Beijer Institute
OF ECOLOGICAL ECONOMICS

KUNGL. VETENSKAPS-AKADEMIEN
THE ROYAL SWEDISH ACADEMY OF SCIENCES

DISCUSSION PAPER

Algorithms and the Shift in Modern Science

Beijer Discussion Paper Series No. 269

W. Brian Arthur. 2020.

Beijer Institute of Ecological Economics
The Royal Swedish Academy of Sciences

Lilla Frescativägen 4A
Stockholm

+46 8 673 95 00
beijer@beijer.kva.se

www.beijer.kva.se

Beijer Institute Discussion Paper 269:
Beijer Discussion Paper Series

Algorithms and the Shift in Modern Science

W. Brian Arthur

March 23, 2020

ABSTRACT

In the 1600s science shifted from being expressed in geometric form to being expressed in terms of algebraic equations. This paper argues that a new shift is underway: scientific thought is moving from being expressed in equation form to being expressed in algorithmic form. The shift is deep and it is gradual. It allows "context" to be considered—the situation surrounding the problem at hand. It brings new questions within the reach of science, allows expression of event-driven processes, is closely aligned with complexity science, allows formal expression for procedural sciences such as biology, and constitutes a new form of "theory."

Key Words: history of algebra, equations, algorithms, context, event-driven processes, complexity, shift in science, formation.

Algorithms and the Shift in Modern Science

W. Brian Arthur [1]

Something not particularly noticeable—yet something important—has been happening in science, economics, and environmental studies. We have traditionally expressed our thinking in the form of equation-based models, but increasingly we are expressing it in the form of algorithm-based models. A change in the language and methods we use is underway, and in this paper I want to explore why this should be and what it means for science.

Such a change is not the first in history. In the 1600s, science shifted from being expressed in geometric form to being expressed in terms of algebraic equations. The shift was slow and gradual, and it took about 120 years to be completed. Now, in the last several decades, I believe scientific thought is moving from being expressed in equation form to being expressed in algorithmic form.[2] I will argue that algorithms are steadily becoming a new and rigorous mode of expression in science. They are not replacing equations, we will continue to have equations indefinitely; rather they are adding a new layer—a new language—to the standard equation mode of expression, and as in the earlier mathematical shift from geometry to equations they will expand the scope of science.

The shift toward algorithmic expression, like the earlier shift to equations, is deep and gradual, and like many gradual shifts it is not particularly noticeable. It began roughly in the 1930s, and as with the earlier transition, the new system is layering on top of the old. What brought about the current move toward algorithmic expression most of all was the conditional clause: the addition of if-then possibilities to the old equation-based system. I will argue that these allow "context" to be considered, whether local, global, or external—"context" that can be arbitrarily complicated and can change the nature of the system the algorithm is dealing with. The new algorithmic system of expression brings new problems within the reach of science, it allows the expression of event-driven processes, it is closely aligned with complexity science, and it allows formal expression for procedural sciences such as biology.

---

[1] External Professor, Santa Fe Institute; Visiting Researcher, PARC; and Fellow at the Center for Advanced Study for the Behavioral Sciences, Stanford. This is a revised version of a lecture given first at Nanyang Technological University, Singapore, March 4, 2015, and on October 10, 2018 at the Stockholm Resilience Centre. I have expanded it and kept its informal style. I thank Ronan Arthur, John Seely Brown, Teresita Heiser, John Maxwell, Paul McJones, Ann Pendleton-Jullian, and Rika Preiser for useful comments. I also thank my former colleagues at the Beijer Institute for Ecological Economics, Royal Swedish Academy of Sciences, for discussions on this topic.

[2] Algorithms of course are "processes or sets of rules that are followed in calculations or other problem-solving operations," (*Oxford Living Dictionaries*). They are as old as mathematics itself, and predate both algebra and science. They have been used traditionally more as calculating methods than as descriptions of nature.

In this paper I will not attempt to examine all that algorithms can do, nor will I produce a history of algorithms. Rather I will explore this shift in science toward algorithmic expression, what it can provide that the old system can't, and the new way of thinking it brings.

Two Earlier Shifts

To understand the current change, I want to look at the previous one in the 1600s where science switched from being expressed in geometric terms to being expressed in algebraic terms. Algebra goes back much earlier than 1600, to the ninth-century Persian mathematician Muhammad ibn Musa al-Khwarizmi (from whose last name we get the modern word, algorithm). Al-Khwarizmi introduced the method of "restoring and balancing," a system for connecting unknown quantities with known ones and rearranging (restoring and balancing) these to arrive at values for the unknowns. The new method was greatly boosted in Europe in 1202 by Leonardo di Pisa (now known as Fibonacci). In his book *Liber Abbaci*, Leonardo demonstrated how the new method could be used to deal with difficult trading problems (Devlin, 2017). These early versions weren't concerned with applications in science—at that time science barely existed. Algebra was largely a tool for merchants, it existed only in wordy form, and as such was very much an art.

Things changed in 1591. The lawyer and mathematician François Piète introduced a new symbolism; he designated the unknowns being sought by vowels, and the given values by consonants. In doing so, he could speak of a general equation such as $ax^2 + bx + c = 0$ [in our modern notation] that could comprise particular instances such as $3x^2 + 12x + 8 = 0$. Algebra became abstract. And it got a further decisive stimulus in the 1630s from Pierre Fermat and Rene Descartes, both of whom developed systems by which geometry could be expressed in algebraic terms.[3]

The new method allowed difficult problems in geometry to be settled conveniently and transparently. Schuster (1977) shows a problem worked by Euclid in a thicket of detailed geometry then dispatched by Descartes in five lines of simple algebra (Gaukroger, 1995, p.175). And new questions could be formulated. Geometry was restricted to what could be expressed in 3 dimensions; algebra could go to 4 dimensions, indeed to an arbitrary number of dimensions. Algebra led in due course from equations to a theory of equations, to the calculus, to series expansions, to abstract algebras, to group theory. A new set of possibilities had opened that would redefine mathematics.

What advantage in thinking did the new method bring? What algebra did was to take problems formerly expressed in geometry and express these in a form of arithmetic. Simple algebra at its base is about designating unknown quantities as symbols, then performing arithmetic—addition, subtraction, multiplication, raising to powers, taking square roots—on the result. In fact, Newton called his 1707 text on algebra *Universal Arithmetick*. As algebra came into practice, shortcut rules emerged:

[3] Descartes brought in the modern notation where unknowns are x, y, z and parameters a, b, c; and x *cubus* becomes x^3.

substituting variables, collecting terms, completing the square, factoring out, and so on. Once the practitioner mastered these, algebra became a fast way to manipulate a set of expressions into a useful form.[4]

Given this advantage you might expect algebra to have quickly caught on. But it didn't. All through the decades of the 1600s, algebra was not quite acceptable. Geometry was familiar, algebra was not. Geometry was visual, algebra was not. Geometry had been inherited from the Greeks—to the Greeks geometry *was* mathematics. (The Greeks, for example, saw numbers as line lengths and areas—as geometrical objects.) Algebra had no such heritage. And geometry held an exalted position. "Geometry is a reflection of the mind of God," said Kepler. This now sounds grandiose, but in Europe up to the 1600s, natural philosophy (science) and theology were not yet fully separate, and the task of science in a God-created universe was to understand how God had designed the world and its planets and creatures. If God gave Order and Order was mathematical, Order must be geometrical. Geometry belonged to God.

Algebra by contrast belonged to trade. In fact, to use algebra was something of a cheat. Kepler called its use in astronomy "gauche." Thomas Hobbes, though peripheral in mathematics, called algebra a "scab of symbols," Galileo largely ignored it, and Isaac Newton wrote in a letter to Scottish mathematician David Gregory that "Algebra is the analysis of bunglers in mathematics" (Kline 1980). Newton was familiar with Descartes' methods but when he published the *Principia Mathematica* in 1687 he expressed the argument in what he called "good old-fashioned geometry" so it could be absorbed and accepted by other natural philosophers.

It wasn't until the 1720s or later that expression in algebra came to be accepted as normal and became "the language of mathematics itself" (Mahony, 1977). By the 1760s algebra's victory was complete, and science itself became defined in the new equational terms. Immanuel Kant, who had been a theoretical astronomer, took the new science of chemistry to task. "Chemistry," he wrote in 1786, "is a science but not Science. The criterion of true Science lies in its relation to mathematics." He meant of course algebraic mathematics. And 130 years later the biologist D'Arcy Thompson (1917) expressed much the same thought about biology.

In fact, biology is an embarrassment to Kant's principle. It is highly developed and in modern times it is redefining what "science" means, yet is not easily expressible in algebraic form. Evolution, speciation, embryology, protein expression, epigenetics, genetic regulatory networks—all these are very much driven by events, by processes; and processes are not easy to formulate via equations. Biology then is a hugely important science that doesn't lend itself to expression in equation form. And this

[4] Similar remarks extend to other equation systems: differential equations, trigonometry, complex algebra, linear algebra, etc., each of which has its own structures and rules for manipulation. Of course mathematics is much bigger and more abstract than these day-to-day analytical systems.

brings us to a question I want to look at later in the paper: does a science need to be equation-based to be "Science"?

Looking back from today, in spite of outlier fields like biology, the shift from geometry to algebraic equations has been a huge success. Whether in physics, or dynamics, or structural engineering, the new mode of expression has become the language we use unconsciously. To those who speak the language it tells a story, so that now we think of theory itself as contained in or represented by such equations.

Before we move on I want to point out that the shift into equations wasn't the first one to move science forward. Earlier, again due in Asia to Al-Khwarizmi and in Europe to Leonardo Di Pisa, commerce and nascent science had shifted from expression in Roman numerals to expression in Hindu-Arabic numerals. That system is by now so familiar to us it is hard to see the depth of change it brought. (Think of suddenly changing all numbers to binary.) In principle the old means could do much that the new means of expression could do—but only in principle. The new numbers allowed convenience and clarity, and they greatly extended what calculation and commerce could do (Goetzmann, 2004).

In both shifts the older system of expression of course was retained. We don't use Roman numerals much except for decorative purposes. But we still use geometry where we can, in statistics, physics, astronomy, optics, and economics. Geometry speaks directly to our visual senses. The new equation-based system may have triumphed, but it lies on top of the old.

If we take both these shifts we begin to see what a new language of expression means. We see (a) a broad equivalence between the new and old methods (often one can be translated into the other); (b) a new notation or means of expression that gives easier articulation of certain arguments and concepts; and (c) the extension of science into problems that couldn't be easily handled in the previous format. In the case of algebra, the new language redefined what mathematics was, and thereby what science was, and what we mean when we think of something being expressed "scientifically."

§

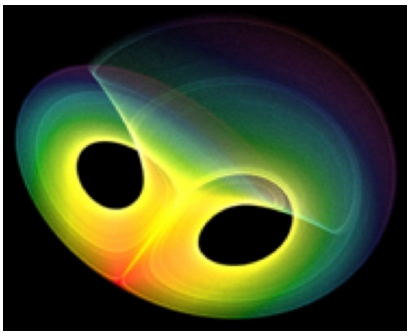
Before we move to the shift into algorithms, I want to point out a key property of equation-based systems that will differ in algorithmic ones.

As an example let us consider the Lorentz equations (Figs. 1a and 1b below). These have to do with atmosphere, but the physical details need not concern us. Notice the equations are really an updating rule. The left side in Fig. 1a denotes the rates of change in each of *X*, *Y*, and *Z*. Loosely, it denotes the direction the system is heading next. The right side tells us this is determined by two things: where the system now is, i.e. the current values of the three variables, *X*, *Y*, and *Z* (called the *state* of the system); and the rule, the formula, that connects the state to the updating terms on the left. Notice the state updates and changes constantly, but the rule by which it does this—the formula on the right

side—does not. It stays the same. Like a ball on a smooth curved surface, or a luge moving down a well-defined path, if you know where you are in Lorentz's system—what the current values of the state are—you can say exactly where you are going to go next.

Fig. 1b maps out the surface, or places the state visits, in 3 dimensions; often we still want to express such systems geometrically.

$$\frac{dX}{d\tau} = -\sigma X + \sigma Y$$
$$\frac{dY}{d\tau} = -XZ + rX - Y$$
$$\frac{dZ}{d\tau} = XY - bZ$$



Figs 1a and 1b. The Lorentz equations and their geometric expression

What is important here is that once the algebraic equation system is laid down, only the current state defines where the system goes next. Algorithmic systems as we will see generalize this.

The Arrival of Context

There is no exact date when the modern era of algorithms arrived, but the 1930s can serve as an arbitrary beginning point. That whole decade became fascinated with the idea that "methods" carried out by "computers" (human beings at that time) could be executed automatically by machines. Many people, including Norbert Wiener, Alan Turing, Claude Shannon, Prosper Eckert, John Mauchly, and John von Neumann, contributed to making real this notion with ideas and working designs, and the sequence of machines that followed progressed from purely mechanical ones to electromechanical ones, to electronic vacuum-tube ones. My purpose here is not to talk about the history of computation[5], it's to look at the workings of algorithms, and for this I want to focus on the ideas of Alan Turing. Turing was among the first to think systematically about algorithms and his ideas about how computation could be mechanized had a deep influence on how computers were later to develop.

In 1936 Alan Turing was a Fellow at King's College, Cambridge, and he was interested in a particular logical problem: Could propositions in mathematics in principle be "decided" as true or false by some universal method—some imaginary machine?[6] In a now classic paper Turing answered this

[5] There are many historical accounts: e.g. Goldstine, (1972), Campbell-Kelly *et al.* (2014), Waldrop (2001).

[6] Hilbert had famously posed this problem in 1928. Turing of course was also interested in which real numbers could be computed by finite methods.

by envisaging a machine that could carry out computational "methods" automatically. (The word algorithm was not yet in full currency.) Turing noticed that people who carried out calculations professionally broke their operations into small steps that could be carried out one step at a time. Often they made side notes about where they were in a complicated calculation and switched to new steps as these directed them. He was also fascinated by machines, ones such as typewriters and knitting machines, and he noticed that these could alter their configuration according to where they were in what they were doing: **Shift** after a period to start a new sentence; **Return** to produce a new line. Accordingly, Turing's machine would "read in" elementary instructions and carry these out step by step, but depending on where it was in the calculation it would shift automatically to other instructions as needed. It would change its "configuration" to do this.[7]

Turing showed that his imagined machine could carry out any computational "method" that humans could. And he showed further that his machine could be set up to be general. It could "read" in as data the set of instructions and configurations that defined any calculation method, and set itself up to execute these. It could therefore mimic any fed-in method. The calculation would change if the method were changed: his machine would be universal.

All this found its way eventually into the architecture of physical computers, though not in the exact form Turing envisaged. Instructions and configurations could be coded into "programs" that were stored in memory and could execute particular logical and arithmetic operations that were built into the physical design of the machine.

§

What is important for us is not the details of Turing's machine; it is the way he envisages methods being implemented. His imagined machine didn't necessarily carry out the same sort of instructions at each step, it could "decide" what to do next based upon some criterion. In modern terms we would say it—or the algorithm driving it—could perform "conditional branching." It could check some condition and based on this could change the set of instructions it was following and branch to another set. Turing set this branching up by the machine reading some symbol and changing its configuration. In modern computation we'd arrange this by the machine checking some form of if/then condition: *If* condition *P* is fulfilled, *then* execute instructions *T*; *if not*, *then* execute instructions *Q*. The condition could be quite general. The algorithm could track not just where it was in the computation, but a wider *context* for the computation: Is *X* currently larger than *Y*? Has array *H* been sorted? Are we conducting

[7] The instructions would be executed by the machine moving back and forth on the squares of a tape, "reading" symbols written on the tape squares one at a time, and applying pre-stored rules or instructions according to the symbol being read (e.g. "write a 0 on this square then move 3 squares to the left"). The symbols on the tape thus tracked the temporary progress of the computation. The symbol currently read could give instructions that might include a change in configuration for the next step. By writing the initial symbols on the tape (the input data), and cleverly choosing the rules or instructions for each configuration, and how the configurations might change, a given method or algorithm could be executed.

some subpart of the calculation? In other words, the algorithm could react to where it was in the overall computation. It could react to the computation itself.

In fact, the possibilities are wider than that. The computation could react to *any* wider context it chose to notice, and this might not be totally internal. In an automatic flight system it might include outside variables such as current airspeed and altitude being fed to the system. It might include whether a particular set of switches was currently open or closed. It might include the price of tea in China.

What I want to emphasize here is that the algorithmic instructions being carried can change by reacting to the context of the calculation, and this sets up a profound interplay between where the computation goes next and its current context. We can think of this interplay as a "conversation" with the current state of the solution and more generally with the outside world that is feeding information to it.

Where does this leave us? We now have two modes of expression for systems that unfold over time: an equation-based (or analytical) one [8] in which where you go next is dependent only on where you are now; or an algorithmic one that is also dependent on where you are now but also on the wider context of the system, a context that the system itself changes and constantly redoes.[9]

§

Algorithms and equation systems are not completely separate, there is quite a bit of overlap between them. Algorithms can often be put in equation form. (They have state equations and if-then clauses, and the latter can be implemented using temporary or flag variables with their corresponding equations.)[10] And an equation system becomes an algorithm when we think of "computing it out," that is, when we think of it not as a set of equalities but as a set of instructions.[11]

This high degree of equivalence between equations and algorithms means that properties of one system often apply to the other system, and it means that for a given situation we can often find both

---

[8] I am talking here of conventional algebraic equation systems in science and engineering. Other algebras used in modern science include linear algebra, Lie algebras, differential forms, twistors, etc.

[9] Of course, an equation-based system has some "context" too if we want to include the system's state as "context." In fact if we add a large number of variables, the context that this enlarged state provides could be quite complicated. Also we can stipulate that the form of the equation changes in different regimes, different regions of X, Y, Z space. So with some stretching, equation-based systems can in effect include context and thereby function as algorithms.

[10] For example we can set a flag = 1 or 0 depending on whether the condition is fulfilled or not, and jump to different instructions given the flag's value. So if we include the flag equations with the state equations we could say that algorithmic systems are equation based. But note the equation symbol here is merely a shorthand way of telling a computer instruction to put a particular value in a particular register for later use. (We could equally set our flag = 200.)

[11] Consider the statement: X = min (287, 342). This is an equation if we see it as the descriptive statement that X is equal to the smaller of 287 and 342, in this case 287. But it is an algorithm if we also see it as an instructive statement: figure out which of 287 and 342 is smaller, and set X equal to this.

an equation-based representation and an algorithmic one. But it doesn't mean the two systems are equal in efficiency. In most cases, it's simply easier to use one system rather than the other. (Try solving a sorting problem using equations.) This echoes the previous shift in that algebraic problems could often be translated into geometry and vice-versa. But for many problems, algebraic expression was simply more efficient, and more clear.

The high degree of equivalence doesn't mean the two systems are the same in character either. Equations express equivalence between measurable quantities: A typical algebraic problem might posit say a truss bridge with given parameters and loading on its deck; it would solve for the stresses on each member by equating the forces at each connection. A typical algorithmic problem would be more action oriented: *Enter* the current financial transaction into the database; *re-express* it in standard format; *confirm* it with at least 125 verifiers; *add it* to Blockchain AT45079-3b#; *repeat* for the next transaction. Equations work largely within a fixed frame (the bridge is given and doesn't change), they inhabit usually a world of stasis or equilibrium, and at best they allow smooth change. Algorithms work with actions, they inhabit a world that is not necessarily framed, that may change as they progress, and they proceed piece by piece.

Algorithms—and this is important—don't always involve equations. Consider the Sierpinski triangle. It is constructed by starting with a filled equilateral triangle and then connecting the midpoints of each side. This produces in the middle an inverted equilateral triangle. We excise this. This creates three new triangles at each apex, and we repeat the excising operation for each of these, proceeding *ad infinitum*. The process is recursive: the creation reproduces evermore its own creation. Notice it is purely geometric, it does not include equations. And it has no "computer," it proceeds purely as a thought process. In fact, in principle at least, algorithms don't require computers. An algorithm might say: Step 1: print 001; Step 2: return to Step 1. This would produce 001 001 001 001 001 indefinitely and this could be written by hand on a tablecloth or a clay tablet if either was infinitely large. Algorithms inhabit a world—a vast and mathematical one—of their own.[12] Edsgar Dijkstra once said, "Computer science is no more about computers than astronomy is about telescopes."

Algorithms can also proceed as purely natural processes. Nature creates embryos, replicates DNA structures, and hollows out river systems with natural mechanisms serving as its "computer." Its algorithms don't execute discrete-time logical steps; usually they are continuous in time, parallel in execution, and analog in character. In what follows we will take a broad view of algorithms: they may be computer based—or not.

What can algorithms do that equations can't easily? I want to show three main attributes.

---

[12] On this see Chaitin (2012) or Wolfram (2002).

What Algorithms Provide

I have said that algorithms are systems that respond to changing context. What then does context provide? Context allows algorithms to show *intelligent response*. Algorithms, as we saw, can express conditions and follow different sets of instructions accordingly. So in particular they lend themselves to conditional logic: *If A*, *B*, *F*, and *not-G* are currently true, *then* execute *R*, and *S*, and *T*. In other words they lend themselves to expressing systems of Boolean logic. In fact it would not be hard to imagine a system that consisted of purely Boolean actions. An algorithm then could "read" at any time the situation it finds itself in (the internal or external *A*, *B*, *F*, and *G*s) and respond with appropriate action.

This property may seem modest but actually it is a primitive form of "intelligence." Biologists define "intelligence" as the ability of an organism to recognize the situation it is in and act appropriately. Jellyfish, which have no central nervous systems, chemically sense nutrient material drifting near them and use a motor-nerve net to close around it. By "reading" their environment and reacting appropriately they are doing something intelligent. Algorithms "read" and react in much the same way, responding perhaps to a single if-then condition, or to a set of these that describe a complicated and changing context, or to full-blown artificially-intelligent conditions that can "recognize" spoken Mandarin say, and react with equivalent spoken English. This ability to monitor the changing "logic of the situation" and react accordingly is an extremely powerful attribute of algorithms.

This ability to read and respond by the way isn't a new theme. It has a long lineage. Primitive automata such as the moving figures of medieval church towers might "read" (encounter) a peg or stop that triggers some new action and new figures appear. Some centuries later, industrial machines might "read" (sense) a workpiece arriving and operate on it accordingly. Still later, Turing's imagined machine might read a symbol and change its configuration. And nowadays autonomous vehicles might "read" the road ahead for cars and pedestrians and take appropriate action. Something inside us finds this sort of "intelligence" fascinating. It is a pure machine creation, yet it signals to us unconsciously a thing "alive."

§

There is a second attribute of algorithms that connects with aliveness, and it's again a consequence of the interaction with context. Algorithms allow *processes* to be expressed. An algorithm can easily set up conditions that test whether a particular set of instructions has been completed, and depending on this, switch in another set of instructions to be completed. This lends itself to the expression of events—sets of steps to be completed—that trigger other events or inhibit other events.

In this way algorithms can switch in other algorithms that trigger and query other algorithms, each time executing a set of steps we can call an event. And things don't stop there. A system like this can switch in other processes or algorithms to execute some more detailed part of the whole. (We familiarly call these subroutines, or functions, or procedures.) So there can be processes nested within processes,

or processes within processes within processes *ad infinitum*, with all parts of the hierarchy creating further context and reacting to that. The architecture of these events defines a logic by which the system unfolds.

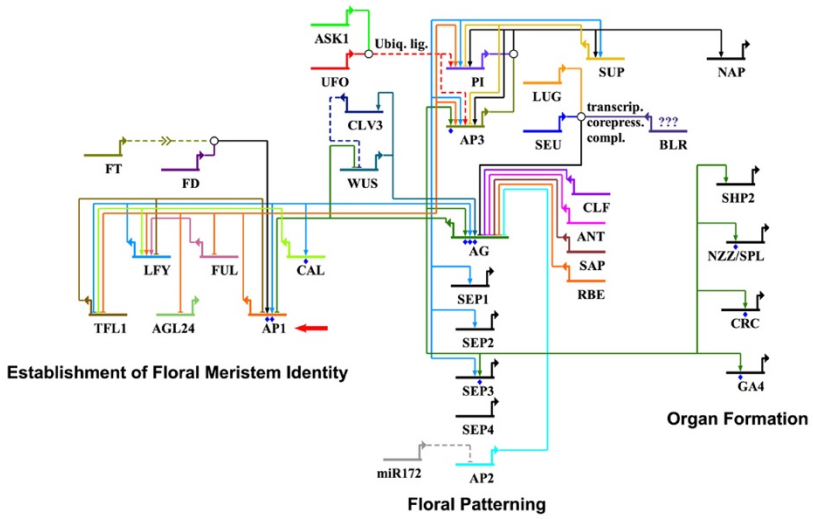


Fig. 2. Visual description of protein expression in the genetic regulatory network of a plant cellular system, showing genes expressing proteins that switch on or inhibit other genes.

It is by such structures that life works—that biology works. Nature's processes call each other, trigger each other, inhibit each other, are nested within each other. See figure 2.

Everything biological in fact is driven by process, and if we see these processes as algorithms they are in general parallel, probabilistic, highly interactive, and continuous in time. Biology describes such systems qualitatively by recognizing events and the networks of interaction between them, in embryonic development, say, or in the workings of the immune system. All these parts of biology are algorithmic, and this in turn means that biology has a well-defined logic, though not an equation-based one. The natural language for biology is algorithmic.

Constructive Means

There is a third attribute of algorithms I want to talk about. Algorithms are instructions for *constructing* something. A computational algorithm constructs a solution. Usually it does this by working on something—some element—locally, changing it or updating it or reorganizing it according to what its current value is, and moving to the next element. A finite-element algorithm computes out stresses on a complicated engineering structure by first dividing the structure into small simple-to-compute elements; it then starts from some end of the structure where the conditions are known, and calculates

its way across the structure, using what's been calculated in an element to calculate the next neighboring element. The calculation so constructed becomes a finished solution.

Sometimes the construction is a mathematical object, the Mandelbrot set, say. Sometimes the construction is an organizational schema, a phylogenetic tree derived from genomic similarities and differences among biological species. Sometimes the construction shows unforeseen qualities. In 1957 von Neumann designed in algorithmic terms an automaton that could reproduce itself. He realized that among other properties his automaton needed to contain the instructions for replicating further copies of itself, much as a human baby needs to contain the instructions (DNA) for replicating copies of itself. The instructions reproduced the instructions, in this sense they were recursive.

Sometimes the construction is evolutionary in nature. An algorithm often constructs an environment—or context—that it further reacts to and constructs from. This is the essence of agent-based models where objects or agents together create an outcome—an "ecology" if you like—the objects react to and thereby change. Agents' behaviors adapt or survive depending on how they perform in within the ecology of other agents' behaviors. Usually the objects and their possible behaviors are specified in advance, but sometimes novel objects themselves are created (constructed) as the evolution progresses. The genetic algorithm constantly creates novel objects for testing within the ecology of previous objects created. Such open-ended creation is awkward to reproduce via equations—Darwin's theory of novel species formation is algorithmic and doesn't lend itself to equation form. And occasionally with such open creation, some novel addition will change the overall game and the system will spontaneously undergo structural change.

In all of this type of work, algorithms give us the possibility to study formation. The researcher studies what generative process produces a given pattern and how this might vary with different algorithmic designs. So there is a back and forth between the pattern or structure formed and the algorithm that has formed it. The style becomes experimental: an algorithm produces some structure, and that structure feeds back to querying the algorithm that produced it. Steven Wolfram's classic study (2002) of the outcomes produced by cellular automata fits this procedure. Which versions produce a random output, and which don't? Which are poised in between order and chaos and how do these produce spatial patterns that interact in complicated ways? Wolfram goes back and forth continually between algorithmic designs and the structures they create.

Something similar happens in mathematical logic. When Turing explores whether we could tell in advance whether a given randomly chosen program will terminate in finite time or not, he imagines a machine that could accomplish this. You could feed it an algorithm to see if it terminates; the machine makes the decision and stops. He then constructs a use of this machine that leads to a logical

contradiction, so such a machine cannot exist.[13] Turing thus answers David Hilbert's Decidability question: Could there exist a method (or machine) that could automatically decide whether a given statement in mathematics was true or false? He shows there couldn't. His thinking is both constructive and deeply algorithmic.

If algorithms' hallmark is construction—or formation, if you prefer—what is the equation-based approach's? I would say it is manipulation. With equations we manipulate the system to arrive at some form we are seeking: some expression of a solution, some formula, some necessary condition, some mathematical structure, some sought-after demonstration of a truth contained in the system. We use manipulation to draw out these implications and in doing so gain clarity about why the needed form is valid. With algorithms we are also seeking to draw out implications but we cannot easily manipulate their instructions. Instead we allow the algorithm to execute and we observe it as it constructs some outcome, some implied structure, step by step. Insight follows if we explore closely how this happens. Equations allow us to think in terms of systems that can be manipulated to produce useful forms. Algorithms allow us to think in terms of systems that form structures we can investigate. Such investigation brings its own story and its own clarity.

Taking the three qualities I've talked about, we can say that algorithms allow us to model, think about, and think in terms of systems that react, that are event-driven, and that construct structures. Equations can do this too, but at best with difficulty. Above all, algorithms are about context and formation, so they bring not just a different style but different possibilities, in engineering, mathematics, and science itself.

On Nouns and Verbs

In fact the means of expression a science uses dictate to a high degree the possibilities it can look at. I said earlier that equation systems deal with *amounts* of something, whether that be velocity, or momentum, or mass, or electrical potential, or rates of change, or chemical concentration. The *x*'s and *y*'s must be quantities—algebra as I said is a form of arithmetic—and so they must be nouns. Given this observation, it follows that to use equations a field of interest must be reduced to nouns.

This has consequences: Consider economics. It became equation-based in Victorian times, and so in modern form it deals with prices, quantities produced, quantities consumed, rates of interest, rates of exchange, rates of inflation, trade surpluses, GDP, and so on. These are all nouns. It doesn't in its models deal with verbs such as invest, trade, explore, buy, sell, adapt, bring new products into being, invent, start companies. This I believe has had a distorting effect on economic theory. Theory these

[13] Turing imagines a procedure that feeds arbitrary programs into the machine and, if it decides they halt, executes them; if they don't halt the machine decides this and halts. Now he feeds into the machine this procedure itself. If it doesn't halt, the machine decides this and halts. A contradiction. If it does halt, the machine decides this and feeds it *de novo* to the machine; the procedure now cycles and doesn't halt. A contradiction. So no such machine can exist.

days in economics is excellent at resolving questions of *allocation*: how quantities of goods and levels of prices are determined within and across markets or trading regions. But it has little to say about questions of *formation*: how an economy emerges in the first place, how innovation works, how economic development takes place, how structural change happens (Arthur, 2014). These have been largely orphaned by economic theory and so they are left to case-based literary or historic description.

Algorithmic systems of course also deal with nouns, but par excellence they deal with verbs—with processes, with events triggering events. They allow verb-based or process description: split, adjust, compare, allow, inhibit, combine, birth a new entity.

We can thus divide the sciences into noun-based ones (19th century physics and chemistry or standard economics, for example) and verb-based sciences, which I will call procedural sciences. Biology is a procedural science.[14] It deals with actions that call in or inhibit further actions, whether in embryology, protein expression, epigenetics, genetic regulatory networks, or speciation. The key events here are algorithmic, though I don't mean they can be perfectly described as computer programs. I mean they can be represented as networks of events contributing to further events. The sciences that have emerged in recent decades—condensed matter physics, genomics, synthetic biology, computer science itself—are procedural, and this is very much at the center of the larger shift I am talking about.

Discussion

One family of possibilities the reader may have anticipated already. I said earlier that algorithms react to the context they find themselves in, and in doing so they modify that context. Complex systems also react to the context—the overall pattern—they find themselves in, and in doing so they modify that context. So is there a strong correspondence between algorithms and complex systems? I believe there is. Complexity studies systems whose elements react to the patterns—to the context—they together create. In doing so the elements update their actions or behavior, changing the pattern so that they must adjust anew. The elements may be cars on a road and the pattern they create traffic. If the traffic slows down, the cars behind slow down one by one: they react to the pattern the cars have together created and thus modify that pattern. Such adjustment, here a parallel one in continuous time, is algorithmic. It creates context and continually reacts to this anew.

Of course, quite often complex systems can be described by equations, especially if their context can be summarized by simple variables. This yields systems expressed as nonlinear dynamics, and we can plot geometrically the surface they move on and the various attractors they may fall into. But where the system is complicated, equations cease to work. Imagine modeling the formation of logjams. Logs

[14] For a process-based philosophy of biology see Nicholson and Dupré (2018). Rigorous philosophy based on process goes back to Alfred North Whitehead (1927), and process philosophy which descends from this often points to the distinction between "nouns and verbs" (Schleuter, 2020), Rescher (2000), Langley et al. (2016), Karonen (2018).

float down a river at random intervals, and at some juncture begin to pile up. They don't pile up higgledy-piggledy, they end up ordered into small parallel bunches pointing at angles to each other. This type of formation is easy to demonstrate via algorithms, almost impossible via equations.

The reader might feel uneasy that if we use a computer-based approach we are giving up the exactness of equations. Computer models are complicated, and they are often held to be more *ad-hoc*, more easily fudged, than equations. But the exactness of equations is an illusion. Equations need to be concise and to summarize, and to do this they often use coefficients or rate parameters, averages of how one part of a system affects another. "As soon as you write an equation it is wrong," said physicist Albert Tarantola, "because reducing a complex reality to an equation is just too simplistic a view of things" (quoted in Bailey 1996, p.29). With algorithms this may be true as well, but we can include more detail and proper heterogeneity, and where these matter algorithms are more accurate.

§

Admitting an algorithmic or process view of the world allows science to point to different qualities than the ones we have learned to take for granted. A few decades after Newton's achievement, enlightenment thinkers began to think of the world and of the science that describes it as possessing three qualities: Order, Predictability, and (usually) some sort of Harmony or Equilibrium. And the entire enlightenment project by and large bore this out—or chose systems to study that bore these out.

Algorithmic investigation does not bear these qualities out. In general algorithmic systems tend to be interrelated, parallel, and highly context dependent, and if their outcomes show order it tends to be complicated and open ended. Cloud streets (parallel bands of cumulous clouds created by convection), for example, can be modeled algorithmically, but they are neither perfectly ordered nor unordered. In general algorithmic systems are not predictable either. Wolfram (2002) shows that cellular automata don't have to be very complicated before it becomes impossible to predict their outcomes far ahead other than by computing them out. (Of course some equation systems, chaotic ones, share this property—Lorentz's system is not predictable in the long run.) And in general algorithmic systems do not settle into stasis or harmony either. Some do. But most are open ended, they never settle down but constantly build upon what has gone before or continually create new entities or new behavior, and thus show perpetual novelty. Stasis or equilibrium is a special case.

Algorithmic thinking, then, opens up a world that has unfamiliar properties: partial order, unpredictability, and perpetual novelty. It opens a world that is organic, action-based rather than object-based, not fully knowable, and strangely alive. It resists what architect Robert Venturi (1966) called "prim dreams of pure order" and gives us instead a world of "messy vitality."

§

Here is a question I mentioned earlier. We think of certain equation systems—Newton's equations, for example—as Science, as constituting "theory." To what extent can we call algorithms that describe systems Science? Can algorithms constitute theory? If I may insert a personal story, much of the thinking here came up because I had written a book called *The Nature of Technology* where I argued that at the center of the evolution of technology was an algorithm or a series of steps. I thought this was remarkable, but one of my friends who teaches at Oxford said, "You know, that's all very fine, I like your book. But it's not scientific." I said "What do you mean it's not scientific?" He said, "Well, what's missing are the equations." This bothered me and I started wondering, does science have to be equation-based at all?

To answer this question we can borrow from mathematician Gregory Chaitin's Algorithmic Information Theory (Chaitin 1990, 2006, 2012). This is a large subject and I'll extract just one central insight. If we have, say, data that show 0001 0001 0001 0001 *ad infinitum*, we can reduce this to an algorithm that says write three 0s then a 1, then repeat. The algorithm, in Chaitin's language, is a *compression* of the information of such a system and as such we can say it constitutes a *theory* of the system, just as Newton's system of equations is similarly a compression and constitutes a theory of planetary orbital information. Algorithms, used to describe systems are theory, they are a form of scientific expression. This means that biology and other procedural sciences are not waiting to be put in equation form to be declared science. They are expressible procedurally—expressible algorithmically. They are already properly scientific.

I have been talking mostly about algorithmic thinking, but I want to say a word here about the practical use of algorithms or computation in science. Historically, science has advanced via novel instruments and tools—novel ways of "seeing" such as telescopes, microscopes, X-ray devices—as much as by rational thought, and Douglas Robertson (2003) argues that the computer is a novel instrument for seeing, in fields as diverse as astronomy, mathematics, meteorology, physics, and geology. Robertson tells us that improvements in the resolution of astronomical objects by optical telescopes from Galileo's day until now have been of the order of 10^2; improvements due to adaptive algorithms have contributed another 10^3 in resolution—more than that of telescopes. Indeed, some fields like genomics or modern high-energy physics or bioinformatics would be impossible without computation.

Computation certainly has given science an enormous boost. But how much of this is due to algorithms—the methods that drive the physical machinery—and how much to the increases in the power of that machinery itself? My answer is that the two cannot be quite separated. We now have computerized tomography, computational seismology, artificial intelligence, deep-learning algorithms, evolutionary algorithms, new optimization methods, finite-element methods, statistical estimation methods, fast Fourier transforms, econometric methods, data-handling methods, to mention only some of a dazzling array of techniques. All these are "methods" and therefore algorithms, but all came

into being—all were made possible—by advances in computing power. Algorithms and computing power push forward together. Notice by the way that most of these algorithmic examples are based on equational systems, specially developed to take advantage of fast computers, which tells me quite cheerfully that equation systems still have much to offer us and are by no means dead.

Conclusion

Science has changed since the turn of the last century. It has become deeply curious about matters of formation. How did the brain evolve? How did continents form? By what steps did language arise? How did multicellular organisms arise? Can a self-reproducing machine be constructed? Science has shifted from deducing the implications of closed, given, formal structures to exploring the implications of open, evolving, generative structures.

What are the drivers of this shift? Well, certainly modern biology and its constituent subjects—molecular biology, genomics, and bioinformatics—are, along with parts of modern physics. And so too is computation, or more precisely the means by which computation is organized: algorithms. But above all, we are starting to recognize that the systems of the world, whether physical or biological or social or intellectual, are rarely passive and fixed in structure. Nearly always they are changing and evolving and re-forming, and discovering new structures as they do so. Such systems do not behave merely in a way similar to computational algorithms, they behave directly algorithmically.

And so algorithms bring not just a difference in calculational possibilities, but a difference in thinking, one that allows a wider set of possibilities. Algorithms are about systems that constantly "read" their environment and react to this; about systems that unfold in a cascade of events rather than ones that take on different values as they glide smoothly on a static surface; about systems that are functional and generative, rather than ones merely changing in value and tightly contained. They are—and this is why they fit with the new science—above all about formation.

This shift toward algorithmic expression and the thinking that goes with them has been gradual, almost imperceptible, but it mirrors almost perfectly the earlier shift from geometry to equations. We are not about to see the end of equations; indeed some, like Maxwell's equations, are the glory of science. They will and should remain an important means of expression in science, but they are no longer the sole means. Algorithms bring a new world of possibilities. They are a new means of *expression* for science—a language science uses to think and reason, to explore and explain, to construct and create; and they open a world that is organic, contingent, and alive.

References

Arthur, W. Brian, *The Nature of Technology: What it Is and How it Evolves*, The Free Press, NY, 2009.

Arthur, W. Brian, "Complexity Economics: A different Framework for Economic Thought," in *Complexity and the Economy,* by W. B. Arthur, Oxford Univ. Press, New York, 2014.

Bailey, James, *After Thought: The Computer Challenge to Human Intelligence*, Basic Books, New York, 1996.

Campbell-Kelly, Martin, W. Aspray, N. Ensmenger, J.R. Jost, *Computer: A History of the Information Machine*, (Sloan Technology Series), Routledge, New York, 2018 (2014).

Chaitin, Gregory, *Information, Randomness, and Incompleteness: Papers on Algorithmic Information Theory*, World Scientific, Singapore, 1990.

Chaitin, Gregory, *Meta Math!* Vintage Books, New York, 2006.

Chaitin, Gregory, *Proving Darwin: Making Biology Mathematical*, Pantheon Books, New York, 2012.

Devlin, Keith, *Finding Fibonacci*, Princeton Univ. Press, Princeton, 2017.

Gaukroger, Stephen*, Descartes, an Intellectual Biography*, Clarendon Press, Oxford, 1995.

Goetzmann, William, *Fibonacci and the Financial Revolution*, NBER Working Paper 10352, 2004.

Goldstine, Herman, *The Computer from Pascal to von Neumann.* Princeton Univ. Press, Princeton, 1972.

Hodges, Andrew, *Alan Turing: The Enigma*, Vintage Books, New York, 2014 (1983).

Kaaronen, Roope, "Reframing Tacit Human-Nature Relations: An Inquiry into Process Philosophy and the Philosophy of Michael Polanyi," *Environmental Values,* 27, 2018.

Kant, Immanuel, *Metaphysical Foundations of Natural Science*, Cambridge Univ. Press, 1970 (1786).

Kline, Morris, *Mathematics: The Loss of Certainty,* Oxford Univ. Press, 1980.

Langley, Ann, and Haridimos Tsoukas, *SAGE Handbook of Process Organization Studies.* SAGE Publications, London, 2016

Mahony, Michael, "The Beginnings of Algebraic Thought in the Seventeenth Century," originally in German as "Die Anfänge der algebraischen Denkweise im 17. Jahrhundert," *Strukturgeschichte der Naturwissenschaften* 1, 1971.

Nicholson, D.J., and J. Dupré. *Everything Flows.* Oxford University Press, New York, 2018.

Petzold, Charles, *The Annotated Turing*, Wiley, Indianapolis, 2008.

Robertson, Douglas S., *Phase Change: The Computer Revolution in Science and Mathematics.* Oxford Univ. Press, New York, 2003.

Schleuter, Maja, personal communication, March 15, 2020.

Schuster, John, *Descartes and the Scientific Revolution, 1681-1634*. Princeton Univ. Ph.D thesis, 1977.

Struik, Dirk J., *A Concise History of Mathematical Thought*, Dover, New York, 1948.

Thompson, D'Arcy, *On Growth and Form,* Cambridge Univ. Press, Cambridge, UK, 1917.

Turing, Alan, "On computable numbers with an application to the Entscheidungsproblem," *Proc. of the London Mathematical Society*, Vol. 43, 1936.

Venturi, Robert, *Complexity and Contradiction in Architecture.* Doubleday, New York, 1966.

Von Neumann, John. *Theory of Self-Reproducing Automata* (A. Burks, ed.), Univ. Illinois Press, 1966.

Waldrop, M. Mitchell, *The Dream Machine*, Viking Press, New York, 2001.

Whitehead, Alfred N., *Process and Reality.* The Gifford Lectures delivered at the University of Edinburgh, 1927-28. The Free Press, New York, 1978.

Wolfram, Steven, *A New Kind of Science*, Wolfram Media, Champaign, IL, 2002.