1 What Is an Algorithm?

aus
Ed Finn,
What algorithms want. Imagination in the age of computing.
Cambridge: MIT Press 2017

If we want to live with the machine, we must understand the machine, we must not worship the machine.
Norbert Wiener[1]

Rise of the Culture Machines

Sometime in the late 2000s, our relationship with computers changed. We began carrying devices around in our pockets, peering at them at the dinner table, muttering quietly to them in the corner. We stopped thinking about hardware and started thinking about apps and services. We have come not just to use but to *trust* computational systems that tell us where to go, whom to date, and what to think about (to name just a few examples). With every click, every terms of service agreement, we buy into the idea that big data, ubiquitous sensors, and various forms of machine learning can model and beneficially regulate all kinds of complex systems, from picking songs to predicting crime. Along the way, an old word has become new again: the algorithm. Either overlooked or overhyped, the algorithm is rarely taken seriously as a key term in the cultural work that computers do for us. This book takes that word apart and puts it back together again, showing how algorithms function as culture machines that we need to learn how to read and understand.

Algorithms are everywhere. They already dominate the stock market, compose music, drive cars, write news articles, and author long mathematical proofs—and their powers of creative authorship are just beginning to take shape. Corporations jealously guard the black boxes running these assemblages of data and process. Even the engineers behind some of the most successful and ubiquitous algorithmic systems in the

world—executives at Google and Netflix, for example—admit that they understand only some of the behaviors their systems exhibit. But their rhetoric is still transcendent and emancipatory, striking many of the same techno-utopian notes as the mythos of code as magic when they equate computation with transformational justice and freedom. The theology of computation that Ian Bogost identified is a faith militant, bringing the gospel of big data and disruption to huge swaths of society.

This is the context in which we use algorithms today: as pieces of quotidian technical magic that we entrust with booking vacations, suggesting potential mates, evaluating standardized test essays, and performing many other kinds of cultural work. Wall Street traders give their financial "algos" names like Ambush and Raider, yet they often have no idea how their money-making black boxes work.[2] As a keyword in the spirit of cultural critic Raymond Williams,[3] the word algorithm frequently encompasses a range of computational processes including close surveillance of user behaviors, "big data" aggregation of the resulting information, analytics engines that combine multiple forms of statistical calculation to parse that data, and finally a set of human-facing actions, recommendations, and interfaces that generally reflect only a small part of the cultural processing going on behind the scenes. Computation comes to have a kind of presence in the world, becoming a "thing" that both obscures and highlights particular forms of what Wendy Hui Kyong Chun calls "programmability," a notion we will return to in the guise of computationalism below.[4]

It is precisely this protean nature of computation that both troubles and attracts us. At some times computational systems appear to conform to that standard of discrete "thingness," like the *me* of Sumerian myth or a shiny application button on a smartphone screen. At other moments they are much harder to distinguish from broader cultural environments: to what extent are spell-check programs changing diction and grammatical choices through their billions of subtle corrections, and how do we disentangle the assemblage of code, dictionaries, and grammars that underlie them? While the cultural effects and affects of computation are complex, these systems function in the world through instruments designed and implemented by human beings. In order to establish a critical frame for reading cultural computation, we have to begin with those instruments, jammed together in the humble vessel of the algorithm.

Our look at *Snow Crash* revealed the layers of magic, "sourcery," and structured belief that underpin the facade of the algorithm in culture today. Now we turn to the engineers and computer scientists who implement computational systems. Rooted in computer science, this version of the algorithm relies on the history of mathematics. An algorithm is a recipe, an instruction set, a sequence of tasks to achieve a particular calculation or result, like the steps needed to calculate a square root or tabulate the Fibonacci sequence. The word itself derives from Abū ʿAbdallāh Muḥammad ibn Mūsā al-Khwārizmī, the famed ninth-century CE mathematician (from whose name algebra is also derived). *Algorismus* was originally the process for calculating Hindu-Arabic numerals. Via al-Kwarizmi, the algorithm was associated with the revolutionary concepts of positional notation, the decimal point, and zero.

As the word gained currency in the centuries that followed, "algorithm" came to describe any set of mathematical instructions for manipulating data or reasoning through a problem. The Babylonians used some of the first mathematical algorithms to derive square roots and factor numbers.[5] Euclid devised an algorithm for taking two numbers and finding the greatest common divisor they share. Throughout this evolution, the algorithm retained an essential feature that will soon become central to the story: it just works. That is to say, an algorithm reliably delivers an expected result within a finite amount of time (except, perhaps, for those edge cases that fascinate mathematicians and annoy engineers).

Historian Nathan Ensmenger recounts how the academic discipline of computer science coalesced only after its advocates embraced the concept of the algorithm, with one of the field's founders, Donald Knuth, tracing the field's origins to al-Khwarizmi in his seminal textbook *The Art of Computer Programming*.[6] The algorithm was an ideal object of study, both easily grasped and endlessly puzzling:

> By suggesting that the algorithm was as fundamental to the technical activity of computing as Sir Isaac Newton's laws of motion were to physics, Knuth and his fellow computer scientists could claim full fellowship with the larger community of scientists.[7]

And yet, as mathematician Yiannis Moschovakis points out, Knuth's argument about what algorithms actually are is an extremely rare instance where the question is foregrounded.[8] For computer scientists the term

remains more of an intuitive, unexamined notion than a delineated logical concept grounded in a mathematical theory of computation.

Thanks in large part to Knuth, the algorithm today is a fundamental concept in computer science, an intellectual keystone typically covered in the introductory Algorithms and Data Structures course for undergraduate majors. Algorithms represent repeatable, practical solutions to problems like factoring a number into its smallest prime number components or finding the most efficient pathway through a network. The major focus for contemporary algorithmic research is not whether they work but how efficiently, and with what tradeoffs in terms of CPU cycles, memory, and accuracy.

We can distill this pragmatic approach to algorithms down to a single PowerPoint slide. Robert Sedgewick, a leading researcher on computational algorithms, also happened to teach the version of Algorithms and Data Structures that I took as an undergraduate; he calls the algorithm a "method for solving a problem" in his widely circulated course materials.[9] This is what I term the *pragmatist's definition*: an engineer's notion of algorithms geared toward defining problems and solutions. The pragmatist's definition grounds its truth claim in utility: algorithms are fit for a purpose, illuminating pathways between problems and solutions. This is the critical frame that dominates the breakout rooms and workstations of engineers at Google, Apple, Amazon, and other industry giants. As Google describes them: "Algorithms are the computer processes and formulas that take your questions and turn them into answers."[10] For many engineers and technologists, algorithms are quite simply the work, the medium of their labor.

The pragmatic definition lays bare the essential politics of the algorithm, its transparent complicity in the ideology of instrumental reason that digital culture scholar David Golumbia calls out in his critique of computation.[11] Of course this is what algorithms do: they are methods, inheriting the inductive tradition of the scientific method and engineering from Archimedes to Vannevar Bush. They solve problems that have been identified as such by the engineers and entrepreneurs who develop and optimize the code. But such implementations are never just code: a method for solving a problem inevitably involves all sorts of technical and intellectual inferences, interventions, and filters.

As an example, consider the classic computer science problem of the traveling salesman: how can one calculate an efficient route through a geography of destinations at various distances from one another? The question has many real-world analogs, such as routing UPS drivers, and indeed that company has invested hundreds of millions of dollars in a 1,000-page algorithm called ORION that bases its decisions in part on traveling salesman heuristics.[12] And yet the traveling salesman problem imagines each destination as an identical point on a graph, while UPS drop-offs vary greatly in the amount of time they take to complete (hauling a heavy package up with a handcart, say, or avoiding the owner's terrier). ORION's algorithmic model of the universe must balance between particular computational abstractions (each stop is a featureless, fungible point), the lived experience and feedback of human drivers, and the data the company has gathered about the state of the world's stop signs, turn lanes, and so on. The computer science question of optimizing paths through a network must share the computational stage with the autonomy of drivers, the imposition of quantified tracking on micro-logistical decisions like whether to make a right or left turn, and the unexpected interventions of other complex human systems, from traffic jams to pets.

ORION and its 1,000-page "solution" to this tangled problem is, of course, a process or system in continued evolution rather than an elegant equation for the balletic coordination of brown trucks. Its equations and computational models of human behavior are just one example among millions of algorithms attempting to regularize and optimize complex cultural systems. The pragmatist's definition achieves clarity by constructing an edifice (a cathedral) of tacit knowledge, much of it layered in systems of abstraction like the traveling salesman problem. At a certain level of cultural success, these systems start to create their own realities as well: various players in the system begin to alter their behavior in ways that short-circuit the system's assumptions. Internet discussion boards catalog complaints about delivery drivers who do not bother to knock and instead leave door tags claiming that the resident was not at home. These shortcuts work precisely because they are invisible to systems like ORION, allowing the driver to save valuable seconds and perhaps catch up on all those other metrics that *are* being tracked on a hectic day when the schedule starts to slip.

Many of the most powerful corporations in existence today are essentially cultural wrappers for sophisticated algorithms, as we will see in the following chapters. Google exemplifies a company, indeed an entire worldview, built on an algorithm, PageRank. Amazon's transformational algorithm involved not just computation but logistics, finding ways to outsource, outmaneuver, and outsell traditional booksellers (and later, sellers of almost every kind of consumer product). Facebook developed the world's most successful social algorithm for putting people in contact with one another. These are just a few examples of powerful, pragmatic, lucrative algorithms that are constantly updated and modified to cope with the messy cultural spaces they attempt to compute.

We live, for the most part, in a world built by algorithmic pragmatists. Indeed, the ambition and scale of corporate operations like Google means that their definitions of algorithms—what the problems are, and how to solve them—can profoundly change the world. Their variations of pragmatism then inspire elaborate responses and counter-solutions, or what communication researcher Tarleton Gillespie calls the "tacit negotiation" we perform to adapt ourselves to algorithmic systems: we enunciate differently when speaking to machines, use hashtags to make updates more machine-readable, and describe our work in search engine-friendly terms.[13]

The tacit assumptions lurking beneath the pragmatist's definition are becoming harder and harder to ignore. The apparent transparency and simplicity of computational systems are leading many to see them as vehicles for unbiased decision-making. Companies like UpStart and ZestFinance view computation as a way to judge financial reliability and make loans to people who fail more traditional algorithmic tests of credit-worthiness, like credit scores.[14] These systems essentially deploy algorithms to counter the bias of other algorithms, or more cynically to identify business opportunities missed by others. The companies behind these systems are relatively unusual, however, in acknowledging the ideological framing of their business plans, and explicitly addressing how their systems attempt to judge "character."

But if these are reflexive counter-algorithms designed to capitalize on systemic inequities, they are responding to broader cultural systems that typically lack such awareness. The computational turn means that many algorithms now reconstruct and efface legal, ethical, and perceived reality according to mathematical rules and implicit assumptions that are shielded

from public view. As legal ethicist Frank Pasquale writes about algorithms for evaluating job candidates:

> Automated systems claim to rate all individuals the same way, thus averting discrimination. They may ensure some bosses no longer base hiring and firing decisions on hunches, impressions, or prejudices. But software engineers construct the datasets mined by scoring systems; they define the parameters of data-mining analyses; they create the clusters, links, and decision trees applied; they generate the predictive models applied. Human biases and values are embedded into each and every step of development. Computerization may simply drive discrimination upstream.[15]

As algorithms move deeper into cultural space, the pragmatic definition gets scrutinized more closely according to critical frames that reject the engineering rubric of problem and solution, as Pasquale, Golumbia, and a growing number of algorithmic ethics scholars have argued. The cathedral of abstractions and embedded systems that allow the pragmatic algorithms of the world to flourish can be followed down to its foundations in symbolic logic, computational theory, and cybernetics, where we find a curious thing among that collection of rational ideas: desire.

From Computation to Desire

What are the truth claims underlying the engineer's problems and solutions, or the philosophy undergirding the technological magic of sourcery? They depend on the protected space of computation, the logical, procedural, immaterial space where memory and process work according to very different rules from material culture. The pragmatist's approach gestures toward, and often depends on, a deeper philosophical claim about the nature of the universe. We need to understand that claim as the grounding for the notion of "effective computability," a transformational concept in computer science that fuels algorithmic evangelism today. In her book *My Mother Was a Computer*, media theorist N. Katherine Hayles labels this philosophical claim the Regime of Computation.[16] This is another term for what I sometimes refer to as the age of the algorithm: the era dominated by the figure of the algorithm as an ontological structure for understanding the universe. We can also think of this as the "computationalist definition," which extends the pragmatist's notion of the algorithm and informs the core business models of companies like Google and Amazon.

In its softer version, computationalism argues that algorithms have no ontological claim to truly describing the world but are highly effective at solving particular technical problems. The engineers are agnostic about the universe as a system; all they care about is accurately modeling certain parts of it, like the search results that best correspond to certain queries or the books that users in Spokane, Washington, are likely to order today. As Pasquale and a host of other digital culture critics from Jaron Lanier to Evgeny Morozov have argued, even the implicit claims to efficiency and "good-enough" rationalism at the heart of the engineer's definition of algorithms have a tremendous impact on policy, culture, and the practice of everyday life, because the compromises and analogies of algorithmic approximations tend to efface everything that they do not comprehend.[17]

The expansion of the rhetoric of computation easily bleeds into what Hayles calls the "hard claim" for computationalism. In this argument algorithms do not merely describe cultural processes with more or less accuracy: those processes are themselves computational machines that can be mathematically duplicated (given enough funding). According to this logic it is merely a matter of time and applied science before computers can simulate election outcomes or the future price of stocks to *any desired degree* of accuracy. Computer scientist and polymath Stephen Wolfram lays out the argument in his ambitious twenty-year undertaking, *A New Kind of Science*:

> The crucial idea that has allowed me to build a unified framework for the new kind of science that I describe in this book is that just as the rules for any system can be viewed as corresponding to a program, so also its behavior can be viewed as corresponding to a computation.[18]

Wolfram's principle of computational equivalence makes the strong claim that all complex systems are fundamentally computational and, as he hints in the connections he draws between his work and established fields like theoretical physics and philosophy, he believes that computationalism offers "a serious possibility that [a fundamental theory for the universe] can actually be found."[19] This notion that the computational metaphor could unlock a new paradigm of scientific inquiry carries with it tremendous implications about the nature of physical systems, social behavior, and consciousness, among other things, and at its most extreme serves as an ideology of transcendence for those who seek to use computational systems to model and understand the universe.

Citing Wolfram and fellow computer scientists Harold Morowitz and Edward Fredkin, Hayles traces the emergence of an ideology of universal computation based on the science of complexity: if the universe is a giant computer, it is not only efficient but intellectually necessary to develop computational models for cultural problems like evaluating loan applications or modeling consciousness. The models may not be perfect now but they will improve as we use them, because they employ the same computational building blocks as the system they emulate. On a deeper level, computationalism suggests that our knowledge of computation will answer many fundamental questions: computation becomes a universal solvent for problems in the physical sciences, theoretical mathematics, and culture alike. The quest for knowledge becomes a quest for computation, a hermeneutics of modeling.

But of course models always compress or shorthand reality. If the anchor point for the pragmatist's definition of the algorithm is its indefinable flexibility based on tacit understanding about what counts as a problem and a solution, the anchor point here is the notion of abstraction. The argument for computationalism begins with the Universal Turing Machine, mathematician Alan Turing's breathtaking vision of a computer that can complete any finite calculation simply by reading and writing to an infinite tape marked with 1s and 0s, moving the tape forward or backward based on the current state of the machine. Using just this simple mechanism one could emulate any kind of computer, from a scientific calculator finding the area under a curve to a Nintendo moving Mario across a television screen. In other words, this establishes a computational "ceiling" where any Turing computer can emulate any other: the instructions may proceed more slowly or quickly, but are mathematically equivalent.

The Universal Turing Machine is a thought experiment that determines the bounds of what is computable: Turing and his fellow mathematician Alonzo Church were both struggling with the boundary problems of mathematics. In one framing, posed by mathematician David Hilbert, known as the *Entscheidungsproblem*, the question is whether it's possible to predict when or if a particular program will halt, ending its calculations with or without an answer. Their responses to Hilbert, now called the Church–Turing thesis, define algorithms for theorists in a way that is widely accepted but ultimately unprovable: a calculation with natural numbers, or what most of us know as whole numbers, is "effectively computable" (that is,

given enough time and pencils, a human could do it) only if the Universal Turing Machine can do it. The thesis uses this informal definition to unite three different rigorous mathematical theses about computation (Turing machines, Church's lambda calculus, and mathematician Kurt Gödel's concept of recursive functions), translating their specific mathematical claims into a more general boundary statement about the limits of computational abstraction.

In another framing, as David Berlinski argues in his mathematical history *The Advent of the Algorithm*, the computability boundary that Turing, Gödel, and Church were wrestling with was also an investigation into the deep foundations of mathematical logic.[20] Gödel proved, to general dismay, that it was impossible for a symbolic logical system to be internally consistent and provable using only statements within the system. The truth claim or validation of such a system would always depend on some external presumption or assertion of logical validity: turtles all the way down. Church grappled with this problem and developed the lambda calculus, a masterful demonstration of abstraction that served as the philosophical foundation for numerous programming languages decades after his work.[21] As Berlinski puts it, Turing had "an uncanny and almost unfailing ability to sift through the work of his time and in the sifting discern the outlines of something far simpler than the things that other men saw."[22] In other words, he possessed a genius for abstraction, and his greatest achievement in this regard was the Turing machine.

Turing's simple imaginary machine is an elegant mathematical proof for universal computation, but it is also an ur-algorithm, an abstraction generator. The mathematical equivalence of Church and Turing's work quickly suggested that varying proofs of effective computability (there are now over thirty) all gesture toward some fundamental universal truth. But every abstraction has a shadow, a puddled remainder of context and specificity left behind in the act of lifting some idea to a higher plane of thought. The Turing machine leaves open the question of what "effectively computable" might really mean in material reality, where we leave elegance and infinite tapes behind. As it has evolved from a thought experiment to a founding tenet of computationalism (and the blueprint for the computational revolution of the twentieth and twenty-first centuries), the Church–Turing thesis has developed a gravitational pull, a tug many feel to organize the universe according to its logic. The concept of universal computation

encodes at its heart an intuitive notion of "effective": achievable in a finite number of steps, and reaching some kind of desired result. From the beginning, then, algorithms have encoded a particular kind of abstraction, the *abstraction of the desire for an answer*. The spectacular clarity and rigor of these formative proofs in computation exists in stark contrast to the remarkably ill-defined way that the term is deployed in the field of computer science and elsewhere.[23]

This desire encoded in the notion of effectiveness is typically obscured in the regime of computation, but the role of abstraction is celebrated. The Universal Turing Machine provides a conceptual platform for uniting all kinds of computing: algorithms for solving a set of problems in particle physics might suddenly be useful in genetics; network analysis can be deployed to analyze and compare books, business networks, and bus systems. Abstraction itself is one of the most powerful tools the Church–Turing thesis—and computation in general—gives us, enabling platform-agnostic software and the many metaphors and visual abstractions we depend on, like the desktop user interface.

Abstraction is the ladder Wolfram et al. use to climb from particular computational systems to the notion of universal computation. Many complex systems demonstrate computational features or appear to be computable. If complex systems are themselves computational Turing Machines, they are therefore equivalent: weather systems, human cognition, and most provocatively the universe itself.[24] The grand problems of the cosmos (the origins thereof, the relationship of time and space) and the less grand problems of culture (box office returns, intelligent web searching, natural language processing) are irreducible but also calculable: they are not complicated problems with simple answers but rather simple problems (or rulesets) that generate complicated answers. These assumptions open the door to a *mathesis universalis*, a language of science that the philosophers Gottfried Wilhelm Leibniz, René Descartes, and others presaged as a way to achieve perfect understanding of the natural world.[25] This perfect language would exactly describe the universe through its grammar and vocabulary, becoming a new kind of rational magic for scientists that would effectively describe and be the world.

Effective computability continues to be an alluring, ambiguous term today, a fault line between the pragmatist and computationalist definition of algorithms. I think of this as computation's first seduction, rooted at the

heart of the Church–Turing thesis. It has expanded its sway with the growth of computing power, linking back to the tap root of rationalism, gradually becoming a deeper, more romantic mythos of a computational ontology for the universe. The desire to make the world effectively calculable drives many of the seminal moments of computer history, from the first ballistics computers replacing humans in mid-century missile defense to Siri and the Google search bar.[26] It is the ideology that underwrites the age of the algorithm, and its seductive claims about the status of human knowledge and complex systems in general form the central tension in the relationship between culture and culture machines.

To understand the consequences of effective computability, we need to follow three interwoven threads as the implications of this idea work themselves out across disciplines and cultural fields: cybernetics, symbolic language, and technical cognition.

Thread 1: Embodying the Machine

"Effective computability" is an idea with consequences not just for our conception of humanity's place in the universe but how we understand biological, cultural, and social systems. Leibniz's vision of a *mathesis universalis* is seductive because it promises that a single set of intellectual tools can make all mysteries accessible, from quantum mechanics to the circuits inside the human brain. After World War II, a new field emerged to pursue that promise, struggling to align mathematics and materiality, seeking to map out direct correlations between computation and the physical and social sciences. In its heyday cybernetics, as the field was known, was a sustained intellectual argument about the place of algorithms in material culture—a debate about the politics of implementing mathematical ideas, or claiming to find them embodied, in physical and biological systems.

The polymathic mathematician Norbert Wiener published the founding text of this new discipline in 1949, calling it *Cybernetics; or Control and Communication in the Animal and the Machine*. Wiener names Leibniz the patron saint of cybernetics: "The philosophy of Leibniz centers about two closely related concepts—that of a universal symbolism and that of a calculus of reasoning."[27] As the book's title suggests, the aim of cybernetics in the 1940s and 1950s was to define and implement those two ideas: an intellectual system that could encompass all scientific fields, and a means

of quantifying change within that system. Using them, the early cyberneticians sought to forge a synthesis between the nascent fields of computer science, information theory, physics, and many others (indeed, Wiener nominated his patron saint in part as the last man to have "full command of all the intellectual activity of his day").[28] The vehicle for this synthesis was, intellectually, the field of information theory and the ordering features of communication between different individual and collective entities, and pragmatically, the growing power of mechanical and computational systems to measure, modulate, and direct such communications.

On a philosophical level, Wiener's vision of cybernetics depended on the transition from certainty to probability in the twentieth century.[29] The advances of Einsteinian relativity and quantum mechanics suggested that uncertainty, or indeterminacy, was fundamental to the cosmos and that observation always affected the system being observed. This marked the displacement of a particular rationalist ideal of the Enlightenment, the notion that the universe operated by simple, all-powerful laws that could be discovered and mastered. Instead, as the growing complexity of mathematical physics in the twentieth and twenty-first centuries has revealed, the closer we look at a physical system, the more important probability becomes. It is unsettling to abandon the comfortable solidity of a table, that ancient prop for philosophers of materialism, and replace it with a probabilistic cloud of atoms. And yet only with probability—more important, a language of probability—can we begin to describe our relativistic universe.

But far more unsettling, and the central thesis of the closely allied field of information theory, is the notion that probability applies to information as much as to material reality. By framing information as uncertainty, as surprise, as unpredicted new data, mathematician Claude Shannon created a quantifiable measurement of communication.[30] Shannon's framework has informed decades of work in signal processing, cryptography, and several other fields, but its starkly limited view of what counts has become a major influence in contemporary understandings of computational knowledge. This measurement of information is quite different from the common cultural understanding of knowledge, though it found popular expression in cybernetics, particularly in Wiener's general audience book *The Human Use of Human Beings*. This is where Wiener lays one of the cornerstones for the cathedral of computation: "To live effectively is to live with adequate

information. Thus, communication and control belong to the essence of man's inner life, even as they belong to his life in society."[31] In its limited theoretical sense, information provided a common yardstick for understanding any kind of organized system; in its broader public sense, it became the leading edge of computationalism, a method for quantifying patterns and therefore uniting biophysical and mathematical forms of complexity.

As Wiener's quote suggests, the crucial value of information for cybernetics was in making decisions.[32] Communication and control became the computational language through which biological systems, social structures, and physics could be united. As Hayles argues in *How We Became Posthuman*, theoretical models of biophysical reality like the early McCulloch–Pitts Neuron (which the logician Walter Pitts proved to be computationally equivalent to a Turing machine) allowed cybernetics to establish correlations between computational and biological processes at paradigmatic and operational levels and lay claim to being what informatics scholar Geoffrey Bowker calls a "universal discipline."[33] Via cybernetics, information was the banner under which "effective computability" expanded to vast new territories, first presenting the tantalizing prospect that Wolfram and others would later reach for as universal computation.[34] As early as *The Human Use of Human Beings*, Wiener popularized these links between the Turing machine, neural networks, and learning in biological organisms, work that is now coming to startling life in the stream of machine learning breakthroughs announced by the Google subsidiary DeepMind over the past few years.

This is Wiener ascending the ladder of abstraction, positioning cybernetics as a new Liebnitzian *mathesis universalis* capable of uniting a variety of fields. Central to this upper ascent is the notion of homeostasis, or the way that a system responds to feedback to preserve its core patterns and identity. A bird maintaining altitude in changing winds, a thermostat controlling temperature in a room, and the repetition of ancient myths through the generations are all examples of homeostasis at work. More provocatively, Wiener suggests that homeostasis might be the same thing as identity or life itself, if "the organism is seen as message. Organism is opposed to chaos, to disintegration, to death, as message is to noise."[35] This line of argument evolved into the theory of autopoiesis proposed by philosophers Humberto Maturana and Francisco Varela in the 1970s, the second wave of cybernetics which adapted the pattern-preservation of homeostasis more

fully into the context of biological systems. Describing organisms as information also suggests the opposite, that information has a will to survive, that as Stewart Brand famously put it, "information wants to be free."[36]

Like Neal Stephenson's programmable minds, like the artificial intelligence researchers who seek to model the human brain, this notion of the organism as message reframes biology (and the human) to exist at least aspirationally within the boundary of effective computability. Cybernetics and autopoiesis lead to complexity science and efforts to model these processes in simulation. Mathematician John Conway's game of life, for

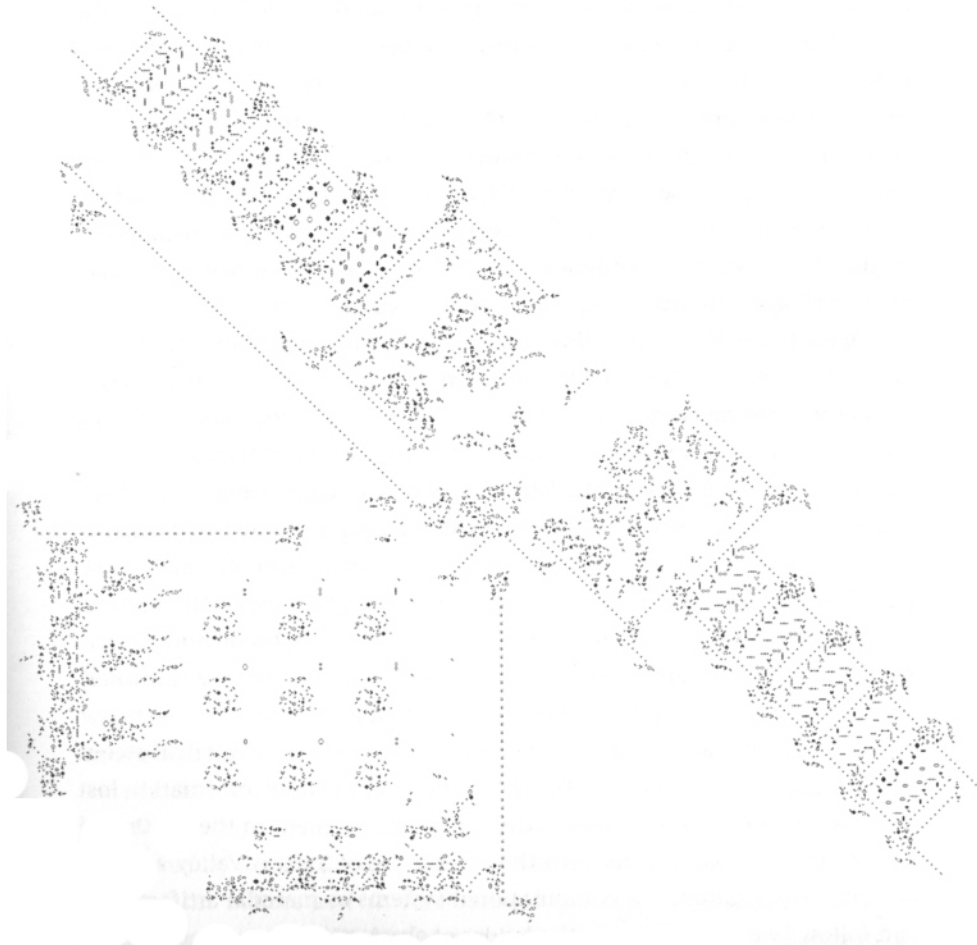

Figure 1.1
"This is a Turing Machine implemented in Conway's Game of Life." Designed by Paul Rendell.

example, seeks to model precisely this kind of spontaneous generation of information, or seemingly living or self-perpetuating patterns, from simple rule-sets. It, too, has been shown to be mathematically equivalent to a Turing machine, and indeed mathematician Paul Rendell designed a game of life that he proved to be Turing-equivalent (figure 1.1).[37]

In fact, if we accept the premise of organism as message, of informational patterns as a central organizing logic for biological life, we inevitably come to depend on computation as a frame for exploring that premise. Wiener's opening gambit of the turn from certainty to probability displaced but did not eliminate the old Enlightenment goals of universal, consilient knowledge. That ambition has now turned to building the best model, the finest simulation of reality's complex probabilistic processes. Berlinski observed the same trend in the distinction between analytic and computational calculus, noting how the discrete modeling of intractable differential equations allows us to better understand how complex systems operate, but always at the expense of gaining a temporally and numerically discrete, approximated view of things.[38] The embrace of cybernetic theory has increasingly meant an embrace of computational *simulations* of social, biological, and physical systems as central objects of study.

Hayles traces this plumb line in cybernetics closely in *How We Became Posthuman*, arguing that the Macy Conferences, where Wiener and his collaborators hammered out the vision for a cybernetic theory, also marked a concerted effort to erase the embodied nature of information through abstraction. In the transcripts, letters, and other archival materials stemming from these early conversations, she argues that the synthesizing ambitions of cybernetics led participants to shy away from considerations of reflexivity and the complications of embodiment, especially human embodiment, as they advanced their theory. But, as Hayles puts it, "In the face of such a powerful dream, it can be a shock to remember that for information to exist, it must *always* be instantiated in a medium."[39]

While Hayles's reading of cybernetics pursues the field's rhetorical ascent of the ladder of abstraction as she frames the story of "how information lost its body," there is a second side to the cybernetic moment in the 1940s and 1950s, one that fed directly into the emergence of Silicon Valley and the popular understanding of computational systems as material artifacts. We can follow Wiener back down the ladder of abstraction, too, through a second crucial cybernetic term, the notion of "feedback." The feedback loop,

as Hayles notes, is of interest to Wiener primarily as a universal intellectual model for understanding how communication and control can be generalized across different systems.[40] But the feedback loop was also a crucial moment of implementation for cybernetics, where the theoretical model was tested through empirical experiments and, perhaps more important, demonstrations.

Consider Wiener's "moth" or "bedbug," a single machine designed to demonstrate a feedback loop related to seeking or avoiding light. Wiener worked with electrical engineer Jerry Wiesner to create the machine, a simple mechanical apparatus with one photocell facing to the right and another to the left, with their inputs directing a "tiller" mechanism that would aim the cart's wheels as they moved. The demonstration achieved its intended purpose of showing lifelike behavior from a simple feedback mechanism, creating a seeming existence proof both of the similarity of mechanical and biophysical control mechanisms and of the efficacy of cybernetics as the model for explaining them. In fact, as historian Ronald Kline describes, the entire enterprise was a public relations stunt, the construction of the robot financed by *Life* magazine, which planned to run an article on cybernetics.[41] Wiener's demonstration machine presaged future spectacles of human–machine interaction like early Silicon Valley icon Douglas Engelbart's "mother of all demos," which first showcased several aspects of a functional personal computer experience in 1968.

The theoretical aspirations of cybernetics were always dependent on material implementation, a fact that has challenged generations of artificial intelligence researchers pursuing the platonic ideal of neural networks that effectively model the human mind.[42] Kline reports that *Life* never ran photos of Wiener's moth because an editor felt the machine "illustrated the *analogy* between humans and machines by modeling the nervous system, rather than showing the *human characteristics* of computers, which was *Life*'s objective."[43] In the end, Wiener had built a bug. The material context of the moth included not just a functioning feedback mechanism on wheels but the cultural aperture through which that construct would be viewed. In implementation, the mechanical feedback loop was overshadowed by an intellectual one, the relationship between a public scientist and his editors at *Life*. As it turned out they were less interested in Wiener's argument, that feedback mechanisms could be computationally

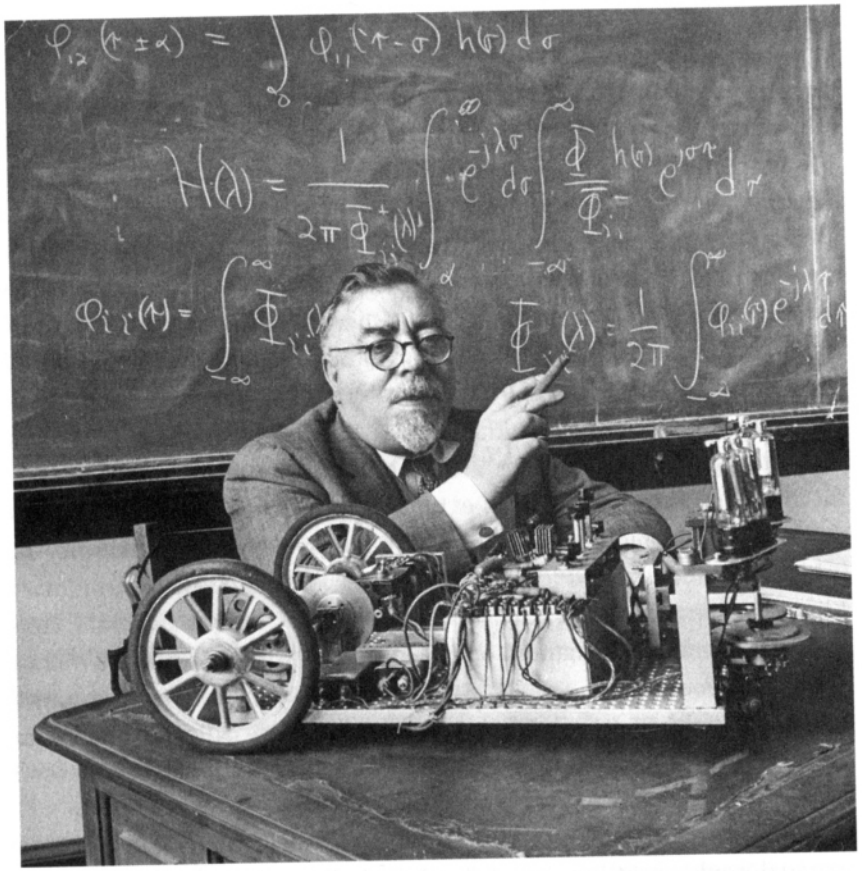

Figure 1.2
Norbert Wiener and his "moth" circa 1950. Alfred Eisenstaedt / The LIFE Picture Collection / Getty Images.

and mechanically modeled, than they were in searching out the human in the machine.

Thread 2: Metaphors for Magic

More than anything else, cybernetics was an attempt to create a new controlling metaphor for communication, one that integrated technological, biological, and social forms of knowledge. The story of Wiener's moth illustrates the hazards of this approach: defining a controlling metaphor for communication, and by extension for knowledge, requires a deep

examination of how language itself can shape both ideas and reality. The cybernetic vision of a unified biological and computational understanding of the world has never left us, continuing to reappear in the technical and critical metaphors we use to manipulate and understand computational systems. Chun explores the deeper implications of this persistent interlacing of computational and biological metaphors for code in *Programmed Visions*, demonstrating the interconnections of research into DNA and computer programming, and how those metaphors open up the interpretive problem of computation. For Chun the key term is "software," a word she uses to encompass many of the same concerns I explore here in the context of the algorithm.

Programmed Visions draws a direct link between the notion of fungible computability reified by the Turing machine and the kinds of linguistic magic that have come to define so many of our computational experiences:

> Software is unique in its status as metaphor for metaphor itself. As a universal imitator/machine, it encapsulates a logic of general substitutability; a logic of ordering and creative, animating disordering. Joseph Weizenbaum has argued that computers have become metaphors for "effective procedures," that is, for anything that can be solved in a prescribed number of steps, such as gene expression and clerical work.[44]

With the "logic of general substitutability," software has become a *thing*, Chun argues, embodying the central function of magic—the manipulation of symbols in ways that impact the world. This fundamental alchemy, the mysterious fungibility of sourcery, reinforces a reading of the Turing machine as an ur-algorithm that has been churning out effective computability abstractions in the minds of its "users" for eighty years. The "thing" that software has become is the cultural figure of the algorithm: instantiated metaphors for effective procedures. Software is like Bogost's cathedral of computation, Chun argues, "a powerful metaphor for everything we believe is invisible yet generates visible effects, from genetics to the invisible hand of the market, from ideology to culture."[45] Like the crucifix or a bell-tower signaling Sunday mass, software is ubiquitous and mysterious even when it is obvious, manifesting in familiar forms that are only symbolic representations of the real work it does behind the scenes.

The elegant formulation of software as a metaphor for metaphor, paired with Chun's quotation of Weizenbaum—the MIT computer