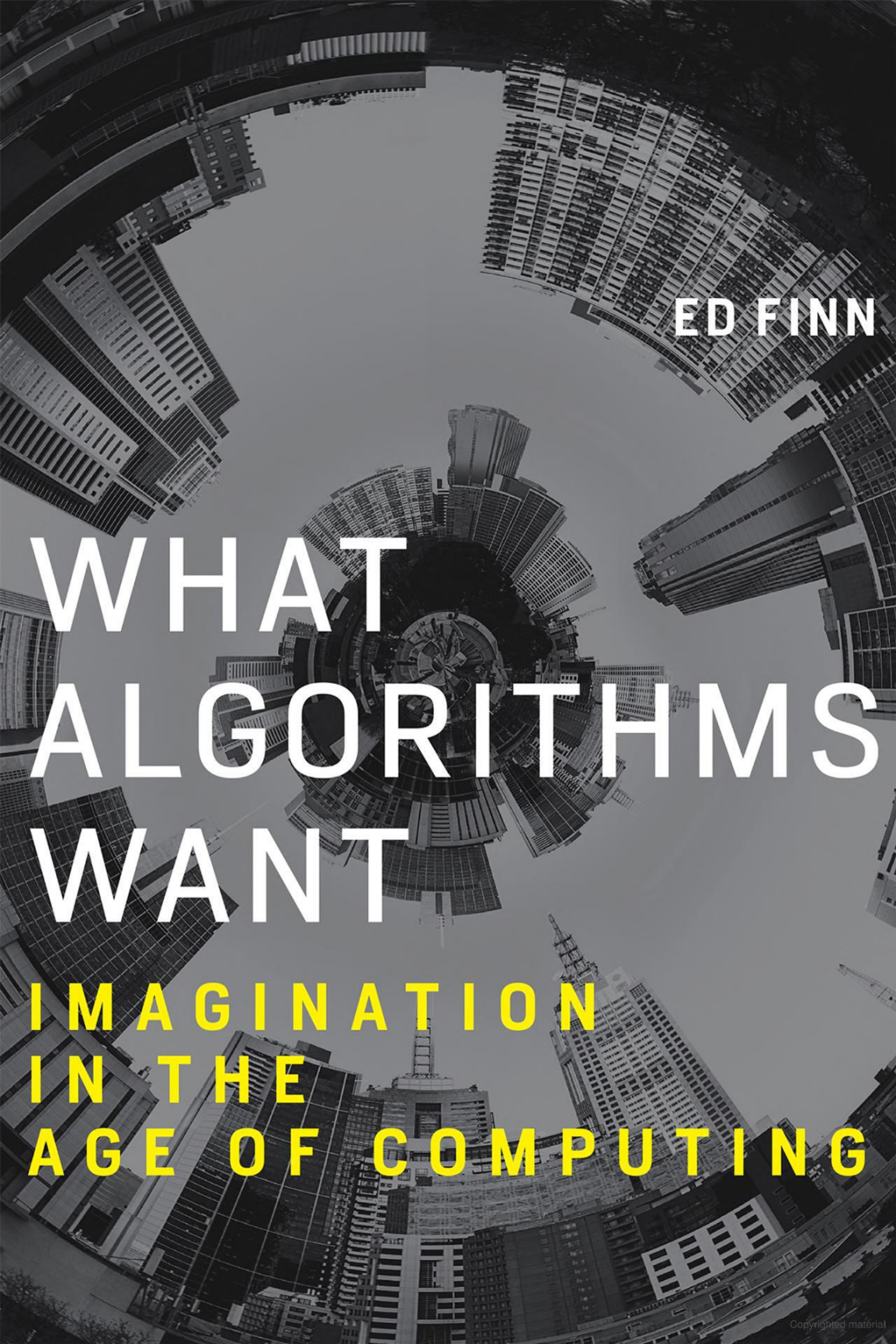

What Algorithms Want

Imagination in the Age of Computing

Ed Finn

The MIT Press
Cambridge, Massachusetts
London, England

This book was set in Stone Sans and Stone Serif by Toppan Best-set Premedia Limited. Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Names: Finn, Ed.
Title: What algorithms want : imagination in the age of computing / Ed Finn.
Description: Cambridge, MA : MIT Press, [2017] | Includes bibliographical references and index.
Identifiers: LCCN 2016030924 | ISBN 9780262035927 (hardcover : alk. paper)
Subjects: LCSH: Information technology--Social aspects. | Computers--Social aspects. | Algorithms--Social aspects.
Classification: LCC HM851 .F5565 2017 | DDC 303.48/34--dc23 LC record available at https://lccn.loc.gov/2016030924

10 9 8 7 6 5 4 3 2 1

Contents

Acknowledgments vii

Introduction 1
1 What Is an Algorithm? 15
2 Building the *Star Trek* Computer 57
3 *House of Cards*: The Aesthetics of Abstraction 87
4 Coding *Cow Clicker*: The Work of Algorithms 113
5 Counting Bitcoin 151
Coda: The Algorithmic Imagination 181

Notes 197
Works Cited 213
Figure Credits 233
Index 235

Acknowledgments

This book owes its existence to the generosity and support of many people and institutions. I count myself very lucky to have the support of my academic home, Arizona State University, in a tremendous range of large and small ways. Thanks go to President Michael Crow, for hiring me and making my unique position possible, and to the many university leaders who continue to support our strange experiment in imagination. I am especially thankful to the School of Arts, Media & Engineering, Director Sha Xin Wei, and Deans Steven Tepper and George Justice, for granting a vital research leave during the early composition phase of the book.

I am deeply grateful to my colleagues at the Center for Science and the Imagination who have supported my long and solitary sojourn as I carved out time to work on this book among many other pressing projects. Thanks to Ruth Wylie for taking on a huge burden of leadership during this period, and to Joey Eschrich, Michael Bennett, Brian David Johnson, Nina Miller, Bob Beard, Cody Staats, and Chelsea Courtney for making CSI such an exciting and rewarding place to work. A special thank you to Joey for heroic editorial efforts as I revised the manuscript, and to Joseph Bianchi for assisting me with image permissions.

A number of people gave me vital feedback on the work as it emerged: Lee Konstantinou, Corey Pressman, Jacqueline Wernimont, Sam Arbesman, G. Pascal Zachary, and George Justice. Nathaniel Greene and Connor Syrewicz were invaluable as research assistants for the project. I'm also grateful to the students of my Arts, Media & Engineering graduate seminar, Reading the Algorithm, for helping me clarify a number of ideas relating to the book. Perhaps the single greatest day for feedback came from a tremendous event titled "The Tyranny of Algorithms," organized by my wonderful colleagues at Future Tense, a partnership of ASU, New America, and *Slate*

magazine. Thanks to Torie Bosch and Will Oremus at Slate and Richard Gallant at CNN for feedback, conversation, and the chance to publish some of my thoughts along the way. And finally, I am very grateful to my collaborators at MIT Press: my editor, Doug Sery, for believing in this book; Michael Sims for his dedicated copyediting; and director Amy Brand for her support and enthusiasm of our multiple editorial projects.

All of these interventions, redirections, shows of support, and good advice vastly improved the book and I could not have finished it without them. All remaining imperfections are entirely my own.

Finally, I thank the Finns of Phoenix, my own intrepid band of adventurers, dancers, ad-hoc parade leaders, and dessert aficionados. Anna, Nora, Declan: I love you more than numbers can count or words can say.

Introduction

"Remember the first time you learned binary code?"
"Sure."
"You were forming pathways in your brain. Deep structures. Your nerves grow new connections as you use them—the axons split and push their way between the dividing glial cells—your bioware self-modifies—the software becomes part of the hardware. So now you're vulnerable—all hackers are vulnerable—to a *nam-shub*. We have to look out for one another."

Neal Stephenson, *Snow Crash*, p. 126

Codes and Magic

The myth is probably as old as language itself. There are spells in the world: incantations that can transform reality through the power of procedural utterances. The marriage vow, the courtroom sentence, the shaman's curse: these words are codes that change reality. It is an old and attractive idea.[1] From the *logos* of Genesis to the many religious traditions identifying the "true names" of God, humanity has persistently believed that certain invocations do not merely describe the world but make it. And why not? Language has always operated at the troubled boundary between reality and the description of reality. The more structured, abstract, and esoteric an idea, the less likely we are to divine its substance without first gleaning a name to call it by.

Today our languages sprawl across many registers: procedural computer languages, critical languages of film and new media, creoles, fictional languages, newspeak, emoji. In our perception, each of those registers ascribes certain magical powers to symbols and meaning; each of them generates cultural power based on the inherent tension between reality and

representation. The link between spoken language and abstract symbolic systems, particularly mathematics, has created new avenues for mystical connections between numbers, universal truths, and the fundamental structure of reality. Jewish kabbalah, Isaac Newton's fascination with alchemy, and biological examples of mathematical figures like the Golden Ratio all reinforce a particular metaphysical notion that some logical order, some grammar and symbolic vocabulary, underlies the universe.

In debating these questions, philosophers and mathematicians developed increasingly sophisticated understandings of symbolic languages, laying the groundwork for the contemporary era of computation. From its bones in set theory and symbolic logic to the latest articulations of data-driven machine learning, computation casts a cultural shadow that is informed by this long tradition of magical thinking. As computation transforms almost every aspect of cultural life, the stories we tell about it, the balance of myth and reason, will play a major role in determining what we can know and think. Language has power in the world, and may in some sense define the world. When enacted, symbolic logic can effect procedural alterations to reality.

The key term here is "enacted." This book uncovers how the humble vehicle of computation, the algorithm, has its roots not only in mathematical logic but in the philosophical traditions of cybernetics, consciousness, and the magic of symbolic language. To understand the algorithm we need to uncover those roots and then build a new model of "algorithmic reading" that incorporates a deep understanding of abstraction and process. The algorithm deploys concepts from the idealized space of computation in messy reality, implementing them in what I call "culture machines": complex assemblages of abstractions, processes, and people. Algorithms enact theoretical ideas in pragmatic instructions, always leaving a gap between the two in the details of implementation. The implementation gap is the most important thing we need to know, and the thing we most frequently misunderstand, about algorithmic systems. Understanding *how* we can know that requires the critical methods of the humanities. This is algorithmic reading: a way to contend with both the inherent complexity of computation and the ambiguity that ensues when that complexity intersects with human culture.

was the Sumerian language by releasing a nam-shub virus that, "coiled like a serpent around the human brainstem," scrambled humanity's ability to understand other Sumerian signals.[6] Linking this moment to the mythic Tower of Babel, *Snow Crash* makes the nam-shub a relic of a universal language once lost but now regained (and being put to nefarious use). Thus Stephenson taps into the much deeper mythos of language as incantation. If code can be magical and hackers are its shamans, we still recognize it as a symbolic system that operates at the intersection of cognition and reality. By investing the figure of code with cultural power, we also endorse the notion that it functions on a platform: the idea that humanity might run a universal operating system.

So code can be magical, code can change the world, and code can change the mind. But how does this actually work? What are the entities, the structures of operation in that space of computation? In *Snow Crash*'s neo-Sumerian operating system there are *me*, particular units of language that embody vital civilizational concepts. This trope is familiar from other traditions as well, where trickster figures like Prometheus or Coyote steal conceptual technologies (e.g., fire) from the gods. In one sense *me* are objects that can be transported and somehow deployed among populations. But they are also bodies of knowledge, sets of rules and procedures, that can be implemented in practice. They are technical entities that have their own existence independent of their human practitioners, but which operate through the medium of culture. They are algorithms.

This is a book about the algorithm as the vehicle or tool of computation: the object at the intersection of computational space, cultural systems, and human cognition. We need a deeper understanding of the algorithm in order to understand how computational systems are transforming our world today. In that sense this is a literacy exercise, an experiment in developing an "algorithmic reading" of the world. The role of the humanities and methodologies of critical reading—algorithmic reading—are vital to effectively contend with the ambiguity and complexity at play in the awkward intersection of computation and culture. But this is also a character study of an idea from its contemporary cultural presence to its philosophical foundations. *Snow Crash* neatly illustrates the tensions at play when algorithms stitch together computational, mythic, and cultural spaces. It's not so much a story about the power of code but its awkward embrace of the real, the ideal, and the imaginary in the guise of the algorithm.

This figure of the algorithm as a quasi-mystical structure of implemented knowledge is both pervasive and poorly understood. We have never been closer to making the metaphor of fully implemented computational knowledge real than we are today, when an explosion of platforms and systems is reinventing cultural practice and identity, often by implementing a *me* downloaded as an app or set up as an online service. We are surrounded by nam-shubs that we obey almost unquestioningly, from the dialog boxes and prompts we fill out on social media platforms to the arcane computation of credit scores. To begin excavating the algorithm, we need to understand the full scope of computational thinking and its interactions with the mythos of procedural language, starting with what we think algorithms *ought* to be.

The Cathedral of Computation

When technologists, researchers, and entrepreneurs speak about computational culture today, this deep myth of the algorithm is typically obscured by layers of rationalizing rhetoric and the procedural metaphors of software design. Indeed the most prevalent set of metaphors seems to be that of code as structure: platforms, architectures, objects, portals, gateways. This serves to both depersonify software, diluting the notion of software agency (buildings are passive; it's the architects, engineers, and users who act), and reifying code as an objective construct, like a building, that exists in the world.

Yet even within this architectural language, the mythological figure of the algorithm reasserts itself. Consider the popularity of the cathedral as a metaphor for code. George Dyson's wonderful history of the rise of computation is titled *Turing's Cathedral*. Another classic instantiation is Eric Raymond's book on open source software development, *The Cathedral and the Bazaar* (Raymond was arguing for the more transparent bazaar model, rather than the top-down approach of the cathedral). But perhaps the best analogy was offered at the IEEE Computer Society in 1988: "Software and cathedrals are much the same—first we build them, then we pray."[7] This was meant as a joke, of course, but it hides a deeper truth about our relationship to the figure of the algorithm today. The architecture of code relies on a structure of belief as well as a logical organization of bits.

The cathedral is not a perfect metaphor for computation, but its flaws signal precisely what we are missing. A cathedral is a physical and a spiritual structure, a house of God. In that sense the physical appearance of the building tells particular stories about faith and practice (e.g., a baptismal font, a nave pointing east, illustrations of biblical stories). But it also suggests a particular mode of access to the invisible space of religion, the house of God that exists beyond physical reality: transubstantiation, relics, and ceremonies are all part of the spectacle of the cathedral that reflect the invisible machinery of faith. Yet most of that machinery inevitably remains hidden: schisms, budgets, scandals, doctrinal inconsistencies, and other elements of what a software engineer might call the "back-end" of the cathedral are not part of the physical or spiritual facade presented to the world. Indeed, when the spectacle stutters for a moment and some uncomfortable fact lurches into view, the normal instinct is to ignore it, to shore up the facade of the cathedral in order to maintain one's faith. A cathedral is a space for collective belief, a structure that embodies a framework of understandings about the world, some visible and some not.

This is a useful metaphor for understanding the relationship we have with algorithms today. Writing in *The Atlantic* in early 2015, digital culture critic and game designer Ian Bogost called out our increasingly mythological relationship with software in an article titled "The Cathedral of Computation." Bogost argues that we have fallen into a "computational theocracy" that replaces God with the algorithm:

> Our supposedly algorithmic culture is not a material phenomenon so much as a devotional one, a supplication made to the computers people have allowed to replace gods in their minds, even as they simultaneously claim that science has made us impervious to religion.[8]

We have, he argues, adopted a faith-based relationship with the algorithmic culture machines that navigate us through city streets, recommend movies to us, and provide us with answers to search queries. We imagine these algorithms as elegant, simple, and efficient, but they are sprawling assemblages involving many forms of human labor, material resources, and ideological choices.

Bogost's central argument is this: while we imagine algorithms as a pinnacle of Enlightenment, rationalist thought, our engagements with them function in a very different mode. Through black boxes, cleanly designed dashboards, and obfuscating Application Program Interfaces, we are asked

to take this computation on faith. Just as the poorly paid factory workers who produce our high-tech gadgets are obscured behind the sleek design and marketing of brushed-metal objects that seem to manifest directly from some kind of machine utopia, untouched by human hands, so do we, the eager audience of that utopia, accept the results of software algorithms unquestioningly as the magical products of computation. The commodification of the Enlightenment comes at a price. It turns progress and computational efficiency into a performance, a spectacle that occludes the real decisions and trade-offs behind the mythos of omniscient code.

And we believe it because we have lived with this myth of the algorithm for a long time—much longer than computational pioneers Alan Turing or even Charles Babbage and their speculations about thinking machines. The cathedral is a pervasive metaphor here because it offers an ordering logic, a superstructure or ontology for how we organize meaning in our lives. Bogost is right to cite the Enlightenment in his piece, though I will argue the relationship between algorithmic culture and that tradition of rationalism is more complicated than a simple rejection or deification. The problem we are struggling with today is not that we have turned computation into a cathedral, but that computation has increasingly replaced a cathedral that was already here. This is the cathedral of the Enlightenment's ambitions for a universal system of knowledge. When we juxtapose the two we invest our faith into a series of implemented systems that promise to do the work of rationalism on our behalf, from the automated factory to automated science.

I address this relationship more closely in chapter 2, but for now we need only to appreciate the implications of the cathedral of computation as shorthand for a unified system of understanding. The bas-relief work, statues, and inscriptions of great European cathedrals are microcosms of Christianity, recapitulating the Gospel and other key biblical narratives as well as the histories of their own creation as enduring and complete statements of faith. Contemporary computational systems perform the same role of presenting a unified vision of the world through clean interfaces and carefully curated data—everything you might want to know, now available as an app. Computation offers a pathway for consilience, or the unification of all fields of knowledge into a single tree: an ontology of information founded on the idea that computation is a universal solvent that can untangle any complex system, from human consciousness to the universe itself.

One of the few long-form investigations of the algorithm as a concept, mathematical historian David Berlinski's *Advent of the Algorithm*, even concludes with an argument connecting the notion of universal computation to intelligent design. He argues that the algorithm, a lens for the notion of "effective calculation," has done nothing less than to have "made possible the modern world."[9] Berlinski sees "the appearance of *intelligence* on alien shores"—that is, in the spaces of computation—as further evidence that some explanation for the nature of the universe must exist beyond the system itself.[10] His work turns on the distinction between information and meaning, between the work a Turing Machine does in processing symbols on a tape and the impact of those symbols on the human mind. We hear echoes of *Snow Crash* in the suggestion that Turing and fellow mathematician Emil Post's visions of universal calculating machines are

> responsive to a world of *thought*, and not matter at all. … The essence of their machines is elsewhere, in a universe in which symbols are driven by symbols according to rules that are themselves expressed in symbols.
>
> The place in which these machines reside is the human mind.[11]

This is precisely the apotheosis that Bogost calls out in his essay, suggesting that we have veiled the material realities of algorithms behind a mystical notion of computation as a universal truth. We see this faith in computation invoked repeatedly at the intersection of algorithms and culture. Facebook's mission statement is "to give people the power to share and make the world more open and connected," a position that embeds assumptions like the argument that its social graph algorithms will grant us power; that its closed, proprietary platform will lead to more transparency; and that transparency leads to freedom, and perhaps to empathy. Uber is "evolving the way the world moves. By seamlessly connecting riders to drivers through our apps, we make cities more accessible, opening up more possibilities for riders and more business for drivers." The theocracy of computation will not merely change the world but evolve it, and it will open new possibilities for users, linking proprietary commerce and individual freedom. These changes will be effected not only in the material realm but in the cultural, mental, and even spiritual spaces of empowerment and agency. The algorithm offers us salvation, but only after we accept its terms of service.

The important lesson here is not merely that the venture capitalism of Silicon Valley is the ideology bankrolling much of our contemporary

cultural arbitrage by manipulating certain kinds of computational abstraction to achieve cultural and financial success.

As algorithms become more adept at reading cultural data and performing real-time arbitrage (used here in the sense of financial pricing arbitrage but also cultural arbitrage as described in the previous chapter), they are taking on new forms of intellectual labor. They are authoring and creating, but they are also simplifying and abstracting, creating an interface layer between consumers and the messy process of, say, getting a cab or hiring a housekeeper. Chapter 4 begins with Ian Bogost's satirical Facebook game *Cow Clicker* and its send-up of the "gamification" movement to add quantification and algorithmic thinking to many facets of everyday life. Such games trouble the boundaries between work and play, as do much more serious forms of gamification like Uber and the high-tech warehouse workers whose every second and step are measured for efficiency. Taken together, these new models of work herald a novel form of alienated labor for the algorithmic age. In our science fiction present, humans are processors handling simple tasks assigned by an algorithmic apparatus. Drawing on the historical figure of the automaton, a remarkable collection of Mechanical Turk-powered poetry titled *Of the Subcontract*, and Adam Smith's conception of empathy in his *Theory of Moral Sentiments*, I explore the consequences of computational capitalism on politics, empathy, and social value.

The root of the algorithmic sea change is the reimagination of value in computational terms. Chapter 5 leads with the flash crash in 2010 and the growing dominance of algorithmic trading in international markets (described by journalist Michael Lewis's *Flash Boys*, among others) to frame a reading of Bitcoin and related cryptocurrencies. By defining the unit of exchange through computational cycles, Bitcoin fundamentally shifts the faith-based community of currency from a materialist to an algorithmic value system. Algorithmic arbitrage is forcing similar transitions in the attribution of value and meaning in many spaces of cultural exchange, from Facebook to journalism. The fundamental shift from valuing the cultural object itself to valuing the networks of relations that the object establishes or supports leads to new practices and aesthetics of production, where form and genre give way to memes and nebulous collaborative works. Using Bitcoin as an example of this new value model, I close by considering the consequences of programmable value for the notion of a

public sphere in the twenty-first century, an era when arbitrage trumps content.

In the coda I briefly retrace this genealogy of the algorithm to consider our future prospects for achieving the twinned desires embedded in the heart of effective computability: the quest for universal knowledge and perfect self-knowledge. These ambitions are particularly vital for the humanities, and we cannot stop at algorithmic reading. To truly grapple with the age of the algorithm and our growing entanglement with computational cultural processes, we need to take action as scholars, teachers, and most of all performers of humanistic inquiry. We need an experimental humanities, a set of strategies for direct engagement with algorithmic production and scholarship, drawing on theories of improvisation and experimental investigation to argue that a culture of process, of algorithmic production, requires a processual criticism that is both reflexive and playful. This is how we can begin to understand the figure of the algorithm as a redrawing of the space for cultural imagination and become true collaborators with culture machines rather than their worshippers or, worse, their pets.

1 What Is an Algorithm?

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.

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,

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

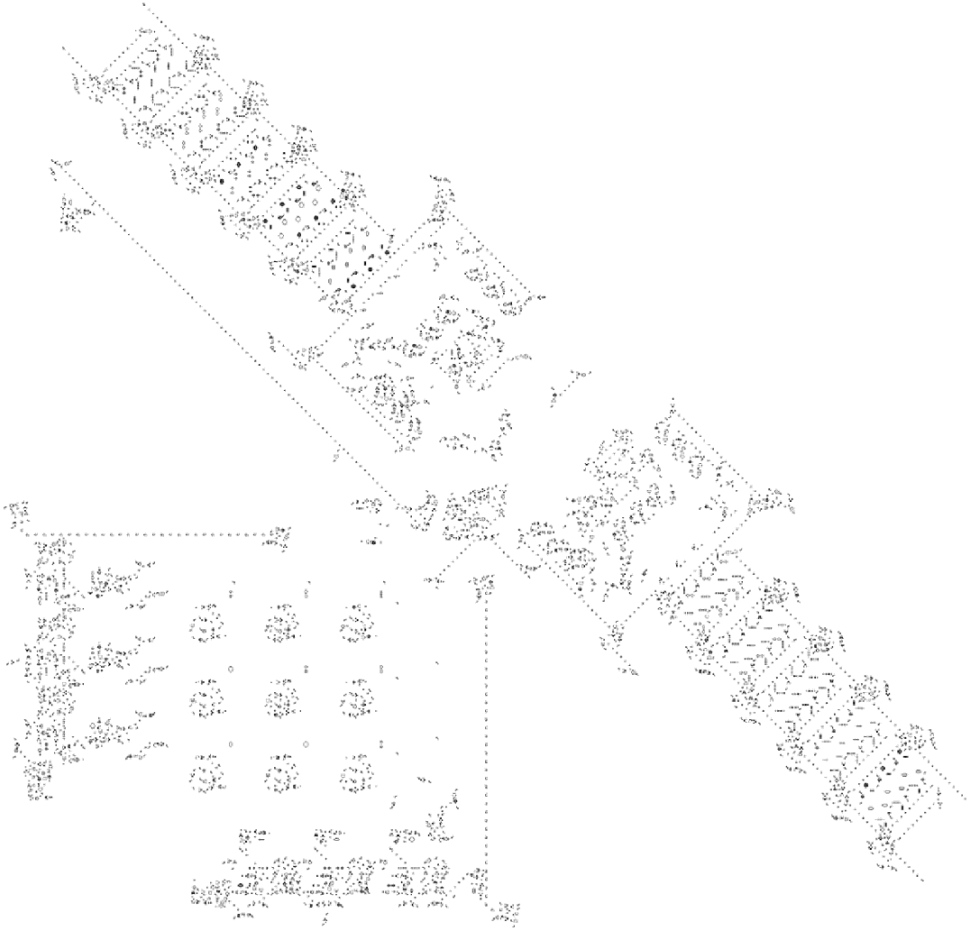

Figure 1.1
"This is a Turing Machine implemented in Conway's Game of Life." Designed by Paul Rendell.

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,

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

scientist who created an alarmingly successful algorithmic psychotherapist called ELIZA in the 1960s—draws together cybernetics and magic through the notion that computers themselves have become metaphors for the space of effective computability. The algorithm is not a space where the material and symbolic orders are contested, but rather a magical or alchemical realm where they operate in productive indeterminacy. Algorithms span the gap between code and implementation, between software and experience.

In this light, computation is a universal solvent precisely because it is both metaphor and machine. Like Wiener's robotic moth, the implemented algorithm is on the one hand an intellectual gesture ("Hello, world!"), a publicity stunt, and on the other a functioning system that embeds material assumptions about perception, decision-making, and communication in its construction. For example, think of the humble progress bar. When a new piece of software presents an indicator allegedly graphing the pace of installation, that code might well be a bit of magic (the status of the bar holding little relation to the actual work going on behind the scenes). But that familiar inching bar is also a functional reality for the user because no matter how fictitious the "progress" being mapped, nothing else is going to happen until the bar hits 100 percent—the illusion dictates reality. The algorithm of the progress bar depends not only on the code generating it but the cultural calculus of waiting itself, on a user seeking feedback from the system, and on the opportunity—increasingly capitalized on—to show that user other messages, entertainments, or advertising during the waiting phase.

As our generally unthinking acceptance of the progress bar demonstrates, we are primed to accept these magical calculations on multiple levels. We believe in the power of code as a set of magical symbols linking the invisible and visible, echoing our long cultural tradition of *logos*, or language as an underlying system of order and reason, and its power as a kind of sourcery. We believe in the elegant abstractions of cybernetics and, ultimately, the computational universe—that algorithms embody and reproduce the mathematical substrate of reality in culturally readable ways. This is what it means to say that an algorithm is a *culture machine*: it operates both within and beyond the reflexive barrier of effective computability, producing culture at a macro-social level at the same time as it produces cultural objects, processes, and experiences.