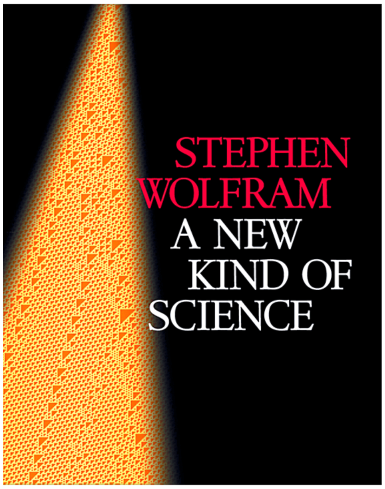
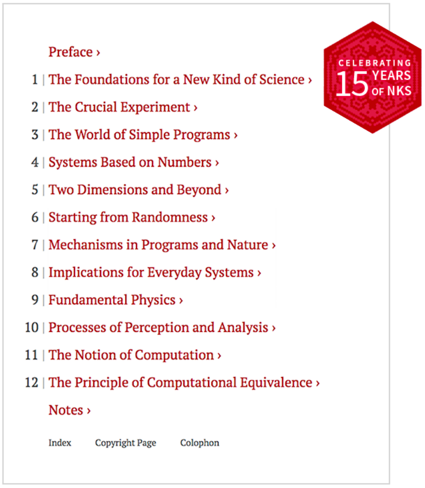
A New Kind of Science: A 15-Year View

Stephen Wolfram
Founder and CEO
Wolfram Research, Inc.
s.wolfram@wolfram.com

Starting now, in celebration of its 15th anniversary, *A New Kind of Science* will be freely available in its entirety, with high-resolution images, on the web or for download.





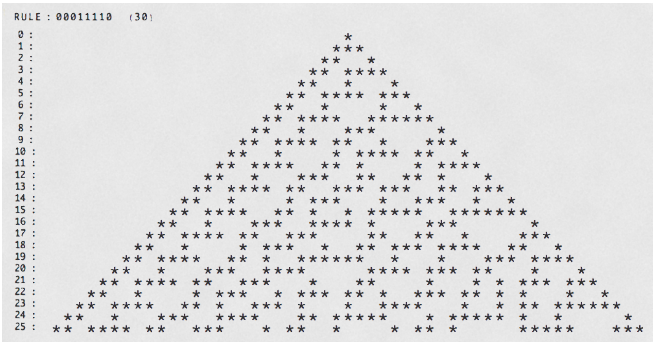
It's now 15 years since I published my book *A New Kind of Science*—more than 25 since I started writing it, and more than 35 since I started working towards it. But with every passing year I feel I understand more about what the book is really about—and why it's important. I wrote the book, as its title suggests, to contribute to the progress of science. But as the years have gone by, I've realized that the core of what's in the book actually goes far beyond science—into many areas that will be increasingly important in defining our whole future.

So, viewed from a distance of 15 years, what is the book really about? At its core, it's about something profoundly abstract: the theory of all possible theories, or the universe of all possible universes. But for me one of the achievements of the book is the realization that one can explore such fundamental things concretely—by doing actual experiments in the computational universe of possible programs. And in the end the book is full of what might at first seem like quite alien pictures made just by running very simple such programs.

Back in 1980, when I made my living as a theoretical physicist, if you'd asked me what I thought simple programs would do, I expect I would have said "not much". I had been very interested in the kind of complexity one sees in nature, but I thought—like a typical reductionistic scientist—that the key to understanding it must lie in figuring out detailed features of the underlying component parts.

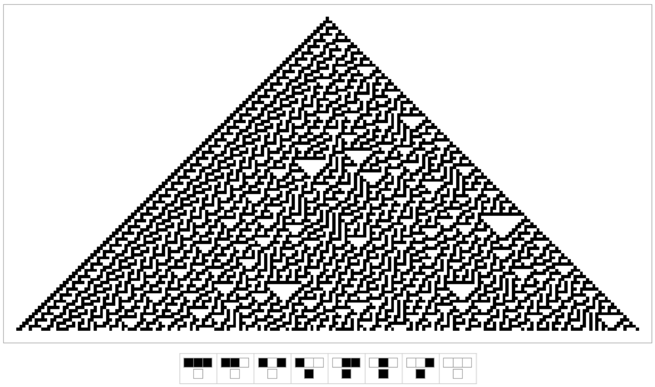
In retrospect I consider it incredibly lucky that all those years ago I happened to have the right interests and the right skills to actually try what is in a sense the most basic experiment in the computational universe: to systematically take a sequence of the simplest possible programs, and run them.

I could tell as soon as I did this that there were interesting things going on, but it took a couple more years before I began to really appreciate the force of what I'd seen. For me it all started with one picture:



```
RULE : 00011110  (30)
 0 :                          *
 1 :                         ***
 2 :                        **  *
 3 :                       ** ****
 4 :                      **  *   *
 5 :                     ** **** ***
 6 :                    **  *    *  *
 7 :                   ** ****  ******
 8 :                  **  *   ***     *
 9 :                 ** **** **  *   ***
10 :                **  *    * **** **  *
11 :               ** ****  ** *    * ****
12 :              **  *   ***  **  ** *   *
13 :             ** **** **  *** ***  ** ***
14 :            **  *    * ***   *  ***  *  *
15 :           ** ****  ** *  * *****  *******
16 :          **  *   ***  **** *    ***      *
17 :         ** **** **  ***    **  **  *    ***
18 :        **  *    * ***  *  ** *** ****  **  *
19 :       ** ****  ** *  ******  *   *   *** ****
20 :      **  *   ***  ****     **** *** **   *   *
21 :     ** **** **  ***   *   **    *   * * *** ***
22 :    **  *    * ***  * *** ** *  *** ** * *   *  *
23 :   ** ****  ** *  *** *   *  ****   *  * ** ******
24 :  **  *   ***  ****   ** *****   * ***** *  *     *
25 : ** **** **  ***   * **  *    * ** *     *****   ***
```

Or, in modern form:



I call it rule 30. It's my all-time favorite discovery, and today I carry it around everywhere on my business cards. What is it? It's one of the simplest programs one can imagine. It operates on rows of black and white cells, starting from a single black cell, and then repeatedly applies the rules at the bottom. And the crucial point is that even though those rules are by any measure extremely simple, the pattern that emerges is not.

It's a crucial—and utterly unexpected—feature of the computational universe: that even among the very simplest programs, it's easy to get immensely complex behavior. It took me a solid decade to understand just how broad this phenomenon is. It doesn't just happen in programs ("cellular automata") like rule 30. It basically shows up whenever you start enumerating possible rules or possible programs whose behavior isn't obviously trivial.

Similar phenomena had actually been seen for centuries in things like the digits of pi and the distribution of primes—but they were basically just viewed as curiosities, and not as signs of something profoundly important. It's been nearly 35 years since I first saw what happens in rule 30, and with every passing year I feel I come to understand more clearly and deeply what its significance is.

Four centuries ago it was the discovery of the moons of Jupiter and their regularities that sowed the seeds for modern exact science, and for the modern scientific approach to thinking. Could my little rule 30 now be the seed for another such intellectual revolution, and a new way of thinking about everything?

In some ways I might personally prefer not to take responsibility for shepherding such ideas ("paradigm shifts" are hard and thankless work). And certainly for years I have just quietly used such ideas to develop technology and my own thinking. But as computation and AI become increasingly central to our world, I think it's important that the implications of what's out there in the computational universe be more widely understood.

Implications of the Computational Universe

Here's the way I see it today. From observing the moons of Jupiter we came away with the idea that—if looked at right—the universe is an ordered and regular place, that we can ultimately understand. But now, in exploring the computational universe, we quickly come upon things like rule 30 where even the simplest rules seem to lead to irreducibly complex behavior.

One of the big ideas of *A New Kind of Science* is what I call the Principle of Computational Equivalence. The first step is to think of every process—whether it's happening with black and white squares, or in physics, or inside our brains—as a computation that somehow transforms input to output. What the Principle of Computational Equivalence says is that above an extremely low threshold, all processes correspond to computations of equivalent sophistication.

It might not be true. It might be that something like rule 30 corresponds to a fundamentally simpler computation than the fluid dynamics of a hurricane, or the processes in my brain as I write this. But what the Principle of Computational Equivalence says is that in fact all these things are computationally equivalent.

It's a very important statement, with many deep implications. For one thing, it implies what I call computational irreducibility. If something like rule 30 is doing a computation just as sophisticated as our brains or our mathematics, then there's no way we can "outrun" it: to figure out what it will do, we have to do an irreducible amount of computation, effectively tracing each of its steps.

The mathematical tradition in exact science has emphasized the idea of predicting the behavior of systems by doing things like solving mathematical equations. But what computational irreducibility implies is that out in the computational universe that often won't work, and instead the only way forward is just to explicitly run a computation to simulate the behavior of the system.

A Shift in Looking at the World

One of the things I did in *A New Kind of Science* was to show how simple programs can serve as models for the essential features of all sorts of physical, biological and other systems. Back when the book appeared, some people were skeptical about this. And indeed at that time there was a 300-year unbroken tradition that serious models in science should be based on mathematical equations.

But in the past 15 years something remarkable has happened. For now, when new models are created—whether of animal patterns or web browsing behavior—they are overwhelmingly more often based on programs than on mathematical equations.

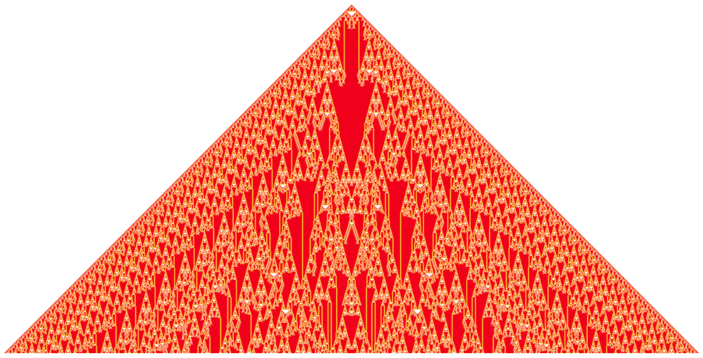
Year by year, it's been a slow, almost silent, process. But by this point, it's a dramatic shift. Three centuries ago pure philosophical reasoning was supplanted by mathematical equations. Now in these few short years, equations have been largely supplanted by programs. For now, it's mostly been something practical and pragmatic: the models work better, and are more useful.

But when it comes to understanding the foundations of what's going on, one's led not to things like mathematical theorems and calculus, but instead to ideas like the Principle of Computational Equivalence. Traditional mathematics-based ways of thinking have made concepts like force and momentum ubiquitous in the way we talk about the world. But now as we think in fundamentally computational terms we have to start talking in terms of concepts like undecidability and computational irreducibility.

Will some type of tumor always stop growing in some particular model? It might be undecidable. Is there a way to work out how a weather system will develop? It might be computationally irreducible.

These concepts are pretty important when it comes to understanding not only what can and cannot be modeled, but also what can and cannot be controlled in the world. Computational irreducibility in economics is going to limit what can be globally controlled. Computational irreducibility in biology is going to limit how generally effective therapies can be—and make highly personalized medicine a fundamental necessity.

And through ideas like the Principle of Computational Equivalence we can start to discuss just what it is that allows nature—seemingly so effortlessly—to generate so much that seems so complex to us. Or how even deterministic underlying rules can lead to computationally irreducible behavior that for all practical purposes can seem to show "free will".



Mining the Computational Universe

A central lesson of *A New Kind of Science* is that there's a lot of incredible richness out there in the computational universe. And one reason that's important is that it means that there's a lot of incredible stuff out there for us to "mine" and harness for our purposes.

Want to automatically make an interesting custom piece of art? Just start looking at simple programs and automatically pick out one you like—as in our WolframTones music site from a decade ago.

Want to find an optimal algorithm for something? Just search enough programs out there, and you'll find one.

We've normally been used to creating things by building them up, step by step, with human effort—progressively creating architectural plans, or engineering drawings, or lines of code. But the discovery that there's so much richness so easily accessible in the computational universe suggests a different approach: don't try building anything; just define what you want, and then search for it in the computational universe.

Sometimes it's really easy to find. Like let's say you want to generate apparent randomness. Well, then just enumerate cellular automata (as I did in 1984), and very quickly you come upon rule 30—which turns out to be one of the very best known generators of apparent randomness (look down the center column of cell values, for examples). In other situations you might have to search 100 000 cases (as I did in finding the simplest axiom system for logic, or the simplest universal Turing machine), or you might have to search millions or even trillions of cases. But in the past 25 years, we've had incredible success in just discovering algorithms out there in the computational universe—and we rely on many of them in implementing the Wolfram Language.

At some level it's quite sobering. One finds some tiny program out in the computational universe. One can tell it does what one wants. But when one looks at what it's doing, one doesn't have any real idea how it works. Maybe one can analyze some part—and be struck by how "clever" it is. But there just isn't a way for us to understand the whole thing; it's not something familiar from our usual patterns of thinking.

Of course, we've often had similar experiences before—when we use things from nature. We may notice that some particular substance is a useful drug or a great chemical catalyst, but we may have no idea why. But in doing engineering and in most of our modern efforts to build technology, the great emphasis has instead been on constructing things whose design and operation we can readily understand.

In the past we might have thought that was enough. But what our explorations of the computational universe show is that it's not: selecting only things whose operation we can readily understand misses most of the immense power and richness that's out there in the computational universe.

A World of Discovered Technology

What will the world look like when more of what we have is mined from the computational universe? Today the environment we build for ourselves is dominated by things like simple shapes and repetitive processes. But the more we use what's out there in the computational universe, the less regular things will look. Sometimes they may look a bit "organic", or like what we see in nature (since after all, nature follows similar kinds of rules). But sometimes they may look quite random, until perhaps suddenly and incomprehensibly they achieve something we recognize.

For several millennia we as a civilization have been on a path to understand more about what happens in our world—whether by using science to decode nature, or by creating our own environment through technology. But to use more of the richness of the computational universe we must at least to some extent forsake this path.

In the past, we somehow counted on the idea that between our brains and the tools we could create we would always have fundamentally greater computational power than the things around us—and as a result we would always be able to "understand" them. But what the Principle of Computational Equivalence says is that this isn't true: out in the computational universe there are lots of things just as powerful as our brains or the tools we build. And as soon as we start using those things, we lose the "edge" we thought we had.

Today we still imagine we can identify discrete "bugs" in programs. But most of what's powerful out there in the computational universe is rife with computational irreducibility—so the only real way to see what it does is just to run it and watch what happens.

We ourselves, as biological systems, are a great example of computation happening at a molecular scale—and we are no doubt rife with computational irreducibility (which is, at some fundamental level, why medicine is hard). I suppose it's a tradeoff: we could limit our technology to consist only of things whose operation we understand. But then we would miss all that richness that's out there in the computational universe. And we wouldn't even be able to match the achievements of our own biology in the technology we create.

Machine Learning and the Neural Net Renaissance

There's a common pattern I've noticed with intellectual fields. They go for decades and perhaps centuries with only incremental growth, and then suddenly, usually as a result of a methodological advance, there's a burst of "hypergrowth" for perhaps five years, in which important new results arrive almost every week.

I was fortunate enough that my own very first field—particle physics—was in its period of hypergrowth right when I was involved in the late 1970s. And for myself, the 1990s felt like a kind of personal period of hypergrowth for what became *A New Kind of Sci-*

ence—and indeed that's why I couldn't pull myself away from it for more than a decade.

But today, the obvious field in hypergrowth is machine learning, or, more specifically, neural nets. It's funny for me to see this. I actually worked on neural nets back in 1981, before I started on cellular automata, and several years before I found rule 30. But I never managed to get neural nets to do anything very interesting—and actually I found them too messy and complicated for the fundamental questions I was concerned with.

And so I "simplified them"—and wound up with cellular automata. (I was also inspired by things like the Ising model in statistical physics, etc.) At the outset, I thought I might have simplified too far, and that my little cellular automata would never do anything interesting. But then I found things like rule 30. And I've been trying to understand its implications ever since.

In building Mathematica and the Wolfram Language, I'd always kept track of neural nets, and occasionally we'd use them in some small way for some algorithm or another. But about five years ago I suddenly started hearing amazing things: that somehow the idea of training neural nets to do sophisticated things was actually working. At first I wasn't sure. But then we started building neural net capabilities in the Wolfram Language, and finally two years ago we released our ImageIdentify.com website—and now we've got our whole symbolic neural net system. And, yes, I'm impressed. There are lots of tasks that had traditionally been viewed as the unique domain of humans, but which now we can routinely do by computer.

But what's actually going on in a neural net? It's not really to do with the brain; that was just the inspiration (though in reality the brain probably works more or less the same way). A neural net is really a sequence of functions that operate on arrays of numbers, with each function typically taking quite a few inputs from around the array. It's not so different from a cellular automaton. Except that in a cellular automaton, one's usually dealing with, say, just 0s and 1s, not arbitrary numbers like 0.735. And instead of taking inputs from all over the place, in a cellular automaton each step takes inputs only from a very well-defined local region.

Now, to be fair, it's pretty common to study "convolutional neural nets", in which the patterns of inputs are very regular, just like in a cellular automaton. And it's becoming clear that having precise (say 32-bit) numbers isn't critical to the operation of neural nets; one can probably make do with just a few bits.

But a big feature of neural nets is that we know how to make them "learn". In particular, they have enough features from traditional mathematics (like involving continuous numbers) that techniques like calculus can be applied to provide strategies to make them incrementally change their parameters to "fit their behavior" to whatever training examples they're given.

It's far from obvious how much computational effort, or how many training examples, will be needed. But the breakthrough of about five years ago was the discovery that for many important practical problems, what's available with modern GPUs and modern web-collected training sets can be enough.

Pretty much nobody ends up explicitly setting or "engineering" the parameters in a neural net. Instead, what happens is that they're found automatically. But unlike with simple programs like cellular automata, where one's typically enumerating all possibilities, in current neural nets there's an incremental process, essentially based on calculus, that manages to progressively improve the net—a little like the way biological evolution progressively improves the "fitness" of an organism.

It's plenty remarkable what comes out from training a neural net in this way, and it's plenty difficult to understand how the neural net does what it does. But in some sense the neural net isn't venturing too far across the computational universe: it's always basically keeping the same basic computational structure, and just changing its behavior by changing parameters.

But to me the success of today's neural nets is a spectacular endorsement of the power of the computational universe, and another validation of the ideas of *A New Kind of Science*. Because it shows that out in the computational universe, away from the constraints of explicitly building systems whose detailed behavior one can foresee, there are immediately all sorts of rich and useful things to be found.

NKS Meets Modern Machine Learning

Is there a way to bring the full power of the computational universe—and the ideas of *A New Kind of Science*—to the kinds of things one does with neural nets? I suspect so. And in fact, as the details become clear, I wouldn't be surprised if exploration of the computational universe saw its own period of hypergrowth: a "mining boom" of perhaps unprecedented proportions.

In current work on neural nets, there's a definite tradeoff one sees. The more what's going on inside the neural net is like a simple mathematical function with essentially arithmetic parameters, the easier it is to use ideas from calculus to train the network. But the more what's going on is like a discrete program, or like a computation whose whole structure can change, the more difficult it is to train the network.

It's worth remembering, though, that the networks we're routinely training now would have looked utterly impractical to train only a few years ago. It's effectively just all those quadrillions of GPU operations that we can throw at the problem that makes training feasible. And I won't be surprised if even quite pedestrian (say, local exhaustive search) techniques will fairly soon let one do significant training even in cases where no incremental numerical approach is possible. And perhaps even it will be possible to invent some major generalization of things like calculus that will operate in the full computational universe. (I have some suspicions, based on thinking about generalizing basic notions of geometry to cover things like cellular automaton rule spaces.)

What would this let one do? Likely it would let one find considerably simpler systems that could achieve particular computational goals. And maybe that would bring within reach some qualitatively new level of operations, perhaps beyond what we're used to being possible with things like brains.

There's a funny thing that's going on with modeling these days. As neural nets become more successful, one begins to wonder: why bother to simulate what's going on inside a system when one can just make a black-box model of its output using a neural net? Well, if we manage to get machine learning to reach deeper into the computational universe, we won't have as much of this tradeoff any more—because we'll be able to learn models of the mechanism as well as the output.

I'm pretty sure that bringing the full computational universe into the purview of machine learning will have spectacular consequences. But it's worth realizing that computational universality—and the Principle of Computational Equivalence—make it less a matter of principle. Because they imply that even neural nets of the kinds we have now are universal, and are capable of emulating anything any other system can do. (In fact, this universality result was essentially what launched the whole modern idea of neural nets, back in 1943.)

And as a practical matter, the fact that current neural net primitives are being built into hardware and so on will make them a desirable foundation for actual technology systems, though, even if they're far from optimal. But my guess is that there are tasks where for the foreseeable future access to the full computational universe will be necessary to make them even vaguely practical.

Finding AI

What will it take to make AI? As a kid, I was very interested in figuring out how to make a computer know things, and be able to answer questions from what it knew. And when I studied neural nets in 1981, it was partly in the context of trying to understand how to build such a system. As it happens, I had just developed SMP, which was a forerunner of Mathematica (and ultimately the Wolfram Language)—and which was very much based on symbolic pattern matching ("if you see this, transform it to that"). At the time, though, I imagined that AI was somehow a "higher level of computation", and I didn't know how to achieve it.

I returned to the problem every so often, and kept putting it off. But then when I was working on *A New Kind of Science* it struck me: if I'm to take the Principle of Computational Equivalence seriously, then there can't be any fundamentally "higher level of computation"—so AI must be achievable just with the standard ideas of computation that I already know.

And it was this realization that got me started building Wolfram|Alpha. And, yes, what I found is that lots of those very "AI-oriented things", like natural language understanding, could be done just with "ordinary computation", without any magic new AI invention. Now, to be fair, part of what was happening was that we were using ideas and methods from *A New Kind of Science*: we weren't just engineering everything; we were often searching the computational universe for rules and algorithms to use.

So what about "general AI"? Well, I think at this point that with the tools and understanding we have, we're in a good position to automate essentially anything we can define. But definition is a more difficult and central issue than we might imagine.

The way I see things at this point is that there's a lot of computation even near at hand in the computational universe. And it's powerful computation. As powerful as anything that happens in our brains. But we don't recognize it as "intelligence" unless it's aligned with our human goals and purposes.

Ever since I was writing *A New Kind of Science*, I've been fond of quoting the aphorism "the weather has a mind of its own". It sounds so animistic and pre-scientific. But what the Principle of Computational Equivalence says is that actually, according to the most modern science, it's true: the fluid dynamics of the weather is the same in its computational sophistication as the electrical processes that go on in our brains.

But is it "intelligent"? When I talk to people about *A New Kind of Science*, and about AI, I'll often get asked when I think we'll achieve "consciousness" in a machine. Life, intelligence, consciousness: they are all concepts that we have a specific example of, here on Earth. But what are they in general? All life on Earth shares RNA and the structure of cell membranes. But surely that's just because all life we know is part of one connected thread of history; it's not that such details are fundamental to the very concept of life.

And so it is with intelligence. We have only one example we're sure of: us humans. (We're not even sure about animals.) But human intelligence as we experience it is deeply entangled with human civilization, human culture and ultimately also human physiology—even though none of these details are presumably relevant in the abstract definition of intelligence.

We might think about extraterrestrial intelligence. But what the Principle of Computational Equivalence implies is that actually there's "alien intelligence" all around us. But somehow it's just not quite aligned with human intelligence. We might look at rule 30, for example, and be able to see that it's doing sophisticated computation, just like our brains. But somehow it just doesn't seem to have any "point" to what it's doing.

We imagine that in doing the things we humans do, we operate with certain goals or purposes. But rule 30, for example, just seems to be doing what it's doing—just following some definite rule. In the end, though, one realizes we're not so very different. After all, there are definite laws of nature that govern our brains. So anything we do is at some level just playing out those laws.

Any process can actually be described either in terms of mechanism ("the stone is moving according to Newton's laws"), or in terms of goals ("the stone is moving so as to minimize potential energy"). The description in terms of mechanism is usually what's most useful in connecting with science. But the description in terms of goals is usually what's most useful in connecting with human intelligence.

And this is crucial in thinking about AI. We know we can have computational systems whose operations are as sophisticated as anything. But can we get them to do things that are aligned with human goals and purposes?

In a sense this is what I now view as the key problem of AI: it's not about achieving underlying computational sophistication, but instead it's about communicating what we want from this computation.

The Importance of Language

I've spent much of my life as a computer language designer—most importantly creating what is now the Wolfram Language. I'd always seen my role as a language designer being to imagine the possible computations people might want to do, then—like a reductionist scientist—trying to "drill down" to find good primitives from which all these computations could be built up. But somehow from *A New Kind of Science*, and from thinking about AI, I've come to think about it a little differently.

Now what I more see myself as doing is making a bridge between our patterns of human thinking, and what the computational universe is capable of. There are all sorts of amazing things that can in principle be done by computation. But what the language does is to provide a way for us humans to express what we want done, or want to achieve—and then to get this actually executed, as automatically as possible.

Language design has to start from what we know and are familiar with. In the Wolfram Language, we name the built-in primitives with English words, leveraging the meanings that those words have acquired. But the Wolfram Language is not like natural language. It's something more structured, and more powerful. It's based on the words and concepts that we're familiar with through the shared corpus of human knowledge. But it gives us a way to build up arbitrarily sophisticated programs that in effect express arbitrarily complex goals.

Yes, the computational universe is capable of remarkable things. But they're not necessarily things that we humans can describe or relate to. But in building the Wolfram Language my goal is to do the best I can in capturing everything we humans want—and being able to express it in executable computational terms.

When we look at the computational universe, it's hard not to be struck by the limitations of what we know how to describe or think about. Modern neural nets provide an interesting example. For the ImageIdentify function of the Wolfram Language we've trained a neural net to identify thousands of kinds of things in the world. And to cater to our human purposes, what the network ultimately does is to describe what it sees in terms of concepts that we can name with words—tables, chairs, elephants, etc.

But internally what the network is doing is to identify a series of features of any object in the world. Is it green? Is it round? And so on. And what happens as the neural network is trained is that it identifies features it finds useful for distinguishing different kinds of things in the world. But the point is that almost none of these features are ones to which we happen to have assigned words in human language.

Out in the computational universe it's possible to find what may be incredibly useful ways to describe things. But they're alien to us humans. They're not something we know how to express, based on the corpus of knowledge our civilization has developed.

Now of course new concepts are being added to the corpus of human knowledge all the time. Back a century ago, if someone saw a nested pattern they wouldn't have any way to describe it. But now we'd just say "it's a fractal". But the problem is that in the computational universe there's an infinite collection of "potentially useful concepts"—with which we can never hope to ultimately keep up.

The Analogy in Mathematics

When I wrote *A New Kind of Science* I viewed it in no small part as an effort to break away from the use of mathematics—at least as a foundation for science. But one of the things I realized is that the ideas in the book also have a lot of implications for pure mathematics itself.

What is mathematics? Well, it's a study of certain abstract kinds of systems, based on things like numbers and geometry. In a sense it's exploring a small corner of the computational universe of all possible abstract systems. But still, plenty has been done in mathematics: indeed, the three million or so published theorems of mathematics represent perhaps the largest single coherent intellectual structure that our species has built.

Ever since Euclid, people have at least notionally imagined that mathematics starts from certain axioms (say, $a + b = b + a$, $a + 0 = a$, etc.), then builds up derivations of theorems. Why is math hard? The answer is fundamentally rooted in the phenomenon of computational irreducibility—which here is manifest in the fact that there's no general way to shortcut the series of steps needed to derive a theorem. In other words, it can be arbitrarily hard to get a result in mathematics. But worse than that—as Gödel's Theorem showed—there can be mathematical statements where there just aren't any finite ways to prove or disprove them from the axioms. And in such cases, the statements just have to be considered "undecidable".

And in a sense what's remarkable about math is that one can usefully do it at all. Because it could be that most mathematical results one cares about would be undecidable. So why doesn't that happen?

Well, if one considers arbitrary abstract systems it happens a lot. Take a typical cellular automaton—or a Turing machine—and ask whether it's true that the system, say, always settles down to periodic behavior regardless of its initial state. Even something as simple as that will often be undecidable.

So why doesn't this happen in mathematics? Maybe there's something special about the particular axioms used in mathematics. And certainly if one thinks they're the ones that uniquely describe science and the world there might be a reason for that. But one of the whole points of the book is that actually there's a whole computational universe of possible rules that can be useful for doing science and describing the world.

And in fact I don't think there's anything abstractly special about the particular axioms that have traditionally been used in mathematics: I think they're just accidents of history.

What about the theorems that people investigate in mathematics? Again, I think there's a strong historical character to them. For all but the most trivial areas of mathematics, there's a whole sea of undecidability out there. But somehow mathematics picks the islands where theorems can actually be proved—often particularly priding itself on places close to the sea of undecidability where the proof can only be done with great effort.

I've been interested in the whole network of published theorems in mathematics (it's a thing to curate, like wars in history, or properties of chemicals). And one of the things I'm curious about is whether something there's an inexorable sequence to the mathematics that's done, or whether, in a sense, random parts are being picked.

And here, I think, there's a considerable analogy to the kind of thing we were discussing before with language. What is a proof? Basically it's a way of explaining to someone why something is true. I've made all sorts of automated proofs in which there are hundreds of steps, each perfectly verifiable by computer. But—like the innards of a neural net—what's going on looks alien and not understandable by a human.

For a human to understand, there have to be familiar "conceptual waypoints". It's pretty much like with words in languages. If some particular part of a proof has a name ("Smith's Theorem"), and has a known meaning, then it's useful to us. But if it's just a lump of undifferentiated computation, it won't be meaningful to us.

In pretty much any axiom system, there's an infinite set of possible theorems. But which ones are "interesting"? That's really a human question. And basically it's going to end up being ones with "stories". In the book I show that for the simple case of basic logic, the theorems that have historically been considered interesting enough to be given names happen to be precisely the ones that are in some sense minimal.

But my guess is that for richer axiom systems pretty much anything that's going to be considered "interesting" is going to have to be reached from things that are already considered interesting. It's like building up words or concepts: you don't get to introduce new ones unless you can directly relate them to existing ones.

In recent years I've wondered quite a bit about how inexorable or not progress is in a field like mathematics. Is there just one historical path that can be taken, say from arithmetic to algebra to the higher reaches of modern mathematics? Or are there an infinite diversity of possible paths, with completely different histories for mathematics?

The answer is going to depend on—in a sense—the "structure of metamathematical space": just what is the network of true theorems that avoid the sea of undecidability? Maybe it'll be different for different fields of mathematics, and some will be more "inexorable" (so it feels like the math is being "discovered") than others (where it seems more like the math is arbitrary, and "invented").

But to me one of the most interesting things is how close—when viewed in these kinds of terms—questions about the nature and character of mathematics end up being to questions about the nature and character of intelligence and AI. And it's this kind of commonality that makes me realize just how powerful and general the ideas in *A New Kind of Science* actually are.

When Is There a Science?

There are some areas of science—like physics and astronomy—where the traditional mathematical approach has done quite well. But there are others—like biology, social science and linguistics—where it's had a lot less to say. And one of the things I've long believed is that what's needed to make progress in these areas is to generalize the kinds of models one's using, to consider a broader range of what's out there in the computational universe.

And indeed in the past 15 or so years there's been increasing success in doing this. And there are lots of biological and social systems, for example, where models have now been constructed using simple programs.

But unlike with mathematical models which can potentially be "solved", these computational models often show computational irreducibility, and are typically used by doing explicit simulations. This can be perfectly successful for making particular predictions, or for applying the models in technology. But a bit like for the automated proofs of mathematical theorems one might still ask, "is this really science?".

Yes, one can simulate what a system does, but does one "understand" it? Well, the problem is that computational irreducibility implies that in some fundamental sense one can't always "understand" things. There might be no useful "story" that can be told; there may be no "conceptual waypoints"—only lots of detailed computation.

Imagine that one's trying to make a science of how the brain understands language—one of the big goals of linguistics. Well, perhaps we'll get an adequate model of the precise rules which determine the firing of neurons or some other low-level representation of the brain. And then we look at the patterns generated in understanding some whole collection of sentences.

Well, what if those patterns look like the behavior of rule 30? Or, closer at hand, the innards of some recurrent neural network? Can we "tell a story" about what's happening? To do so would basically require that we create some kind of higher-level symbolic representation: something where we effectively have words for core elements of what's going on.

But computational irreducibility implies that there may ultimately be no way to create such a thing. Yes, it will always be possible to find patches of computational reducibility, where some things can be said. But there won't be a complete story that can be told. And one might say there won't be a useful reductionistic piece of science to be done. But that's just one of the things that happens when one's dealing with (as the title says) a new kind of science.

Controlling the AIs

People have gotten very worried about AI in recent years. They wonder what's going to happen when AIs "get much smarter" than us humans. Well, the Principle of Computational Equivalence has one piece of good news: at some fundamental level, AIs will never be "smarter"—they'll just be able to do computations that are ultimately equivalent to what our brains do, or, for that matter, what all sorts of simple programs do.

As a practical matter, of course, AIs will be able to process larger amounts of data more quickly than actual brains. And no doubt we'll choose to have them run many aspects of the world for us—from medical devices, to central banks to transportation systems, and much more.

So then it's important to figure how we'll tell them what to do. As soon as we're making serious use of what's out there in the computational universe, we're not going to be able to give a line-by-line description of what the AIs are going to do. Rather, we're going to have to define goals for the AIs, then let them figure out how best to achieve those goals.

In a sense we've already been doing something like this for years in the Wolfram Language. There's some high-level function that describes something you want to do ("lay out a graph", "classify data", etc.). Then it's up to the language to automatically figure out the best way to do it.

And in the end the real challenge is to find a way to describe goals. Yes, you want to search for cellular automata that will make a "nice carpet pattern", or a "good edge detector". But what exactly do those things mean? What you need is a language that a human can use to say as precisely as possible what they mean.

It's really the same problem as I've been talking about a lot here. One has to have a way for humans to be able to talk about things they care about. There's infinite detail out there in the computational universe. But through our civilization and our shared cultural history we've come to identify certain concepts that are important to us. And when we describe our goals, it's in terms of these concepts.

Three hundred years ago people like Leibniz were interested in finding a precise symbolic way to represent the content of human thoughts and human discourse. He was far too early. But now I think we're finally in a position to actually make this work. In fact, we've already gotten a long way with the Wolfram Language in being able to describe real things in the world. And I'm hoping it'll be possible to construct a fairly complete "symbolic discourse language" that lets us talk about the things we care about.

Right now we write legal contracts in "legalese" as a way to make them slightly more precise than ordinary natural language. But with a symbolic discourse language we'll be able to write true "smart contracts" that describe in high-level terms what we want to have happen—and then machines will automatically be able to verify or execute the contract.

But what about the AIs? Well, we need to tell them what we generally want them to do. We need to have a contract with them. Or maybe we need to have a constitution for them. And it'll be written in some kind of symbolic discourse language, that both allows us humans to express what we want, and is executable by the AIs.

There's lots to say about what should be in an AI Constitution, and how the construction of such things might map onto the political and cultural landscape of the world. But one of the obvious questions is: can the constitution be simple, like Asimov's Laws of Robotics?

And here what we know from *A New Kind of Science* tells us the answer: it can't be. In a sense the constitution is an attempt to sculpt what can happen in the world and what can't. But computational irreducibility says that there will be an unbounded collection of cases to consider.

For me it's interesting to see how theoretical ideas like computational irreducibility end up impinging on these very practical—and central—societal issues. Yes, it all started with questions about things like the theory of all possible theories. But in the end it turns into issues that everyone in society is going to end up being concerned about.

There's an Endless Frontier

Will we reach the end of science? Will we—or our AIs—eventually invent everything there is to be invented?

For mathematics, it's easy to see that there's an infinite number of possible theorems one can construct. For science, there's an infinite number of possible detailed questions to ask. And there's also an infinite array of possible inventions one can construct.

But the real question is: will there always be interesting new things out there?

Well, computational irreducibility says there will always be new things that need an irreducible amount of computational work to reach from what's already there. So in a sense there'll always be "surprises", that aren't immediately evident from what's come before.

But will it just be like an endless array of different weirdly shaped rocks? Or will there be fundamental new features that appear, that we humans consider interesting?

It's back to the very same issue we've encountered several times before: for us humans to find things "interesting" we have to have a conceptual framework that we can use to think about them. Yes, we can identify a "persistent structure" in a cellular automaton. Then maybe we can start talking about "collisions between structures". But when we just see a whole mess of stuff going on, it's not going to be "interesting" to us unless we have some higher-level symbolic way to talk about it.

In a sense, then, the rate of "interesting discovery" isn't going to be limited by our ability to go out into the computational universe and find things. Instead, it's going to be limited by our ability as humans to build a conceptual framework for what we're finding.

It's a bit like what happened in the whole development of what became *A New Kind of Science*. People had seen related phenomena for centuries if not millennia (distribution of primes, digits of pi, etc.). But without a conceptual framework they just didn't seem "interesting", and nothing was built around them. And indeed as I understand more about what's out there in the computational universe—and even about things I saw long ago there—I gradually build up a conceptual framework that lets me go further.

By the way, it's worth realizing that inventions work a little differently from discoveries. One can see something new happen in the computational universe, and that might be a discovery. But an invention is about figuring out how something can be achieved in the computational universe.

And—like in patent law—it isn't really an invention if you just say "look, this does that". You have to somehow understand a purpose that it's achieving.

In the past, the focus of the process of invention has tended to be on actually getting something to work ("find the lightbulb filament that works", etc.). But in the computational universe, the focus shifts to the question of what you want the invention to do. Because once you've described the goal, finding a way to achieve it is something that can be automated.

That's not to say that it will always be easy. In fact, computational irreducibility implies that it can be arbitrarily difficult. Let's say you know the precise rules by which some chemicals can interact. Can you find a chemical synthesis pathway that will let you get to some particular chemical structure? There may be a way, but computational irreducibility implies that there may be no way to find out how long the pathway may be. And if you haven't found a pathway you may never be sure if it's because there isn't one, or just because you didn't reach it yet.

The Fundamental Theory of Physics

If one thinks about reaching the edge of science, one cannot help but wonder about the fundamental theory of physics. Given everything we've seen in the computational universe, is it conceivable that our physical universe could just correspond to one of those programs out there in the computational universe?

Of course, we won't really know until or unless we find it. But in the years since *A New Kind of Science* appeared, I've become ever more optimistic about the possibilities.

Needless to say, it would be a big change for physics. Today there are basically two major frameworks for thinking about fundamental physics: general relativity and quantum field theory. General relativity is a bit more than 100 years old; quantum field theory maybe 90. And both have achieved spectacular things. But neither has succeeded in delivering us a complete fundamental theory of physics. And if nothing else, I think after all this time, it's worth trying something new.

But there's another thing: from actually exploring the computational universe, we have a huge amount of new intuition about what's possible, even in very simple models. We might have thought that the kind of richness we know exists in physics would require some very elaborate underlying model. But what's become clear is that that kind of richness can perfectly well emerge even from a very simple underlying model.

What might the underlying model be like? I'm not going to discuss this in great detail here, but suffice it to say that I think the most important thing about the model is that it should have as little as possible built in. We shouldn't have the hubris to think we know how the universe is constructed; we should just take a general type of model that's as unstructured as possible, and do what we typically do in the computational universe: just search for a program that does what we want.

My favorite formulation for a model that's as unstructured as possible is a network: just a collection of nodes with connections between them. It's perfectly possible to formulate such a model as an algebraic-like structure, and probably many other kinds of things. But we can think of it as a network. And in the way I've imagined setting it up, it's a network that's somehow "underneath" space and time: every aspect of space and time as we know it must emerge from the actual behavior of the network.

Over the past decade or so there's been increasing interest in things like loop quantum gravity and spin networks. They're related to what I've been doing in the same way that they also involve networks. And maybe there's some deeper relationship. But in their usual formulation, they're much more mathematically elaborate.

From the point of view of the traditional methods of physics, this might seem like a good idea. But with the intuition we have from studying the computational universe—and using it for science and technology—it seems completely unnecessary. Yes, we don't yet know the fundamental theory of physics. But it seems sensible to start with the simplest hypothesis. And that's definitely something like a simple network of the kind I've studied.

At the outset, it'll look pretty alien to people (including myself) trained in traditional theoretical physics. But some of what emerges isn't so alien. A big result I found nearly 20 years ago (that still hasn't been widely understood) is that when you look at a large enough network of the kind I studied you can show that its averaged behavior follows Einstein's equations for gravity. In other words, without putting any fancy physics into the underlying model, it ends up automatically emerging. I think it's pretty exciting.

People ask a lot about quantum mechanics. Yes, my underlying model doesn't build in quantum mechanics (just as it doesn't build in general relativity). Now, it's a little difficult to pin down exactly what the essence of "being quantum mechanical" actually is. But there are some very suggestive signs that my simple networks actually end up showing what amounts to quantum behavior—just like in the physics we know.

OK, so how should one set about actually finding the fundamental theory of physics if it's out there in the computational universe of possible programs? Well, the obvious thing is to just start searching for it, starting with the simplest programs.

I've been doing this—more sporadically than I would like—for the past 15 years or so. And my main discovery so far is that it's actually quite easy to find programs that aren't obviously not our universe. There are plenty of programs where space or time are obviously completely different from the way they are in our universe, or there's some other pathology. But it turns out it's not so difficult to find candidate universes that aren't obviously not our universe.

But we're immediately bitten by computational irreducibility. We can simulate the candidate universe for billions of steps. But we don't know what it's going to do—and whether it's going to grow up to be like our universe, or completely different.

It's pretty unlikely that in looking at that tiny fragment of the very beginning of a universe we're going to ever be able to see anything familiar, like a photon. And it's not at all obvious that we'll be able to construct any kind of descriptive theory, or effective physics. But in a sense the problem is bizarrely similar to the one we have even in systems like neural networks: there's computation going on there, but can we identify "conceptual waypoints" from which we can build up a theory that we might understand?

It's not at all clear our universe has to be understandable at that level, and it's quite possible that for a very long time we'll be left in the strange situation of thinking we might have "found our universe" out in the computational universe, but not being sure.

Of course, we might be lucky, and it might be possible to deduce an effective physics, and see that some little program that we found ends up reproducing our whole universe. It would be a remarkable moment for science. But it would immediately raise a host of new questions—like why this universe, and not another?

Box of a Trillion Souls

Right now us humans exist as biological systems. But in the future it's certainly going to be technologically possible to reproduce all the processes in our brains in some purely digital—computational—form. So insofar as those processes represent "us", we're going to be able to be "virtualized" on pretty much any computational substrate. And in this case we might imagine that the whole future of a civilization could wind up in effect as a "box of a trillion souls".

Inside that box there would be all kinds of computations going on, representing the thoughts and experiences of all those disembodied souls. Those computations would reflect the rich history of our civilization, and all the things that have happened to us. But at some level they wouldn't be anything special.

It's perhaps a bit disappointing, but the Principle of Computational Equivalence tells us that ultimately these computations will be no more sophisticated than the ones that go on in all sorts of other systems—even ones with simple rules, and no elaborate history of civilization. Yes, the details will reflect all that history. But in a sense without knowing what to look for—or what to care about—one won't be able to tell that there's anything special about it.

OK, but what about for the "souls" themselves? Will one be able to understand their behavior by seeing that they achieve certain purposes? Well, in our current biological existence, we have all sorts of constraints and features that give us goals and purposes. But in a virtualized "uploaded" form, most of these just go away.

I've thought quite a bit about how "human" purposes might evolve in such a situation, recognizing, of course, that in virtualized form there's little difference between human and AI. The disappointing vision is that perhaps the future of our civilization consists in disembodied souls in effect "playing videogames" for the rest of eternity.

But what I've slowly realized is that it's actually quite unrealistic to project our view of goals and purposes from our experience today into that future situation. Imagine talking to someone from a thousand years ago and trying to explain that people in the future would be walking on treadmills every day, or continually sending photographs to their friends. The point is that such activities don't make sense until the cultural framework around them has developed.

It's the same story yet again as with trying to characterize what's interesting or what's explainable. It relies on the development of a whole network of conceptual waypoints.

Can we imagine what the mathematics of 100 years from now will be like? It depends on concepts we don't yet know. So similarly if we try to imagine human motivation in the future, it's going to rely on concepts we don't know. Our best description from today's viewpoint might be that those disembodied souls are just "playing videogames". But to them there might be a whole subtle motivation structure that they could only explain by rewinding all sorts of steps in history and cultural development.

By the way, if we know the fundamental theory of physics then in a sense we can make the virtualization complete, at least in principle: we can just run a simulation of the universe for those disembodied souls. Of course, if that's what's happening, then there's no particular reason it has to be a simulation of our particular universe. It could as well be any universe from out in the computational universe.

Now, as I've mentioned, even in any given universe one will never in a sense run out of things to do, or discover. But I suppose I myself at least find it amusing to imagine that at some point those disembodied souls might get bored with just being in a simulated version of our physical universe—and might decide it's more fun (whatever that means to them) to go out and explore the broader computational universe. Which would mean that in a sense the future of humanity would be an infinite voyage of discovery in the context of none other than *A New Kind of Science*!

The Economics of the Computational Universe

Long before we have to think about disembodied human souls, we'll have to confront the issue of what humans should be doing in a world where more and more can be done automatically by AIs. Now in a sense this issue is nothing new: it's just an extension of the long-running story of technology and automation. But somehow this time it feels different.

And I think the reason is in a sense just that there's so much out there in the computational universe, that's so easy to get to. Yes, we can build a machine that automates some particular task. We can even have a general-purpose computer that can be programmed to do a full range of different tasks. But even though these kinds of automation extend what we can do, it still feels like there's effort that we have to put into them.

But the picture now is different—because in effect what we're saying is that if we can just define the goal we want to achieve, then everything else will be automatic. All sorts of computation, and, yes, "thinking", may have to be done, but the idea is that it's just going to happen, without human effort.

At first, something seems wrong. How could we get all that benefit, without putting in more effort? It's a bit like asking how nature could manage to make all the complexity it does—even though when we build artifacts, even with great effort, they end up far less complex. The answer, I think, is it's mining the computational universe. And it's exactly the same thing for us: by mining the computational universe, we can achieve essentially an unbounded level of automation.

If we look at the important resources in today's world, many of them still depend on actual materials. And often these materials are literally mined from the Earth. Of course, there are accidents of geography and geology that determine by whom and where that mining can be done. And in the end there's a limit (if often very large) to the amount of material that'll ever be available.

But when it comes to the computational universe, there's in a sense an inexhaustible supply of material—and it's accessible to anyone. Yes, there are technical issues about how to "do the mining", and there's a whole stack of technology associated with doing it well. But the ultimate resource of the computational universe is a global and infinite one. There's no scarcity and no reason to be "expensive". One just has to understand that it's there, and take advantage of it.

The Path to Computational Thinking

Probably the greatest intellectual shift of the past century has been the one towards the computational way of thinking about things. I've often said that if one picks almost any field "X", from archaeology to zoology, then by now there either is, or soon will be, a field called "computational X"—and it's going to be the future of the field.

I myself have been deeply involved in trying to enable such computational fields, in particular through the development of the Wolfram Language. But I've also been interested in what is essentially the meta problem: how should one teach abstract computational thinking, for example to kids? The Wolfram Language is certainly important as a practical tool. But what about the conceptual, theoretical foundations?

Well, that's where *A New Kind of Science* comes in. Because at its core it's discussing the pure abstract phenomenon of computation, independent of its applications to particular fields or tasks. It's a bit like with elementary mathematics: there are things to teach and understand just to introduce the ideas of mathematical thinking, independent of their specific applications. And so it is too with the core of *A New Kind of Science*. There are things to learn about the computational universe that give intuition and introduce patterns of computational thinking—quite independent of detailed applications.

One can think of it as a kind of "pre computer science" , or "pre computational X". Before one gets into discussing the specifics of particular computational processes, one can just study the simple but pure things one finds in the computational universe.

And, yes, even before kids learn to do arithmetic, it's perfectly possible for them to fill out something like a cellular automaton coloring book—or to execute for themselves or on a computer a whole range of different simple programs. What does it teach? Well, it certainly teaches the idea that there can be definite rules or algorithms for things—and that if one follows them one can create useful and interesting results. And, yes, it helps that systems like cellular automata make obvious visual patterns, that for example one can even find in nature (say on mollusc shells).

As the world becomes more computational—and more things are done by AIs and by mining the computational universe—there's going to an extremely high value not only in understanding computational thinking, but also in having the kind of intuition that develops from exploring the computational universe and that is, in a sense, the foundation for *A New Kind of Science*.

What's Left to Figure Out?

My goal over the decade that I spent writing *A New Kind of Science* was, as much as possible, to answer all the first round of "obvious questions" about the computational universe. And looking back 15 years later I think that worked out pretty well. Indeed, today, when I wonder about something to do with the computational universe, I find it's incredibly likely that somewhere in the main text or notes of the book I already said something about it.

But one of the biggest things that's changed over the past 15 years is that I've gradually begun to understand more of the implications of what the book describes. There are lots of specific ideas and discoveries in the book. But in the longer term I think what's most significant is how they serve as foundations, both practical and conceptual, for a whole range of new things that one can now understand and explore.

But even in terms of the basic science of the computational universe, there are certainly specific results one would still like to get. For example, it would be great to get more evidence for or against the Principle of Computational Equivalence, and its domain of applicability.

Like most general principles in science, the whole epistemological status of the Principles of Computational Equivalence is somewhat complicated. Is it like a mathematical theorem that can be proved? Is it like a law of nature that might (or might not) be true about the universe? Or is it like a definition, say of the very concept of computation? Well, much like, say, the Second Law of Thermodynamics or Evolution by Natural Selection, it's a combination of these.

But one thing that's significant is that it's possible to get concrete evidence for (or against) the Principle of Computational Equivalence. The principle says that even systems with very simple rules should be capable of arbitrarily sophisticated computation—so that in particular they should be able to act as universal computers.

And indeed one of the results of the book is that this is true for one of the simplest possible cellular automata (rule 110). Five years after the book was published I decided to put up a prize for evidence about another case: the simplest conceivably universal Turing machine. And I was very pleased that in just a few months the prize was won, the Turing machine was proved universal, and there was another piece of evidence for the Principle of Computational Equivalence.

There's a lot to do in developing the applications of *A New Kind of Science*. There are models to be made of all sorts of systems. There's technology to be found. Art to be created. There's also a lot to do in understanding the implications.

But it's important not to forget the pure investigation of the computational universe. In the analogy of mathematics, there are applications to be pursued. But there's also a "pure mathematics" that's worth pursuing in its own right. And so it is with the computational universe: there's a huge amount to explore just at an abstract level. And indeed (as the title of the book implies) there's enough to define a whole new kind of science: a pure science of the computational universe. And it's the opening of that new kind of science that I think is the core achievement of *A New Kind of Science*—and the one of which I am most proud.

For the 10th anniversary of *A New Kind of Science*, I wrote three posts:

- It's Been 10 Years: What's Happened with *A New Kind of Science*?
- Living a Paradigm Shift: Looking Back on Reactions to *A New Kind of Science*
- Looking to the Future of *A New Kind of Science*

The complete high-resolution *A New Kind of Science* is now available on the web. There are also a limited number of print copies of the book still available (all individually coded!).