Charting a Course for "Complexity": Metamodeling, Ruliology and More

September 23, 2021

This is the first of a series of pieces I'm planning in connection with the upcoming 20th anniversary of the publication of A New Kind of Science.

"There's a Whole New Field to Build..."

For me the story began nearly 50 years ago—with what I saw as a great and fundamental mystery of science. We see all sorts of complexity in nature and elsewhere. But where does it come from? How is it made? There are so many examples. Snowflakes. Galaxies. Lifeforms. Turbulence. Do they all work differently? Or is there some common underlying cause? Some essential "phenomenon of complexity"?

It was 1980 when I began to seriously work on these questions. And at first I did so in the main scientific paradigm I knew: models based on mathematics and mathematical equations. I studied the approaches people had tried to use. Nonequilibrium thermodynamics. Synergetics. Nonlinear dynamics. Cybernetics. General systems theory. I imagined that the key question was: "Starting from disorder and randomness, how could spontaneous self-organization occur, to produce the complexity we see?" For somehow I assumed that complexity must be created as a kind of filtering of ubiquitous thermodynamic-like randomness in the world.

At first I didn't get very far. I could write down equations and do math. But there wasn't any real complexity in sight. But in a quirk of history that I now realize had tremendous significance, I had just spent a couple of years creating a big computer system that was ultimately a direct forerunner of our modern Wolfram Language. So for me it was obvious: if I couldn't figure out things myself with math, I should use a computer.

And there was something else: the computer system I'd built was a language that I'd realized (in a nod to my experience with reductionist physical science) would be the most powerful if it could be based on principles and primitives that were as minimal as possible. It had worked out very well for the language. And so when it came to

minimal, most "meta" kind of model to use.

I didn't know just what magic ingredient I'd need in order to get complexity. But I thought I might as well start absolutely as simple as possible. And so it was that I set about running programs that I later learned were a simplified version of what had been called "cellular automata" before. I don't think it was even an hour before I realized that something very interesting was going on. I'd start from randomness, and "spontaneously" the programs would generate all sorts of complex patterns.

At first, it was experimental work. I'd make observations, cataloging and classifying what I saw. But soon I brought in analysis tools—from statistical mechanics, dynamical systems theory, statistics, wherever. And I figured out all sorts of things. But at the center of everything, there was still a crucial question: what was the essence of what I was seeing? And how did it connect to existing science?

I wanted to simplify still further. What if I didn't start from randomness, but instead started from the simplest possible "seed"? There were immediately patterns like fractals. But somehow I just assumed that a simple program, with simple rules, starting from a simple seed just didn't have what it took to make "true complexity". I had printouts (yes, that was still how it worked back then) that showed this wasn't true. But for a couple of years I somehow ignored them.

Then in 1984 I made my first high-resolution picture of rule 30. And I now couldn't get away from it: a simple rule and simple seed were making something that seemed extremely complex. But was it really that complex? Or was there some magic method of analysis that would immediately "crack" it? For months I looked for one. From mathematics. Mathematical physics. Computation theory. Cryptography. But I found nothing.

And slowly it began to dawn on me that I'd been fundamentally wrong in my basic intuition. And that in the world of simple programs—or at least cellular automata—complexity was actually easy to make. Could it really be that this was the secret that nature had been using all along to make complexity? I began to think it was at least a big part of it. I started to make connections to specific examples in crystal growth, fluid flow, biological forms and other places. But I also wanted to understand the fundamental principles of what was going on.

Simple programs could produce complex behavior. But why? It wasn't long before I realized something fundamental: that this was at its core a computational

different way of thinking about things. A fundamentally computational way.

At first I had imagined that having a program as a model of something was essentially just a convenience. But I realized that it wasn't. I realized that computational models were something fundamentally new, with their own conceptual framework, character and intuition. And as an example of that, I realized that they showed a new central phenomenon that I called computational irreducibility.

For several centuries, the tradition and aspiration of exact science had been to predict numbers that would say what a system would do. But what I realized is that in most of the computational universe of simple programs, you can't do that. Even if you know the rules for a system, you may still have to do an irreducible amount of computational work to figure out what it will do. And that's why its behavior will seem complex.

By 1985 I knew these things. And I was tremendously excited about their implications. I had got to this point by trying to solve the "problem of complexity". And it seemed only natural to label what could now be done as "complex systems theory": a theory of systems that show complexity, even from simple rules.

And so it was that in 1985 I began to promote the idea of a new field of "complex systems research", or, for short "complexity"—fueled by the discoveries I'd made about things like cellular automata.

Now that I know more about history I realize that the thrust of what I wanted to do had definite precursors, especially from the 1950s. For that was a time when the concepts of computing were first being worked out—and through approaches like cybernetics and the nascent area of artificial intelligence, people started exploring the broader scientific implications of computational ideas. But with no inkling of the phenomena I discovered decades later, this didn't seem terribly promising, and the effort was largely abandoned.

By the late 1970s, though, there were other initiatives emerging, particularly coming from mathematics and mathematical physics. Among them were fractals, catastrophe theory and chaos theory. Each in a different way explored some form of complexity. But all of them in a sense operated largely in the "comfort" of traditional mathematical ideas. And while they used computers as practical tools, they never made the jump to seeing computation as a core paradigm for thinking about science.

So what became of the "complex systems research" I championed in 1985? It's been 36 years now. Careers have come and gone. Several academic generations have passed by. Some things have developed well. Some things have not developed so well.

was a foundation for the whole tower of science and technology that I've spent my life since then building, most recently culminating in our Wolfram Physics Project and what in just the past few weeks I've called the multicomputational paradigm.

Nothing I've learned in these 36 years has dulled the strength and beauty of rule 30 and those early discoveries about complexity. But now I have so much more context, and a so-much-bigger conceptual framework—from which it's possible to see so much more about complexity and about its place and potential in science.

Back in 1985 I was pretty much a lone voice expressing the potential for studying complexity in science. Now there are perhaps a thousand scientific institutes around the world nominally focused on complexity. And my goal here is to share what I've learned and figured out about what's now possible to do under the banner of complexity.

There are exciting—and surprising—things. Some I was already beginning to think about in the 1980s. But others have only come into focus—or even become conceivable—as a result of very recent progress around our Physics Project and the formalism it has developed.

The Emergence of a New Kind of Science

Back in 1985 I was tremendously excited about the potential for developing the field of complex systems research. It seemed as if there was a vast new domain that had suddenly been made accessible to scientific exploration. And in it I could see so much great science that could be done, and so many wonderful opportunities for so many people.

I myself was still only 25 years old. But I'd had some organizational experience, both leading a research group, and starting my first company. And I set about applying what I knew to complex systems research. By the following year, I'd founded the first research center and the first journal in the field (*Complex Systems*, still going strong after 35 years). (And I'd also done things like suggesting "complexity" as the theme for what became the Santa Fe Institute.) But somehow everything moved very slowly.

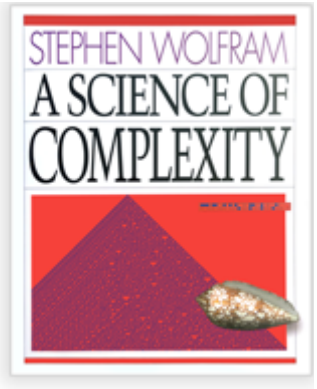
Despite my efforts, complex systems research wasn't a thing yet. It wasn't something universities were teaching; it wasn't something that was a category for funding. There were some applications for the field emerging. And there was tremendous pressure—particularly in the context of those applications—to shoehorn it into some existing area. Yes, it might have to take on the methodology of its "host" area. But at least it would have a home. But it really wasn't physics, or computer science, or math, or

thing, with its own, new, emerging methodology. And that was what I really thought should be developed.

I was impatient to have it happen. And by late 1986 I'd decided the best path was just to try to do it myself—and to set up the best tools and the best environment for that. The result was Mathematica (and now the Wolfram Language), as well as Wolfram Research. For a few years the task of creating these entirely consumed me. But in 1991 I returned to basic science and set about continuing where I had left off five years earlier.

It was an exciting time. I quickly found that the phenomena I had discovered in cellular automata were quite general. I explored all sorts of different kinds of rules and programs, always trying to understand the essence of what they were doing. But every time, the core phenomena I found were the same. Computational irreducibility—as unexpected as it had been when I first saw it in cellular automata—was everywhere. And I soon realized that beneath what I was seeing, there was a deep and general principle—that I called the Principle of Computational Equivalence—that I now consider to be the most fundamental thing we know about the computational universe.

But what did these discoveries about simple programs and the computational universe apply to? My initial target had been immediately observable phenomena in the natural world. And I had somehow assumed that ideas like evolutionary adaptation or mathematical proof would be outside the domain. But as the years went by, I realized that the force of the Principle of Computational Equivalence was much greater than I'd ever imagined, and that it encompassed these things too.



I spent the 1990s exploring the computational universe and its applications, and steadily writing a book about what I was discovering. At first, in recognition of my original objective, I called the book *A Science of Complexity*. But by the mid-1990s I had realized that what I was doing far transcended the specific goal of understanding the phenomenon of complexity.

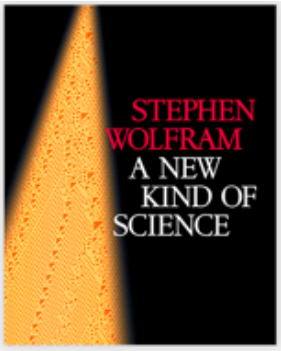
Instead, the core of what I was doing was to introduce a whole new kind of science, based on a new paradigm—essentially what I would now call the paradigm of computation. For three centuries, theoretical science had been dominated by the idea of using mathematical equations to describe the world. But now there was a new idea. The idea not of solving equations, but instead of setting up computational rules that could be explicitly run to represent and reproduce things in the world.

For three centuries theoretical models had been based on the fairly narrow set of constructs provided by mathematical equations, and particularly calculus. But now the

as a source of raw material for making models.

But with this new power came a sobering realization. Out in the unrestricted computational universe, computational irreducibility is everywhere. So, yes, there was now a way to create models for many things. But to figure out the consequences of those models might take irreducible computational work.

Without the computational paradigm, systems that showed significant complexity had seemed quite inaccessible to science. But now there was an underlying way to model them, and to successfully reproduce the complexity of their behavior. But computational irreducibility was all over them, fundamentally limiting what could be predicted or understood about how they behave.



For more than a decade, I worked through the implications of these ideas, continually surprised at how many foundational questions across all sorts of fields they seemed to address. And particularly given the tools and technology I'd developed, I think I became pretty efficient at the research I did. And finally in 2002 I decided I'd pretty much "picked all the low-hanging fruit", and it was time to publish my magnum opus, titled—after what I considered to be its main intellectual thrust—*A New Kind of Science*.

The book was a mixture of pure, basic science about simple programs and what they do, together with a discussion of principles deduced from studying these programs, as well as applications to specific fields. If the original question had been "Where does complexity come from?" I felt I'd basically nailed that—and the book was now an exploration of what one could do in a science where the emergence of complexity from simplicity was just a feature of the deeper idea of introducing the computational paradigm as a foundation for a new kind of science.

I put tremendous effort into making the exposition in the book (in both words and pictures) as clear as possible—and contextualizing it with extensive historical research. And in general all this effort paid off excellently, allowing the message of the book to reach a very wide audience.

What did people take away from the book? Some were confused by its new paradigm ("Where are all the equations?"). Some saw it as a somewhat mysterious wellspring of new forms and structures ("Those are great pictures!"). But what many people saw in it was a thousand pages of evidence that simple programs—and computational rules—could be a rich and successful source of models and ideas for science.

a remarkable—if somewhat silent—transformation. For three hundred years, serious models in science had essentially always been based on mathematical equations. But in the short space of just twenty years that's all changed—and now the vast majority of new models are based not on equations but on programs. It's a dramatic and important paradigmatic change, whose implications are just beginning to be felt.

But what of complexity? In the past it was always a challenge to "get complexity" out of a model. Now—with computational models—it tends to be very easy. Complexity has gone from something mysterious and out of reach to something ubiquitous and commonplace. But what has that meant for the "study of complexity"? Well, that's a story with quite some complexity to it….

The Growth of "Complexity"

From about 1984 to 1986 I put great effort into presenting and promoting the idea of "complex systems research". But by the time I basically left the field in 1986 to concentrate on technology for a few years, I hadn't seen much traction for the idea. A decade later, however, the story was quite different. I myself was quietly working away on what became *A New Kind of Science*. But elsewhere it seemed like my "marketing message" for complexity had firmly taken root, and there were complexity institutes starting to pop up all over the place.

What did people even mean by "complexity"? It often seemed to mean different things to different people. Sometimes it just meant "stuff in our field we haven't figured out yet". More often it meant "stuff that seems fundamental but we haven't been able to figure out". Pretty often there was some visualization component: "Look at how complex this plot seems!" But whatever exactly it might mean to different people, "complexity" was unquestionably becoming a popular science "brand", and there were plenty of people eager to affiliate with it—at the very least to give work they'd been doing for years a new air of modernity.

And while it was easy to be cynical about some of this, it had one very important positive consequence: "complexity" became a kind of banner for interdisciplinary work. As science had gotten bigger and more institutionalized, it inevitably became more siloed, with people in different departments at the same university routinely never having even met. But now people from all kinds of fields could say, "Yes, we run into complexity in our field", and with complexity as a banner they now had a reason to connect, and maybe even to form an institute together.

So what actually got done? Some of it I might summarize as "Yes, it's complex, but we can find something mathematical in it"—with a typical notion being the pursuit of

starts to actually take the computational paradigm on board—with the thrust typically being "We can write a program to reproduce what we're looking at".

And one of the great feelings of power has been that even in fields—like the social sciences—where there haven't really been much more than "verbal" models before, it's now appeared possible to get models that at least seem much more "scientific". Sometimes the models have been purely empirical ("Look, there's a power law!"). Sometimes they have been based on constructing programs to reproduce behavior.

The definition of success has often been a bit questionable, however. Yes, there's a program that shows some features of whatever system one's looking at. But how complicated is the program? How much of what's coming out is basically just being put right into the program? For mathematical models, people have long had familiarity with questions like "How many parameters does that model have?". But when it comes to programs, there's been a tendency just to put more and more into them without doing much accounting of it.

And then there's the matter of complexity. Let's say whatever one's trying to model shows complexity. Then often the thinking seems to be that to get that complexity out, there's a need to somehow have enough complexity in the model. And when complexity does manage to come out, there's a feeling that this is some kind of triumph, and evidence that the model is on the right track.

But actually—as I discovered in studying the computational universe of simple programs—this really isn't the right intuition at all. Because it fundamentally doesn't take into account computational irreducibility. And knowing that computational irreducibility is ubiquitous, we know that complexity is too. It's not something special and "on the right track" that's making a model produce complexity; instead producing complexity is just something a very wide range of computational models naturally do.

Still, the general field and brand of complexity continued to gain traction. Back in 1986 my *Complex Systems* had been the only journal devoted to complex systems research. By the late 2010s there were dozens of journals in the field. And my original efforts from the 1980s to promote the study of complexity had been thoroughly dwarfed by a whole "complexity industry" that had grown up. But looking at what's been done, I feel like there's something important that's missing. Yes, it's wonderful that there's been so much "complexity activity". But it feels scattered and incoherent—and without a strong common thread.

Returning to the Foundations of Complexity

does it fit together? And what are its intellectual underpinnings? The dynamics of academia has led most of the ongoing activity of complexity research to be about specific applications in specific fields—and not really to concern itself with what basic science might lie underneath, and what the "foundations of complexity" might be.

But the great power of basic science is the economy of scale it brings. Find one principle of basic science and it can inform a vast range of different specific applications, that would otherwise each have to be explored on their own. Learn one principle of basic science and you immediately know something that subsumes all sorts of particular things you would otherwise separately have to learn.

So what about complexity? Is there something underneath all those specifics that one can view as a coherent "basic science of complexity"—and for example the raw material for something like a course on the "Foundations of Complexity"? At first it might not be obvious where to look for this. But there's immediately a big clue. And it's what is in a sense the biggest "meta discovery" of the study of complexity over the past few decades: that across all kinds of systems, computational models work.

So then one's led to the question of what the basic science of computational models—or computational systems in general—might be. But that's precisely what my work on the computational universe of simple programs—and my book *A New Kind of Science*—are about. They're about the core basic science of the computational universe, and the principles it involves—in a sense the foundational science of computation.

It's important, by the way, to distinguish this from computer science. Computer science is about programs and computations that we humans construct for certain purposes. But the foundational science we need is instead about programs and computations "in the wild"—and about what's out there in general in the computational universe, independent of whether we humans would have a reason to construct or use it.

It's a very abstract kind of thing. That—like pure mathematics—can be studied completely on its own, without reference to any particular application. And in fact the analogy to pure mathematics is an apt one. Because just as pure mathematics is in a sense the abstract underpinning for the mathematical sciences and the whole mathematical paradigm for representing the world, so now our foundational science of computation is the abstract underpinning for the computational paradigm for representing the world—and for all the "computational X" fields that flow from it.

So, yes, there is a core basic science of complexity. And it's also essentially the foundational science of computation. And by studying this, we can bring together all sorts of seemingly disparate issues that arise in the study of complexity in different

intrinsic randomness generation. Everywhere we'll see the effects of the Principle of Computational Equivalence. These are general, abstract things from pure basic science. They're the intellectual underpinnings of the study of complexity—the "foundations of complexity".

Metamodeling and the Metatheory of Models

> **metamodeling** (*n.*) the metascience of finding minimal models for models

I was at a complexity conference once, talking to someone who was modeling fish and their behavior. Proudly the person showed me his simulated fish tank. "How many parameters does this involve?", I asked. "About 90", he said. "My gosh", I said, "with that many parameters, you could put an elephant in your fish tank too!"

If one wanted to make a simulated fish tank display just for people to watch, then having all those parameters might be just fine. But it's not so helpful if one wants to understand the science of fish. The fish have different shapes. The fish swim around in different configurations. What are the core things that lead to what we see?

To answer that, we have to drill down: we have to find the essence of fish shape, or fish behavior.

At first, if confronted with complexity, we might say "It's hopeless, we'll never find the essence of what's going on—it's all too complicated". But the whole point is that we know that in the computational universe of possible programs, there can in fact be simple programs with simple rules that lead to immense complexity. So even though there's immense complexity in behavior we see, underneath it all there can still be something simple and understandable.

In a sense, the concept of taking phenomena and drilling down to find their underlying essential causes is at the heart of reductionist science. But as this has traditionally been practiced, it's relied on being able to see one's way through this "drilling down" process, or in effect, to explicitly do reverse engineering. But a big lesson of the computational paradigm is the phenomenon of computational irreducibility—and the "irreducible distance" that can exist between rules and the behavior they produce.

It's a double-edged thing, however. Yes, it's hard to drill down through computational irreducibility. But in the end the details of what's underneath may not matter so much; the main features one sees may just be generic reflections of the phenomenon of

Still, there are normally structural features of the underlying models (or their interpretations) that matter for particular applications. Is one dealing with something on a 2D grid? Are there nonlocal effects in the system? Is there directionality to the states of the system? And so on.

If one looks at the literature of complexity, one finds all sorts of models for all sorts of systems. And often—like the fish example—the models are very complicated. But the question is: are there simpler models lurking underneath? Models simple enough that one can readily understand at least their basic rules and structure. Models simple enough that it's plausible that they could be useful for other systems as well.

To find such things is in a sense an exercise in what one can call "metamodeling": trying to make a model of a model, doing reductionist science not on observations of the world, but on the structure of models.

When I first worked on the problem of complexity, one of the main things I did was a piece of metamodeling. I was looking at models for a whole variety of phenomena, from snowflake growth to self-gravitating gases to neural nets. But what I did was to try to identify an underlying "metamodel" that would cover all of them. And what I came up with was simple cellular automata (which, by the way, don't cover everything I had been looking at, but turn out to be very interesting anyway).

As I think about it now, I realize that the activity of metamodeling is not a common one in science. (In mathematics, one could argue that something like categorification is somewhat analogous.) But to me personally, metamodeling has seemed very natural —because it's very much like something I've done for a very long time, which is language design.

What's involved in language design? You start off from a whole collection of computations, and descriptions of how to do them. And then you try to drill down to identify a small set of primitives that let you conveniently build up those computations. Just like metamodeling is about removing all the "hairy" parts of models to get to their minimal, primitive forms, so also language design is about doing that for computations and computational structures.

minimal forms are humans. And it's to humans that they need to seem "simple" and understandable. Some of the practical definition of simplicity has to do with history. What, for example, has become familiar, or are there words for? Some is more about human perception. What can be represented by a diagram that our visual processing system can readily absorb?

But once one's found something minimal, the great value of it is that it tends to be very general. Whereas a detailed "hairy" model tends to have all sorts of features specific to a particular system, a simple model tends to be applicable to all sorts of systems. So by doing the metamodeling, and finding the simplest "common" model, one is effectively deriving something that will have the greatest leverage.

I've seen this quite dramatically with cellular automata over the past forty years. Cellular automata are in a sense minimal models in which there's a definite (discrete) structure for space and time and a finite number of states associated with each discrete cell. And it's been remarkable how many different kinds of systems can successfully be modeled by cellular automata. So that for example of the 256 very simplest 2-color nearest-neighbor 1D rules, a significant fraction have found application somewhere, and many have found several (often completely different) applications.

I have to say that I haven't explicitly thought of myself as pursuing "metamodeling" in the past (and I only just invented the term!). But I believe it's an important technique and idea. And it's one that can "mine" the specific modeling achievements of work on complexity and bring them to a broader and more foundational level.

In *A New Kind of Science* I cataloged and studied minimal kinds of models of many types. And in the twenty years since *A New Kind of Science* was finished, I have only seen a modest number of new minimal models (though I haven't been looking for them with the focus that metamodeling now brings). But recently, I have another major example of what I now call metamodeling. For our Physics Project we've developed a particular class of models based on multiway hypergraph rewriting. But I've recently realized that there's metamodeling to do here, and the result has been the general concept of multicomputation and multicomputational models.

Returning to complexity, one can imagine taking all the academic papers in the field and identifying the models they use—and then trying to do metamodeling to classify and boil down these models. Often, I suspect, the resulting minimal classes of models will be ones we've already seen (and that, for example, appear in *A New Kind of Science*). But occasionally they will be new: in a sense new primitives for the language of modeling, and new "metascientific" output from the study of complexity.

ruliology (*n.*) the pure basic science of what simple rules do

If one sets up a system to follow a particular set of simple rules, what will the system do? Or, put another way, how do all those simple programs out there in the computational universe of possible programs behave?

These are pure, abstract questions of basic science. They're questions one's led to ask when one's operating in the computational paradigm that I describe in *A New Kind of Science*. But at some level they're questions about the specific science of what abstract rules (that we can describe as programs) do.

What is that science? It's not computer science, because that would be about programs we construct for particular purposes, rather than ones that are just "out there in the wilds of the computational universe". It's not (as such) mathematics, because it's all about "seeing what rules do" rather than finding frameworks in which things can be proved. And in the end, it's clear it's actually a new science—that's rich and broad, and that I, at least, have had the pleasure of practicing for forty years.

But what should this science be called? I've wondered about this for decades. I've filled so many pages with possible names. Could it be based on Greek or Latin words associated with rules? Those are *arch-* and *reg-*: very well-trafficked roots. What about words associated with computation? That'd be *logis-* or *calc-*. None of these seem to work. But—in something akin to the process of metamodeling—we can ask: What is the essence of what we want to communicate in the word?

It's all about studying rules, and what their consequences are. So why not the simple and obvious "ruliology"? Yes, it's a new and slightly unusual-sounding word. But I think it does well at communicating what this science that I've enjoyed for so long is about. And I, for one, will be pleased to call myself a "ruliologist".

But what is ruliology really about? It's a pure, basic science—and a very clean and precise one. It's about setting up abstract rules, and then seeing what they do. There's no "wiggle room". No issue with "reproducibility". You run a rule, and it does what it does. The same every time.

What does the rule 73 cellular automaton starting from a single black cell do? What does some particular Turing machine do? What about some particular multiway string substitution system? These are specific questions of ruliology.

notice some particular feature. And then you can use whatever methods it takes to get a specific ruliological result—and to establish, for example, that in the rule 73 pattern, black cells appear only in odd-length blocks.

Ruliology tends to start with specific cases of specific rules. But then it generalizes, looking at broader ranges of cases for a particular rule, or whole classes of rules. And it always has concrete things to do—visualizing behavior, measuring specific features, and so on.

But ruliology quickly comes face to face with computational irreducibility. What does some particular case of some particular rule eventually do? That may require an irreducible amount of computational effort to find out—and if one insists on knowing what amounts to a general truly infinite-time result, it may be formally undecidable. It's the same story with looking at different cases of a rule, or different rules. Is there any case that does this? Or any rule that does it?

What's remarkable to me—even after 40 years of ruliology—is how many surprises there end up being. You have some particular kind of rule. And it looks as if it's only going to behave in some particular way. But no, eventually you find a case where it does something completely different, and unexpected. And, yes, this is in effect computational irreducibility reaching into what one's seeing.

Sometimes I've thought of ruliology as being at first a bit like natural history. You're exploring the world of simple programs, finding what strange creatures exist in it—and capturing them for study. (And, yes, in actual biological natural history, the diversity of what one sees is presumably at its core exactly the same computational phenomenon we see in abstract ruliology.)

So how does ruliology relate to complexity? It's a core part—and in fact the most fundamental part—of studying the foundations of complexity. Ruliology is like studying complexity at its ultimate source. And about seeing just how complexity is generated from its simplest origins.

Ruliology is what builds raw material—and intuition—for making models. It's what shows us what's possible in the computational universe, and what we can use to model—and understand—the systems we study.

In metamodeling we're going from models that have been constructed, and drilling down to see what's underneath them. In ruliology we're in a sense going the other way, building up from the minimal foundations to see what can happen.

In some ways, ruliology is like natural science. It's taking the computational universe

ways, ruliology is something more generative than natural science: because within the science itself, it's thinking not only about what is, but also about what can abstractly be generated.

Ruliology in some ways starts as an experimental science, and in some ways is abstract and theoretical from the beginning. It's experimental because it's often concerned with just running simple programs and seeing what they do (and in general, computational irreducibility suggests you often can't do better). But it's abstract and theoretical in the sense that what's being run is not some actual thing in the natural world, with all its details and approximations, but something completely precise, defined and computational.

Like natural science, ruliology starts from observations—but then builds up to theories and principles. Long ago I found a simple classification of cellular automata (starting from random initial conditions)—somehow reminiscent of identifying solids, liquids and gases, or different kingdoms of organisms. But beyond such classifications, there are also much broader principles—with the most important, I believe, being the Principle of Computational Equivalence.

The everyday course of doing ruliology doesn't require engaging directly with the whole Principle of Computational Equivalence. But throughout ruliology, the principle is crucial in guiding intuition, and having an idea of what to expect. And, by the way, it's from ruliology that we can get evidence (like the universality of rule 110, and of the 2,3 Turing machine) for the broad validity of the principle.

I've been doing ruliology (though not by that name) for forty years. And I've done a lot of it. In fact, it's probably been my top methodology in everything I've done in science. It's what led me to understand the origins of complexity, first in cellular automata. It's what led me to formulate the general ideas in *A New Kind of Science*. And it's what gave me the intuition and impetus to launch our new Physics Project.

I find ruliology deeply elegant, and satisfying. There's something very aesthetic—at least to me—about the purity of just seeing what simple rules do. (And it doesn't hurt that they often make very pleasing images.) It's also satisfying when one can go from so little and get so much—and do so automatically, just by running something on a computer.

And as well I like the fundamental permanence of ruliology. If one's dealing with the simplest rules of some type, they're going to be foundational not only now, but forever. It's like simple mathematical constructs—like the icosahedron. There were icosahedral dice in ancient Egypt. But when we find them today, their shapes still seem completely

the rule 30 pattern or countless other discoveries in ruliology.

In a sense perhaps one of the biggest surprises is that ruliology is such a comparatively new activity. But as I cataloged in *A New Kind of Science*, it has precursors going back hundreds and perhaps thousands of years. But without the whole paradigm of *A New Kind of Science,* there wasn't a context to understand why ruliology is so significant.

So what constitutes a good piece of ruliology? I think it's all about simplicity and minimality. The best ruliology happens after metamodeling is finished—and one's really dealing with the simplest, most minimal class of rules of some particular type. In my efforts to do ruliology, for example in *A New Kind of Science*, I like to be able to "explain" the rules I'm using just by an explicit diagram, if possible with no words needed.

Then it's important to show what the rules do—as explicitly as possible. Sometimes—as in cellular automata—there's a very obvious visual representation that can be used. But in other cases it's important to do the work to find some scheme for visualization that's as explicit as possible, and that both shows the whole of what's going on and doesn't introduce distracting or arbitrary additional elements.

It's amazing how often in doing ruliology I'll end up making an array of thumbnail images of how certain rules behave. And, again, the explicitness of this is important. Yes, one often wants to do various kinds of filtering, say of rules. But in the end I've found that one needs to just look at what happens. Because that's the only way to successfully notice the unexpected, and to get a sense of the irreducible complexity of what's out there in the computational universe of possible rules.

When I see papers that report what amounts to ruliology, I always like it when there are explicit pictures. I'm disappointed if all I see are formal definitions, or plots with curves on them. It's an inevitable consequence of computational irreducibility that in doing good ruliology, one has to look at things more explicitly.

One of the great things about ruliology as a field of study is how easy it is to explore new territory. The computational universe contains an infinite number of possible rules. And even among ones that one might consider "simple", there are inevitably astronomically many on any human scale. But, OK, if one explores some particular ruliological system, what of it?

Exploring some particular class of rules, you may be lucky enough to come upon some new phenomenon, or understand some new general principle. But what you know you'll be doing is systematically adding to the body of knowledge in ruliology.

Why is that important? For a start, ruliology is what provides the raw material for making models, so you're in effect creating a template for some potential future model. And in addition, when it comes to technology, an important approach that I've discussed (and used) quite extensively involves "mining" the computational universe for "technologically useful" programs. And good ruliology is crucial in helping to make that feasible.

It's a bit like creating technology in the physical universe. It was crucial, for example, that good physics and chemistry had been done on liquid crystals. Because that's what allowed them to be identified—and used—in making displays.

Beyond its "pragmatic" value for models and for technology, another thing ruliology does is to provide "empirical raw material" for making broader theories about the computational universe. When I discovered the Principle of Computational Equivalence, it was as a result of several years of detailed ruliology on particular types of rules. And good ruliology is what prepares and catalogs examples from which theoretical advances can be made.

It's worth mentioning that there's a certain tendency to want to "nail down ruliology" using, for example, mathematics. And sometimes it's possible to derive a nice summary of ruliological results using, say, some piece of discrete mathematics. But it's remarkable how quickly the mathematics tends to get out of hand, with even a very simple rule having behavior that can only be captured by large amounts of obscure mathematics. But of course that's in a sense just computational irreducibility rearing its head. And showing that mathematics is not the methodology to use—and that instead something new is needed. Which is precisely where ruliology comes in.

I've spent many years defining the character and subject matter of what I'm now calling ruliology. But there's something else I've done too, which is to build a large tower of practical technology for actually doing ruliology. It's taken more than forty years to build up to what's now the full-scale computational language that is the Wolfram Language. But all that time, I was using what we were building to do ruliology.

The Wolfram Language is great and important for many things. But when it comes to ruliology, it's simply a perfect fit. Of course it's got lots of relevant built-in features. Like visualization, graph manipulation, etc., as well as immediate support for systems like cellular automata, substitution systems and Turing machines. But what's even

represent—and run—essentially any computational rule.

In doing practical ruliological explorations—and for example searching the computational universe—it's also useful to have immediate support for things like parallel computation. But another crucial aspect of the Wolfram Language for doing practical ruliology is the concept of notebooks and computable documents. Notebooks let one organize both the process of research and the presentation of its results.

I've been accumulating research notebooks about ruliology for more than 30 years now—with textual notes, images of behavior, and code. And it's a great thing. Because the stability of the Wolfram Language (and its notebook format) means that I can immediately go back to something I did 30 years ago, run the code, and build on it. And when it comes to presenting results, I can do it as a computational essay, created in a notebook—in which the task of exposition is shared between text, pictures and computational language code.

In a traditional technical paper based on the mathematical paradigm, the formal part of the presentation will normally use mathematical notation. But for ruliology (as for "computational X" fields) what one needs instead is computational notation, or rather computational language—which is exactly what the Wolfram Language provides. And in a good piece of ruliology—and ruliology presentation—the notation should be simple, clear and elegant. And because it's in computational language, it's not just something people read; it's also something that can immediately be executed or integrated somewhere else.

What should the future of ruliology be? It's a huge, wide-open field. In which there are many careers to be made, and immense numbers of papers and theses and books that can be written—that will build up a body of knowledge that advances not just the pure, basic science of the computational universe but also all the science and technology that flows from it.

Philosophy and the Foundations of Complexity

How should the phenomenon of complexity affect one's worldview, and one's general way of thinking about things? It's a bit of a roller-coaster-like ride. When first confronted with complexity in a system, one might think "There doesn't seem to be any science to that". But then with great effort it may turn out to be possible to "drill down" and find the underlying rules for the system, and perhaps they'll even be quite simple. And at that point we might think "OK, science has got this one"—we've solved it.

But that ignores computational irreducibility. And computational irreducibility implies

necessarily "scientifically predict" what the system will do; instead, it may take an irreducible amount of computational work to figure it out.

Yes, you may have a model that correctly captures the underlying rules for a system—and even explains the overall complexity in the behavior of the system. But that absolutely does not mean that you can successfully make specific predictions about what the system will do. Because computational irreducibility gets in the way, essentially "eating away the power of science from the inside"—as an inevitable formal fact about how systems based on the computational paradigm typically behave.

But in a sense even the very phenomenon of computational irreducibility—and even more so, the Principle of Computational Equivalence—give us ways to reason and think about things. It's a bit like in evolutionary biology, or in economics, where there are principles that don't specifically define predictions, but do give us ways to reason and think about things.

So what are some conceptual and philosophical consequences of computational irreducibility? One thing it does is to explain ubiquitous apparent randomness in the world, and say why it must happen—or at least must be perceived to happen by computationally bounded observers like us. And another thing it does is to tell us something about the perception of free will. Even if the underlying rules for a system (such as us humans) are deterministic, there can be an inevitable layer of computational irreducibility which makes the system still seem to a computationally bounded observer to be "free".

Metamodeling and ruliology are in effect the extensions of traditional science needed to handle the phenomenon of complexity. But what about extensions to philosophy?

For that one must think not just about the phenomenology of complexity, but really about its foundations. And that's where I think one inevitably runs into the whole computational paradigm, with all its intellectual implications. So, yes, there's a "philosophy of complexity", but it's really the "philosophy of the computational paradigm".

I started to explore this towards the end of *A New Kind of Science*. But there's much more to be done, and it's yet something else that can be reached by serious study of the foundations of complexity.

Multicomputation and the (Surprise) Return of Reducibility

Computational irreducibility is a very strong phenomenon, that in a sense pervades the computational universe. But within computational irreducibility, there must always be

amenable to a reduced description. And for example in doing ruliology, part of the effort is to catalog the computational reducibility one finds.

But in typical ruliology—or, for example, a random sampling of the computational universe of possible programs—computational reducibility is at best a scattered phenomenon. It's not something one can count on seeing. But there's something confusing about this when it comes to thinking about our universe, and our experience of it. Because perhaps the most striking fact about our universe—and indeed the one that leads to the possibility of what we normally call science—is that there's order in what happens in it.

Yet even if the universe ultimately operates at the lowest level according to simple rules, we might expect that at our level, all we would see is rampant computational irreducibility. But in our recent Physics Project there has been a big surprise. Because with the structure of the models we used, it seemed that within all that computational irreducibility, we were always seeing certain slices of reducibility—that turn out to correspond to the major known laws of physics: general relativity and quantum mechanics.

A more careful examination showed that what was picking out this computational reducibility was really the combination of two things. First, a certain general structure to the underlying model. And second, certain rather general features of us as observers of the system.

In the usual computational paradigm, one imagines rules that are successively applied to determine how the state of a system should evolve in time. But our Physics Project needed a new paradigm—that I've recently called the multicomputational paradigm—in which there can be many possible states of a system evolving in effect on many possible interwoven threads of time. In the computational paradigm, one can always identify the particular state reached after a certain amount of evolution. But in the multicomputational paradigm, it takes an observer to define how a "perceived state" should be extracted from all the possible threads of time.

In the multicomputational paradigm, the actual evolution on all the threads of time will show all sorts of computational irreducibility. But somehow what an observer like us perceives has "smoothed" all of that out. And what's left is something that's a reflection of the core structure of the underlying multicomputational rules. And that turns out to show a certain set of emergent "physics-like laws".

It's all an important piece of metamodeling. We started from a model intended to capture fundamental physics. But we've been able to "drill down" to find the essential

And wherever multicomputation occurs, we can expect that there will be computational reducibility and emergent physics-like laws, at least for certain kinds of observers.

So how does this relate to complexity? Well, when systems fundamentally follow the computational paradigm—with standard computational models—they'll tend to show computational irreducibility and complexity. But if instead they follow the multicomputational paradigm, then there'll be emergent laws to discover in them.

There are all sorts of fields—like economics, linguistics, molecular biology, immunology, etc.—where I have recently come to suspect that there may be good multicomputational models to be made. And in these fields, yes, there will be complexity to be seen. But the multicomputational paradigm suggests that there will also be definite regularities and emergent laws. So in a sense from "within complexity" there will inexorably emerge a certain simplicity. So that if one "observes the right things" one can potentially find what amount to "ordinary scientific laws".

It's a curious twist in the story of complexity, and one that I, for one, did not see coming. Back in the early 1980s when I was first working on complexity, I used to talk about finding "scientific laws of complexity". And at some level computational irreducibility and the Principle of Computational Equivalence are very general such laws—that were at first very surprising to see.

But what we've discovered is that in the multicomputational paradigm, there's another surprise: complexity can produce simplicity. But not just any simplicity. Simplicity that specifically follows physics-like laws. And that for a variety of fields might indeed give us something we could consider to be "scientific laws of complexity".

What Should Happen Now

It's a wonderful thing to see something go from "just an idea" to a whole, developed ecosystem in the world. But that's what's happened over the past forty years with the concept of doing science around the phenomenon of complexity. And over that time countless "workflows" associated with particular applications have been developed—and there's been all sorts of activity in all sorts of areas. But now I think it's time to take stock of what's been achieved—and see what might be possible going forward.

I myself have not been much involved in the "daily work of the complexity field" since my early efforts in the 1980s. And perhaps that distance makes it easier to see what lies ahead. For, yes, by now there's plenty of understanding of how to apply "complexity-inspired methodology" (and computational models) in particular areas. But the great opportunity is to turbocharge all this by focusing again on the "foundations of

various applications whose "workflows" have now been defined.

But what is that basic science? Its great "symptom" is complexity. But there's much more to it than that. It's heavily based on the computational paradigm. And it's full of deep and powerful ideas and methods. And I've been thinking about it for more than forty years. But it's only very recently—particularly based on what I've learned from our Physics Project—that I think I see with true clarity just how that science should be defined and pursued.

First, there's what I'm calling here metamodeling: going from specific models constructed for particular applications, and working out what the underlying more minimal and more general models are. And second, there's what I'm calling ruliology: the study of what possible rules (or possible programs) in the computational universe do.

Metamodeling is a kind of "meta" analog of science, probably most directly related to activities like computational language design. Ruliology is a pure, basic science, a bit like pure mathematics, but based on a very different methodology.

In both metamodeling and ruliology there is much of great value to do. And even after more than forty years of pursuing what I'm now calling ruliology, I feel as if I've only just scratched the surface of what's possible.

Applications under the banner of complexity will come and go as different fields and their objectives ebb and flow. But both metamodeling and ruliology have a certain purity, and clear anchoring to intellectual bedrock. And so we can expect that whatever is discovered there will—like the discoveries of pure mathematics—be part of the permanent corpus of theoretical knowledge.

Hovering over all of what we might study around complexity is the phenomenon of computational irreducibility. But within that irreducibility are pockets and slices of reducibility. And informed by our Physics Project, we now know that multicomputational systems can be expected to expose to observers like us what amount to physics-like laws—in effect leveraging the phenomenon of complexity to deliver accessible scientific laws.

a sense when we see complexity what is really happening is that some lump of irreducible computation is being exposed. So at its core, the study of complexity is a study of irreducible computation. It's computation whose details are irreducibly hard to figure out. But which we can reason about, and which, for example, we can also use for technology.

Even forty years ago, the fundamental origin of complexity still seemed like a complete mystery—a great secret of nature. But now through the computational paradigm, I think we have a clear notion of where complexity fundamentally comes from. And by leveraging the basic science of the computational universe—and what I'm now calling metamodeling and ruliology—there's a tremendous opportunity that now exists to dramatically advance everything that's been done under the banner of complexity.

The first phase of "complexity" is complete. The ecosystem is built. The applications are identified. The workflows are defined. And now it's time to return to the foundations of complexity. And to take the powerful basic science that lies there to define "complexity 2.0". And to deliver on the amazing potential that the concept of studying complexity has for science.

Coming soon: **Ruliology & metamodeling at the Wolfram Summer School… and much more.**

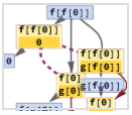
Cite this as > Stephen Wolfram (2021), "Charting a Course for 'Complexity': Metamodeling, Ruliology and More," Stephen Wolfram Writings. writings.stephenwolfram.com/2021/09/charting-a-course-for-complexity-metamodeling-ruliology-and-more.

Posted in: Computational Science, Computational Thinking, Future Perspectives, Historical Perspectives, New Kind of Science, Philosophy, Ruliology

Related Writings

Aggregation and Tiling as Multicomputational Processes
November 3, 2023

How to Think Computationally about AI, the Universe and Everything
October 27, 2023



Expression Evaluation and Fundamental Physics
September 29, 2023

Remembering Doug Lenat (1950–2023) and His Quest to Capture the World with Logic
September 5, 2023

Popular Categories

Artificial Intelligence	Language and Communication	Physics
Big Picture	Life and Times	Ruliology
Companies and Business	Life Science	Software Design
Computational Science	Mathematica	Wolfram|Alpha
Computational Thinking	Mathematics	Wolfram|One
Data Science	New Kind of Science	Wolfram Language
Education	New Technology	Other
Future Perspectives	Personal Analytics	
Historical Perspectives	Philosophy	

Writings by Year

2023 | 2022 | 2021 | 2020 | 2019 | 2018 | 2017 | 2016 | 2015 | 2014 | 2013 | 2012 |
2011 | 2010 | 2009 | 2008 | 2007 | 2006 | 2004 | 2003 | All