unsolvable* if there's no computation, algorithm, or rule-of-thumb to detect which inputs won't ever produce one of these specified target state.

The HPH says that if you have some naturally occurring computation that isn’t obviously simple, then there will probably be some simple notion of a target state that leads to an unsolvable halting problem.

Note that the PCE implies the HPH. Going in the other direction, the HPH does *not* imply the PCE. The HPH claims only that certain computations have unsolvable halting problems, and does *not* claim that these computations are universal. The good thing about the HPH is that, unlike the PCE, the HPH has no difficulties with the many non-universal computations that have unsolvable halting problems. The HPH has a better chance of being true, and is easier to defend against those who doubt the validity of Wolfram’s analysis of computation.

It’s worth noting that it may be possible to drop the two-fold qualifier “mast naturally occurring” from the HPH and to get a Strong Halting Problem Hypothesis as stated below.

Strong Halting Problem Hypothesis: All complex computations have unsolvable halting problems relative to some notion of halting.

This says that *all* complex computations have associated with them some unsolvable halting problem. If this is indeed the case, then the Strong Halting Problem

Hypothesis clarifies what we mean by a "complex computation."

Getting back to the weaker HPH, let me clarify its import by giving some fanciful examples. The table below lists a variety of real world computations. In each row, I suggest a computation, a notion of "target state", and a relevant question that has the form of a halting problem—where we try to detect initial states that produce endlessly running computations that never reach the specified target state. (I'm idealizing here, and temporarily setting aside the issue that none of the physical processes that I mention can in fact run for infinitely many years.)

Assuming that the HPH applies to these computations with these particular definitions of target state, we're faced with unsolvability, which means that none of the questions in the third column can be third column can be answered by a finding a simple way to detect which inputs will set off a process that never reaches the target states.

Computation	**Target States**	**Unsolvable Halting Problem**
The motions of the bodies in our solar system.	Something rams into Earth.	Which possible adjustments to Earth's orbit can make us safe forever?
The evolution of our species as we spread from world to world.	Extinction.	Which possible tweaks to our genetics might allow our race survive indefinitely?
The growth and aging of your body.	Developing cancer.	Which people will never get cancer?
Economics and finance.	Becoming wealthy.	Which people will never get rich?
Crime and punishment.	Going to jail.	Which kinds of careers allow a person to avoid incarceration forever?
Writing a book.	It's obviously finished.	Which projects are doomed from the outset never to be finished?
Working to improve one's mental outlook.	Serenity, tranquility, peace.	When is a person definitely on the wrong path?
Finding a mate.	Knowing that *this* is the one.	Who is doomed never to find true love?
Inventing something.	*Eureka!*	Which research programs are utterly hopeless?

Table 1: Unsolvable Halting Problems In Everyday Life.

A Natural Incompleteness Theorem

Let's begin by defining what I mean by a formal system. A *formal system* F can be characterized as having four components: A set of symbols, a rule for recognizing which finite strings of symbols are grammatical sentences, a rule for deciding which sentences are to be regarded as the axioms of the system, and some inference rules for deducing sentences from other sentences.

A *proof* of a sentence S from the formal system F is a sequence of sentences, with the last sentence of the sequence being the targeted sentence S. Each preceding sentence must either be an axiom or be a sentence which is arrived at by combining still earlier sentences according to the inference rules. If a sentence is provable from F, we call it a *theorem* of F.

Combined with the notion of proof, a formal system becomes the source of a potentially endless number of theorems. Aided by a formal system, we mentally reach out into the unknown and produce facts about entirely new situations.

Now let's think of a formal system as a computation. There are several ways one might do this, but what's going to be most useful here is to work with a computation *FProvable* that captures the key aspect of a formal system: it finds theorems. Our *FProvable* will try to detect — so far as possible — which strings of symbols are theorems of F. That is, for any proposed provable sentence S, the computation *FProvable*(S) will carry out the following computation.

(1) If S fails to be a grammatical sentence *FProvable(S)* returns False.

(2) Otherwise *FProvable* starts mechanically generating proofs from the formal system F in order of proof size, and if S appears at the end of a proof, *FProvable(S)* returns True.

(3) If S is a grammatical sentence but no proof of S is ever found, then *FProvable(S)* fails to halt.

As it turns out, if F is a powerful enough formal system to prove the basic facts of arithmetic, then FProvable will be universal. And then, by Turing's Theorem, FProvable has an unsolvable halting problem.[iv]

Let's come back to Leibniz's dream. Suppose we could formulate some wonderfully rich and inclusive formal system F that includes mathematics, physics, biology, human psychology, and even the laws of human society. And then, just as Leibniz said, whenever we're asked if some statement S about the world were true, we'd set the computation FProvable(S) in motion, and the computation would eventually return

True — provided that S is provable as well as true.

One cloud on the horizon is that, if S isn't provable, then FProvable(S) is going to run forever. And, due to the unsolvability of the halting problem, there's no way to filter out in advance those sentences S that are in fact unprovable sentences.

To delve deeper, we need two more definitions. As a I mentioned before, we'll use ~ to represent negation. So if S is a sentence, ~S means "not S". That is, S is false if and only if ~S is true. Using this notion of negation, we can formulate the notion of consistency.

Definition. F is consistent if and only if there is no sentence S such that F proves S and F proves ~S.

According to the usual rules of logic, if a theory proves even one contradiction, then it will go ahead and prove everything possible. So an inconsistent theory is useless for distinguishing between true and false statements about the world. We can reasonably suppose that our proposed Leibniz's-dream-type theory F is consistent.

What if *neither* S *nor* ~S are provable from F? As it turns out, the neither-nor case *does* happen. A lot! The reason has to do with, once again, the unsovability of the halting problem for FProvable.

Definition. If F is a formal system and S is a particular statement such that F proves neither S nor ~S, we say *S is undecidable for* F.

A priori, we can see that there are four possible situations regarding the behavior of the "Is S provable?" computation.

	FProvable(~S) returns True	**FProvable(~S) doesn't halt.**
FProvable(S) returns True.	F proves both S and ~S, meaning F is inconsistent.	F proves S.
FProvable(S) doesn't halt.	F proves ~S.	F proves neither S nor ~S,

		meaning that S is undecidable for F.

Table 2: Four Kinds of Provability and Unprovability

In their optimism, the early mathematical logicians such as David Hilbert hoped to find a formal system F such that the undecidable and inconsistent cases would never arise. As I mentioned earlier, Hilbert's program proposed finding a provably consistent formal system F that could decide all mathematical questions. But Hilbert's hopes were in vain. For we have *Gödel's Incompleteness Theorem*, which tells us that any formal system designed along the lines of Leibniz's dream or Hilbert's program will leave infinitely many sentences undecidable.

Gödel's Incompleteness Theorem. If F is a consistent formal system as powerful as arithmetic, then there are infinitely many sentences which are undecidable for F.

What are these undecidable sentences like? As I mentioned in the introduction, one simple kind of undecidable sentence, call it G, might be characterized in terms of some algebraic property g[n] that a number n might have. It might look like this, where g[n] can be thought of as being a simple algebraic formula with the parameter n:

(G) For all n, g[n] isn't true.

It's interesting, though a bit dizzying, to compare and contrast two related ways of talking about a sentence S. On the one hand, we can ask if S is true or false in the real world of numbers, and on the other hand we can ask if S or ~S happens to be provable from F . In the case where the sentence G has the form mentioned above, only three possibilities can occur. In order to illuminate the notion of undecidability, let's take a quick look at the three case.

(1) G is false, and ~G is provable. If G is false, his means there is a specific n such that g[n] holds in the world of numbers. F will be able to prove the instance g[n] simply by checking the arithmetic. Therefore, F will be able to prove ~G.

(2) G is true, and G is provable. If the G sentence is true in the world of numbers, then g[n] is false for every n. Now in some situations, there may be a clever proof of this general fact from F. I call such a proof "clever" because it somehow has to prove in a finite number of symbols that that g[n] is impossible for *every* n. A general proof doesn't get bogged down at looking at every possible value of n. It has to use some kind of tricky reasoning to cover infinitely many cases at once.

(3) *G is true, and G is not provable.* In these cases, there is no clever proof. The only way F could prove G would be to look at every possible number n and show that g[n] isn't true — but this would take forever. In a case like this it's almost as if G only *happens* to be true. At least as far as F can see, there's no overarching reason *why* g[n] is impossible for every n. It's just that, as chance would have it, in the real world there *aren't* any such n. And thus G is undecidable by F .

The computer scientist Gregory Chaitin suggests that in a case like the third, we think of G as a *random truth*. It's not true for any deep, theoretical reason. It's just something that turns out to be so.[v]

Note that there's an endless supply of undecidable sentences S beyond the simple kinds of sentences G that I've been discussing . Some initial examples of the next level of complexity might be "For each m there is an n such that g[m, n]" or "There is an m such that for all n, g[m, n]."

Most mathematicians would feel that, in the real world of mathematics, any of these sentences is definitely true or false, regardless of F's inability to prove either of the alternatives. So the undecidable statements are "random" truths about the mathematical world, brute facts that hold for no particular reason.

But so far, we've only been talking about number theory. How do we get to undecidable sentences about the *natural* world? If we accept the HPH, and we assume that any natural process can be regarded as a computation, then we can find undecidability in any complex natural process!

The path leads through the following lemma, proved by Turing in 1936.

Unsolvability and Undecidability Lemma. If P is a computation with an unsolvable halting problem, and F is a correct formal theory, then there will be infinitely many sentences about P which are undecidable for F.

In this Lemma, by the way, I'm using the phrase "correct formal theory" to mean a formal theory that doesn't prove things which are false. I won't go into the somewhat technical details of the proof of this lemma, but the general idea is that there have to be lots of sentences about P that are undecidable for F, for otherwise F could solve P's unsolvable halting problem.

So now we come to the pay-off. Naturally occurring processes can be thought of as computations. If we accept the Halting Problem Hypothesis, then each naturally occurring process will have an unsolvable halting problem. And then, by applying Turing's Unsolvability and Undecidability Lemma, we get the following.

Natural Incompleteness Theorem. For most naturally occurring complex processes, and any correct formal system for science, there will be sentences about the process that are undecidable by the given formal system.

What makes the Natural Incompleteness Theorem attractive is that the undecidable sentences are *not* just about arithmetic. They're about the behavior of actual real-world processes.

No matter how thoroughly you try and figure the world out, there are infinitely many things you can't prove. Here are some examples of potentially undecidable sentences. Each of them may be, in principle, true or false, but only in a random kind of way, in that they're not proved or disproved by any of our formal theories about the world.

Nobody will ever manage to bounce a golf ball a thousand times in a row off a putter head.
There are an endless number of planets in our universe.
There are an endless number of planets with people indistinguishable from you.
No human will ever be born with six functioning arms.
No cow's spots will ever spell out your first name in big puffy letters.

Every year with a big birth rate increase is followed by a big war.
The left wing will dominate American politics more often than the right wing does.
Mankind will evolve into higher forms of life.
The majority of times that you move to a different line in a supermarket, the new line goes slower than one of the lines you didn't pick.
New races of intelligent beings will emerge over and over for the rest of the time.
The time of our cosmos extends forever.

Table 3: Potentially Undecidable Statements about the Natural World.

Do note that, as with our examples about natural halting problems, we need some analysis of how to take into account the issue that so few of our natural systems can in fact be viewed as potentially eternal. But I'll leave the fine points of issue for other investigators to work out.

Undecidability Everywhere

It often happens in the history of science that some odd-ball new category is discovered. At first nobody's sure if any phenomena of this kind exist, but then there's some kind of logical argument why these odd-ball things have to occur. And then, as time goes on, more and more of the curious entities are discovered until finally they're perceived to be quite run of the mill. And I think this is what will happen with the notion of undecidable sentences about the natural world.

To dramatize this notion, I'll present a sustained analogy between the spread of undecidability and the rise of transcendental numbers in mathematics. Brian Silverman suggested this analogy to me in an email.

Transcendental Numbers. 300 BC. The Greeks worked primarily with real numbers that can be expressed either as the fraction of two whole numbers, or which can be obtained by the process of taking square roots. By the time of the Renaissance, mathematicians had learned to work with roots of all kinds, that is, with the full class of algebraic numbers — where an algebraic number can be expressed as the solution to some polynomial algebraic equation formulated in terms of whole numbers. The non-algebraic numbers were dubbed the *transcendental* numbers. And, for a time, nobody was sure if any transcendental numbers existed.

Undecidable Sentences. 1920. In David Hilbert's time, it seemed possible that, at least in mathematics, every problem could be decided on the basis of a reasonable formal system. This was the inspiration for Hilbert's program.

Transcendental Numbers. 1884. The first constructions of transcendental real numbers were carried out by Joseph Liouville. Liouville's numbers were, however, quite artificial, such as the so-called Liouvillian number 0.1100010000000000000000010000... which has a 1 in the decimal positions n! and 0 in all the other places. Someone might readily say that a number like this is unlikely to occur in any real context. (n! stands for "n factorial" which is the product 1*2*...*n of all the integers from 1 to n.)

Undecidable Sentences. 1931. Kurt Gödel proved the existence of some particular undecidable algebraic sentences. These sentences were somewhat unnatural. Relative to a given formal system F, they had the form "This sentence is not provable from F," or the alternate form, "The contradiction 0 = 1 is not provable from the formal system F."

Transcendental Numbers. 1874. Georg Cantor developed his set theory, and showed there are an infinite number of transcendental numbers. Someone could say that Cantor's transcendental numbers aren't numbers that would naturally occur, that they are artificial, and that they depend in an essential way upon higher-order concepts such as treating an infinite enumeration of reals as a completed object.

Undecidable Sentences. 1936. Building on Gödel's work, Alan Turing proved his theorem on the unsolvability of the halting problem. He immediately derived the corollary that there are infinitely many undecidable sentences of mathematics, and that these sentences came in quite arbitrary forms. Even so, the specific examples of such sentences that he could give were still odd and somewhat self-referential, like Gödel's undecidable sentences.

Transcendental Numbers. 1873. Charles Hermite proved that the relatively non-artificial number e is transcendental.

Undecidable Sentences. 1965. On an entirely different front, Paul J. Cohen proved that an important question about infinite sets called the continuum hypothesis is undecidable from the known axioms of mathematics. (Cohen's proof built on an earlier result proved by Kurt Gödel in 1946.) 1970. Back in the realm of unsolvable halting problems, Julia Robinson, Martin Davis, Yuri Matiyasevich showed that among the sentences undecidable for any formal theory we'll find an infinite number of polynomial Diophantine equations which don't have any whole number solutions, but for which we can't prove this fact. This means there a very large range of ordinary mathematical sentences which are undecidable.

Transcendental Numbers. 1882. Ferdinand Lindemann proved that the garden variety number pi is transcendental.

Undecidable Sentences. 2002. Wolfram pointed out that we should be able to find numerous examples of undecidability in the natural world.

And now we have a Natural Incompleteness Theorem telling us that *every* possible complex natural process is going to have undecidable sentences associated with it! Undecidability is everywhere, and all of our theories about nature must remain incomplete.

References

Chaitin, G. J., 1999. *The Unknowable*. New York: Springer.

Leibniz, G.W. and Gerhardt, C.I. (ed.), 1978. *Die philosophischen Schriften von Gottfried Wilhelm Leibniz*. Hildesheim: Georg Olms Verlag.

Rucker, R., 1982. *Infinity and the Mind*. Boston: Birkhäuser.

Rucker, R., 2005. *The Lifebox, the Seashell, and the Soul*. New York: Thunder's Mouth Press.

Wolfram, S., 2002. *A New Kind of Science*. Champaign: Wolfram Media.

---

[i] This paper is adapted from material in Rucker (2005). Formal details about my argument can be found in this book's appendix. Note that in the book, I used a somewhat unfortunate terminology. I gave the *Halting Problem Hypothesis* an imprecise name: the *Natural Undecidability Hypothesis*. And my *Natural Incompleteness Theorem* has too undramatic a name: the *Principle of Natural Undecidability*.

[ii] The quote is from Leibniz and Gerhardt (1978), volume VII, p. 200. The passage is translated by the British philosopher George MacDonald Ross and can be

found on his website.

[iii] When Wolfram formulated his PCE, he was well aware of the problem that there are infinitely many degrees of unsolvability. Therefore he phrased his PCE so that it has two loopholes. (Wolfram 2002, p. 734 and pp. 1130-1131.)

The loopholes are to be found in, respectively, the very first and very last phrases of the PCE, which I italicize here: *Almost all* processes that are not obviously simple can be viewed as computations *of equivalent sophistication.*

Regarding the first loophole, Wolfram is saying that complex non-universal Turing machines "almost never" occur in natural contexts. This is an interesting aspect of the PCE, in that it seems to say something about the kinds of processes that actually occur in the real world.

Keep in mind that Wolfram's work is *empirical*. Unlike physical experiments, computer science experiments are exactly reproducible, and thus have a touch of the mathematical or theoretical. But really his inspiration came from looking at exceedingly many computations in action. And to reduce experimenter bias, he made a point of conducting *exhaustive* surveys of various classes of rudimentary computations such as Turing machines and cellular automata.

To exploit the second loophole we might interpret "computations of equivalent sophistication" more broadly than "computations that can emulate each other." Wolfram feels that the processes by which logicians construct complex but non-universal computations have always depended so essentially on the use of an underlying universal computation that the constructed computations are in some as-yet-to-be-defined-sense "as sophisticated as" the universal computations.

Now, so far as I know, all the existing constructions of these complex non-universal computations *do* use a universal computation. But it seems capricious to conclude that therefore *every* complex non-universal computation in some way relies upon a construction involving a universal Turing machine.

Indeed it seems plausible that there may in fact be naturally occurring processes of intermediate degree. It's tempting to speculate that the one-dimensional CA Rule 30 itself is such a computation. And in this case, the PCE would be false, but the HPH that I describe would be true.

[iv] Turing's work showed that arithmetic is strong enough to emulate the running of Turing machines. More specifically, he showed that for any F as strong as arithmetic, we can set things up so that FProvable emulates M. Since we can do this for any machine M, this means that FProvable is a universal computation, so Turing's Theorem applies, and FProvable has an unsolvable halting problem.

[v] You can find more details in (Chaitin, 1999), and in the papers on Chaitin's home page.