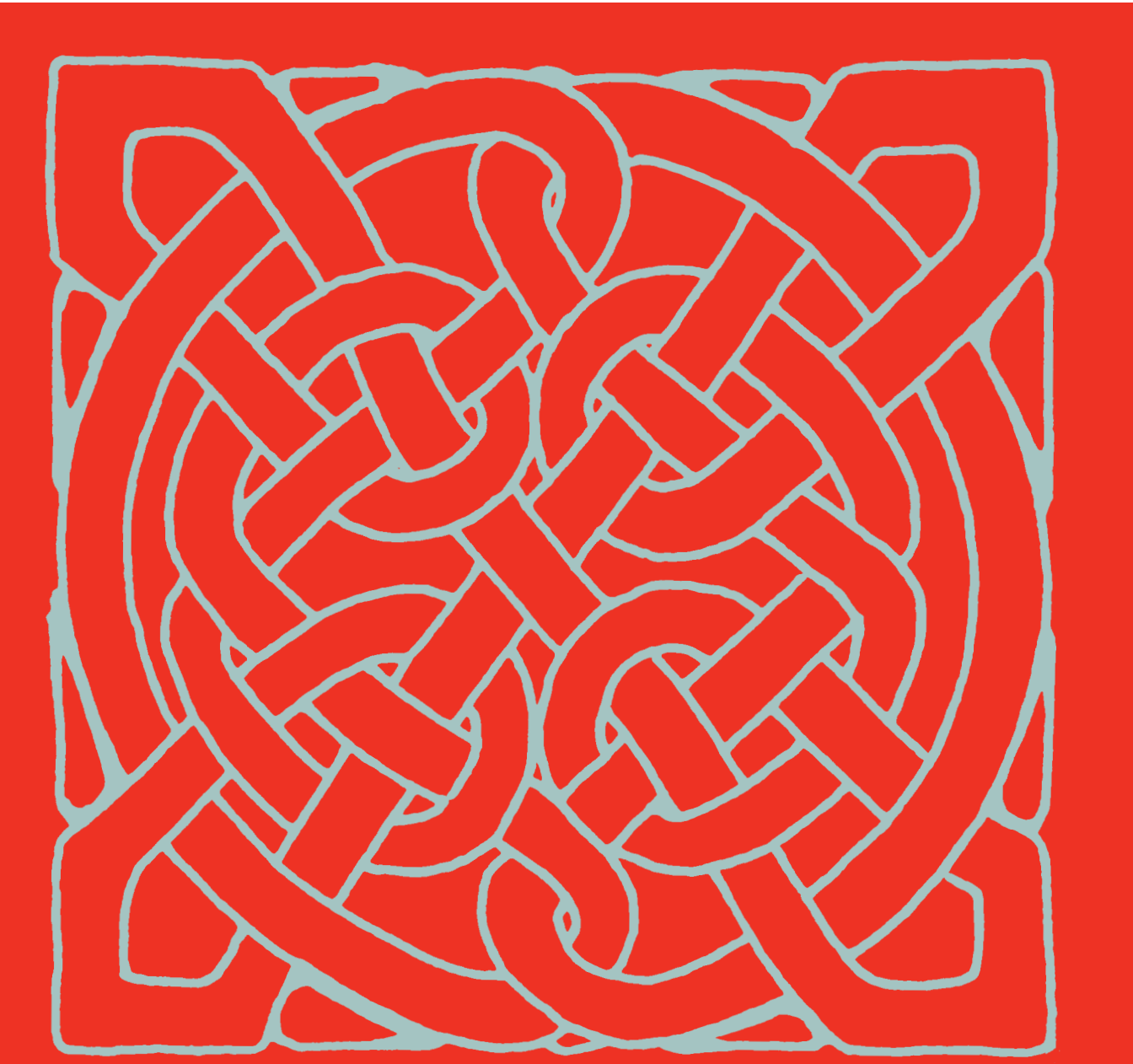
ERIC B. BAUM

WHAT IS THOUGHT?



What Is Thought?

What Is Thought?

Eric B. Baum

A Bradford Book
The MIT Press
Cambridge, Massachusetts
London, England

This book was set in Times New Roman on 3B2 by Asco Typesetters, Hong Kong, and was printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Baum, Eric B., 1957–
What is thought? / Eric B. Baum.
p. cm.
"A Bradford book."
Includes bibliographical references (p.) and index.
ISBN 0-262-02548-5 (hc. : alk. paper)
1. Philosophy of mind. 2. Cognitive science. 3. Thought and thinking. 4. Semantics (Philosophy) I. Title.
BD418.3.B38 2004
128′.2—dc22 2003059544

To my parents, Leonard and Julia Baum

Contents

Acknowledgments

I want to thank and acknowledge all the scientists and scholars whose ideas and teachings have influenced my thoughts, but particularly to thank by name the many individuals who have read and commented on portions of the text, including Elise Baum, Stefi Baum, Gary Flake, Dan Gindikin, David Heckerman, Elliot Justin, Charles Markley, Lee Neuwirth, Chris O'Dea, Barak Pearlmutter, Herman Tull, several anonymous referees, and especially Peter Neuwirth. This book was greatly improved by their comments, but of course any errors that remain are my own. I also thank Small World Coffee for stimulation and a fine working environment.

What Is Thought?

1 Introduction

Over half a century ago, Erwin Schrödinger, the co-inventor of quantum mechanics, wrote a short book called *What Is Life?* (1944). He began as follows:

> How can the events *in space and time* which take place within the spatial boundaries of a living organism be accounted for by physics and chemistry?
>
> The preliminary answer which this little book will endeavor to expound and establish can be summarized as follows:
>
> The obvious inability of present-day physics and chemistry to account for such events is no reason at all for doubting that they can be accounted for by those sciences. (3; italics in original)

Schrödinger was writing ten years before the discovery of the structure of DNA by James Watson and Francis Crick, but the main thesis of his book was that genetic information was carried by a molecule, which he incorrectly thought was a protein. The reason why the physics and chemistry of Schrödinger's day could not understand life, he remarked, was that ordinary physics and chemistry arise from statistics, involving the interaction of vast numbers of atoms. Statistics assumes that the system will be found in a random configuration and thus will have properties characteristic of likely configurations. But life is the result of the evolution of genetic information, which has selected for very complex processes that by ordinary considerations would be enormously unlikely. Thus, most of the ideas, intuitions, and methods of classical physics are inappropriate for understanding life.

It was unusual for Schrödinger, a physicist not a biologist, to be writing on life. However, he believed that, short of an appeal to mysticism, life must be explainable at a fundamental level by physics and chemistry. Yet life seemed to violate the normal behavior of entropy, which is central to physics. The chemicals in the body continue a long cascade of intricately organized reactions for 70-plus years, contrary to usual statistics that expect organization to be dissipated, as a pot scatters into shards when it falls. The fact that life had ultimately to be explainable by physics, yet seemed inconsistent with physics, seemed like a fruitful avenue to explore.

Today I believe we are at a stage where it is productive to write a book called *What Is Thought?* asking how the computational events that take place within the spatial boundaries of your brain can be accounted for by computer science. I argue that the obvious inability of present-day computer science to account for such events is no reason at all for doubting that they can be accounted for by computer science. The situation we have is in fact parallel to that facing Schrödinger: the mind is complex because it is the outcome of evolution. Evolution has built thought processes that act unlike the standard algorithms understood by computer scientists. To understand the mind we need to understand these thought processes, and the evolutionary process that produced them, at a computational level.

My goal is to lay out a plausible picture of mind consistent with all we know, and in fact to lay out what I argue is the most straightforward, simplest picture of mind. I accept no mysticism; assume that we are just the result of mechanical processes explainable by physics; accept that we were created by evolution; accept some unproven hypotheses for which there is near-consensus among the computer science community on the basis of strong evidence (such as "the Church-Turing thesis" and "P $\neq$ NP," both of which I explain); and bring to bear whatever seem like hard results from a variety of fields, including molecular biology, linguistics, ethology, evolutionary psychology, neuroscience, and computational experimentation. As much as seems warranted, the discussion in this book follows what I perceive to be folk wisdom among computer scientists interested in cognition, but the attempt to pull ideas together and see whether a fully coherent picture emerges will lead us in directions that have been underexplored. I am confident that the picture herein will not convince all readers, because at the present level of knowledge it is impossible to offer a proof that the mind in fact works in such and such a way, especially on subjects like "the nature of experience," but I hope to offer a principled proposal to meet all concerns.

A nice feature of Schrödinger's book is that it was written at a level accessible and interesting to both scientists and scientifically literate laypeople. This was possible because Schrödinger was a great writer but also because the understanding of life in the literature was hazy at the big picture level and missing or wrong in all the details, so that what was important was to explain the essence of some big ideas. In the present task also, I think we are sufficiently far from understanding mind that the details of published work on the computational approach to intelligence would probably be irrelevant. My hope is to extract key ideas from the computational and other literatures, to fit them into a big picture, and to explain everything essential about that picture—indeed everything I think I know about the mind—to a mixed audience. Of course, once a suitable picture exists, it would be important to fill in the details with as much mathematical rigor as possible, but I am treating that largely as a matter for future work. I have sketched for the general reader the intuitions behind mathematical arguments where they are crucial. At some places I could have filled in more details but chose not to, so as not to lose the train of thought. But there are many places where the mathematics remains to be worked out.

1.1 Meaning, Understanding, and Thought

There is an underlying theme to almost everything this book says, which can be expressed in a single summary sentence. Here it is.

Semantics is equivalent to capturing and exploiting the compact structure of the world, and thought is all about semantics.

Let's focus on the first half of this summary sentence first. The book explains in some detail why computer scientists are confident that thought, and for that matter life, arises from the execution of a computer program. The execution of a computer program is always equivalent to pure syntax—the juggling of 1s and 0s according to simple rules. The key question, which has been posed primarily by philosophers, is how syntax comes to correspond to semantics, or real meaning in the world.

The answer this book suggests is that semantics arises from the principle, roughly speaking, that a sufficiently compact program explaining and exploiting a complex world essentially captures reality. The point is that the only way one can find an extremely short computer program that makes a huge number of decisions correctly in a vast and complex world is if the world actually has a compact underlying structure and the program essentially captures that structure.

Physics is a good analogy here. Physicists have written down a short list of laws that allow them to predict the outcomes of many experiments. Thus, they believe that the world really does have an underlying simplicity described by these simple laws. I argue here that mind is a complex but still compact program that captures and exploits the underlying compact structure of the world.

I refer to this principle as *Occam's razor*. Simpler versions of Occam's razor date back to at least William of Occam's fourteenth-century dictum *Pluralitas non est ponenda sine necessitate*, "Entities should not be multiplied unnecessarily." Occam's razor has been formalized over the last few decades by computer scientists to describe the training of compact programs that predict many observations, and it is this work that first spurred my investigations. But the reason for using the term *exploiting* in the summary sentence is that this book discusses Occam's razor in a somewhat more general context. Finding a compact underlying structure and finding algorithms to exploit it are two separate hard computational problems. The mind exploits its understanding of the world in order to reason. The programs trained by computer scientists typically output a single classification, positive or negative, to a single instance of some problem. But mind typically produces a computer program capable of behaving, that is, of doing elaborate computation leading to an appropriate series of actions, and in fact of behaving in the face of a whole class of problems. I argue that this additional power is a result of the evolutionary programming that led to mind and implies, or rather is equivalent to, understanding the semantics.

If we look for a compact description underlying mind, one stands out like Venus on a moonless night. With its myriad of neurons and connections the brain is huge, but its DNA program is much smaller—at first glance quite surprisingly small when

one analyzes its function. So I argue that, counter to some of the folk wisdom in the computational and cognitive science communities, mind is essentially inherent in the DNA in some detail. There is no doubt that learning during life is important, but because the DNA is the compact program that is the core, learning during life is essentially guided and programmed by the DNA—a phenomenon called inductive bias. We learn extremely rapidly, much more rapidly than computer scientists have been able to explain, because our learning is entirely based on and guided by semantics. The reason we learn so fast, the reason our learning is guided by semantics, is that the compact DNA code has already extracted the semantics and constrains our reasoning and learning to deal only with meaningful quantities.

The second half of the summary sentence is that mind is all about semantics. This is true on many levels. What distinguishes mind from artificial intelligence programs is that mind understands—it exploits semantics, or meaning, for computation. The way the mind reasons about things, which is different from the way human-written computer programs do, is by manipulating semantic chunks and exploiting the compact structure.

How does mind solve problems as fast as it does, and how did evolution solve the problem of producing us as fast as it did? Evolution had 4 billion years and vast resources, but computer science tells us to expect that these problems are so hard to solve (because there are so many possible answers that must be searched through) that even that amount of time and those resources should not have been enough. I discuss several answers, but the main one is that mind is so fast because it exploits semantics (as evolution did). Evolution discovered semantically meaningful chunks such as the subroutine "build an eye here" and then was able to reason how to construct new creatures by manipulating these semantically meaningful chunks. Mind understands the world in terms of meaningful concepts and is able to reason so fast because it only searches through meaningful possibilities.

The flip side of dealing with an apparently very complex world that however has a very compact underlying structure is that the complexities are often highly constrained, indeed overconstrained. Once the semantics are understood, one can often reason straightforwardly by exploiting the constraints. While there are myriad possibilities, only one or a few make sense.

When people write computer programs, they organize them into small, mostly self-contained units called subroutines or modules, each addressing some particular subcomputation. Evolving a very compact code that deals with the world leads to a code that is highly modular. By producing a program with many subroutines corresponding to meaningful concepts, evolution produced a program that is compact because it reuses these subroutines in multiple different contexts. This is why, I believe, thought

is so often based on analogy and metaphor—mind invokes one of these subroutines to understand a new context.

The brain does vast computations of which we are unaware in order to compute semantically meaningful quantities. What reaches our awareness is only the outcome of these processes—meaningful quantities. Mind is an evolved program that exploits the compact structure of the world to make decisions advancing the interest of the genes. Once one looks at it in these terms, it is straightforward to explain the qualitative nature of experience, the meaning of self, the nature of awareness, free will, and all that. Once one makes the ansatz[1] that every thought is simply the execution of computer code, and understands how that code is evolved to deal with semantics, a self-consistent, compact, and meaningful picture of consciousness and soul will follow as naturally as thoughts follow from the constraints of meaning.

In short, we're going to go back and forth between two closely related concepts and one process: compactness, meaning, and evolutionary programming. This book proposes that meaning arises from evolving a very compact program and that understanding is equivalent to exploiting the compact structure of the world. Thought and learning as well as the evolution of this program are as fast as they are because they exploit meaning and understanding. Awareness is awareness of meaningful quantities. The structure and nature of thought as well as consciousness naturally arise from the dynamics of evolution of programs that exploit the compact structure of the world.

1.2 A Road Map

The previous section painted some conclusions with a very broad brush. The rest of this chapter surveys the paths that led me to these conclusions, which I detail in subsequent chapters to justify them and make their meaning more explicit and concrete.

Chapter 2 begins by describing what an algorithm is, what a program is, and hence why computer scientists are so confident that the mind is equivalent to a computer program. To this end, it reviews Alan Turing's 1936 construction of the Turing machine, the intellectual model on which the computer is based. Turing addressed the question, What is an algorithm? At the time, before the invention of electronic computers, this question was much less settled than it is now. His approach was to analyze the process of thought, breaking it down into simple steps in a very general way. He asked, What is the most general thing that the mind could possibly be doing, and how can I analyze that into small, simple pieces? Since he could capture the most general possible thing in a computer program, he was able to show that

whatever more specific thing was actually going on could be captured in a computer program as well. His analysis provides the foundation for modern computer science theory and simultaneously defines what is sometimes known as *strong Artificial Intelligence*—the thesis that the mind is equivalent to a computer program—because his mathematical definition of a computer program was modeled directly on his analysis of thought. The present book is largely an attempt to spell out the details and ramifications of strong Artificial Intelligence (AI).

The modern picture we come to, following Turing, is that an *algorithm*, also sometimes known as an *effective procedure*, is simply a recipe saying exactly what happens next at each step. As long as a system is following a precise recipe (even if the recipe allows for random elements), it is following an algorithm. Thus, a system running under well-defined physical laws can be considered to be running an algorithm. Evolution can thus be considered an algorithm. And the mind—the working of the brain, which is a system running under physical laws—can be considered a computer program.

Turing's analysis tells us that the mind can be considered to be a computer program, but it doesn't tell us very much about the nature of the program. His construction captures the most general thing that the mind could possibly be doing, restricted only by the assumption that the operation of the brain is subject to simple physical laws. Indeed, his analysis is not specific to brains or minds at all but applies equally well to more general systems. Thus, it is clear that considerable insight must be added to understand whatever is special about the mind.

To demonstrate that the Turing machine view can be fruitful for forming a more detailed picture, chapter 2 describes another gem from the early history of computer science: John von Neumann's 1948 construction of self-reproducing automata. Von Neumann asked, How can a machine sitting in a vat of parts construct a copy of itself? Think of the parts as organic molecules floating in a soup. To understand life, one has to answer this question. Constructing such a copy is a massive computational problem: one must provide an algorithm by which the machine figures out which parts it needs when and assembles them correctly. Writing five years before Watson and Crick unraveled the structure of DNA, von Neumann identified critical computational problems in achieving this and proposed an ingenious solution—seemingly the only possible solution. His solution was to invent DNA from first principles, that is, he was forced by computational considerations to posit a structure serving exactly the same function as we now know DNA does in living things. The picture of the structure he drew is instantly recognizable today as DNA. And von Neumann constructed it, essentially, as a program for a Turing machine. This serves as a dramatic reminder that life and the biological evolution that ultimately pro-

duced mind are simply particular types of Turing machine programs. The final section of chapter 2 surveys some details of how life is the execution of the DNA program and begins to suggest fundamental similarities between the program of life and the program of mind.

Turing's analysis of the nature of algorithms reduces everything to simple syntax. The execution of a computer program, in his picture, is simply a series of steps each of which manipulates formal symbols within the computer according to simple mechanical rules. So, one step might be "take the contents of register x, add them to the contents of register y, and place in register z." Thus, it can be argued that the mind must be reducible to simple syntactic manipulations. But, as has been pointed out by numerous philosophers challenging the strong AI position, this raises many questions. One set of challenges, raised by philosophers such as David Chalmers and Frank Jackson, comprises intuitionist challenges based on our experience of being alive. We feel, we are conscious, we experience. How can we possibly be simply a machine? Where in a machine is pain and the sensation of smelling a rose?

Chapter 3 discusses a second set of challenges, raised by philosophers such as Hubert Dreyfus, John Searle, and Roger Penrose (building on several thousand years of other philosophic investigation), How can symbols in a computer come to mean anything in the world? In what sense can the contents of register x correspond to, say, a snowball that is flying toward my face? If thoughts in the mind simply correspond to syntax in a computer program, how and in what sense do they come to correspond to objects in the world?

This second set of questions is further motivated by sad experience with actual computer programs. AI critics have given example after example where people exercise understanding and computer programs are completely clueless. Computer programs are typically narrow and very brittle at what they do. So, a program might be written to answer questions about stories about people in restaurants, but ask it a nonsense question like, Did the customer eat his food with his mouth or with his foot? and it is immediately lost. People display understanding; ask them anything, and they come up with some kind of reasonable answer. Change the scope of the problem, and they do something intelligent. These are abilities that we currently have no hope of getting a computer to display. Why is this? How can a machine understand? What is understanding anyway? Does some quantity called understanding distinguish mechanical computation from thought? These are pivotal questions that this book must address.

My position is that these philosophic challenges are of great merit and indeed cut to the quick of why mind seems inconsistent with our current computer science techniques. But I further assert that results in computational learning theory over the

last 20 years or so point to the answers to these challenges and thus elucidate the nature of the program of mind and of how it came to exist.

The picture presented here is that mind relies fundamentally on Occam's razor. Occam's razor is the well-known and intuitive prescription that, given any set of facts, the simplest explanation is the best. Occam's razor underlies all of science. It is, for example, the way in which physicists come to their small collection of simple laws that fundamentally explain all physical phenomena, how chemists arrive at the periodic table, why biologists believe in heredity. Newton's laws, for example, are simple in the sense that they can be written down on a single page, yet they explain a vast number of physical experiments and phenomena.

Occam's razor further underlies all of human reasoning. It is why, for example, jurors do not reach for some Rube Goldberg hypothesis that is consistent with any evidence that could possibly be presented and also exonerates the accused. An explanation that is too complex is judged to be "beyond reasonable doubt."

This book claims that Occam's razor (as generalized and extended) is the basis of mind itself.

The examples given of Occam's razor in scientific and ordinary reasoning are intuitively appealing, but examined further the intuition does not make clear exactly in what sense an explanation is simple nor why Occam's razor should hold. Computer scientists have formalized Occam's razor in several related ways, three of which are discussed in chapter 4. Roughly speaking, for computer scientists an explanation is a computer program, and the simplest explanation is just the shortest computer program.

The simplest of the three formal versions of Occam's razor is just a sophisticated version of curve fitting. If you have a large collection of appropriately gathered data points and you succeed in finding a straight line that fits them well, you can be pretty confident that the line has really captured some truth about the world. Its slope is not just a symbol in the curve fitter; rather, it corresponds to reality. If you go out and gather more data points, you expect that they also will lie on the line—the line makes predictions that generally come true in the world. This is why statisticians, social scientists, salespeople, and politicians all like to hold up charts showing straight lines fitting data.

Roughly speaking, the reason a line that agrees with data is believed to have predictive power is that there are very few ways to draw lines. Each data point that the line is required to agree with is another constraint on the line. If the line agrees with a lot of points, that's unlikely to be an accident.

Computer scientists sometimes study a model of brain circuits called neural networks. These contain a collection of objects that represent neurons (the objects are really just simple mathematical operators). The "neurons" are wired together, with

the output of some neurons feeding into the input of others. So all a neuron is (in this model) is something that looks at numerical inputs and produces a numerical output according to a simple mathematical rule. Associated with the connections between neurons are weights that determine how strongly the neurons are connected. Some numbers are fed into the inputs of the whole network. Some neurons compute their outputs and pass them to other neurons, which then compute outputs, until finally the whole net produces its output.

Such a neural network is trained by adjusting the connection weights so that the net learns to do the right thing on many training examples. You might, for example, show it pictures of faces, some smiling and some frowning. The neurons would input a numerical representation of the pictures and produce output numbers. You would adjust the weights so that whenever you show the network a smiling face from a set of training pictures, it outputs a 1, which we take to indicate "smiling," and similarly when you show it a frowning face from the set, it outputs a 0, which we take to indicate "frowning." Now, the interesting thing is that once a network has been trained on a sufficiently large collection of examples—many more than there are weights in the network[2]—it learns to generalize and will correctly distinguish smiling from frowning faces in most pictures it has never seen before. To some limited extent, it "understands" enough to distinguish smiling and frowning.

This is essentially just a more complex example of curve fitting. Neural networks are just a complex class of curve, that is, they are mathematical functions parameterized by weights, and once one has constrained them enough with examples, they are forced, so to speak, to represent the underlying structure in the process classifying the data. That is why they then get the right answers on new examples they have not seen before.

More generally, there are only so many small computer programs of a given type. Again, each data point with which you require a small computer program to agree is another constraint on that program. If the number of constraints vastly outweighs the flexibility in writing the program, you would naively not expect to be able to find a program agreeing with the data. When you do, it is in a sense because the syntax of the program fundamentally reflects the process producing the data. Finding such a compact program thus demonstrates that the data are actually produced by some simple process and also that the program you found reflects the simple process in some way. So, this is my first answer to how and why the syntax of the computer program of mind corresponds to reality in the world: it is based on a program so compact it has no choice but to do so.

The claim is that if one somehow finds a sufficiently compact program agreeing with sufficiently many data points, the mere fact of the existence of the program

more or less guarantees that the data are in fact generated by a simple process and, further, that the syntax of the program reflects that process. Another interesting question is how such a compact program agreeing with the data could be found. Computer scientists train neural networks by a slow process closely related to evolution in that they make a long series of small changes, each of which improves the agreement of the net with the data a little bit. In this way, the whole net slowly settles into a configuration where its syntax reflects the process creating the data. Each weight becomes tuned so that it cooperates with the others in a representation of the world. Such a training process is computationally intensive. The mind was created by an even more computationally intensive evolutionary process over 4 billion years, which yielded a vastly more impressive understanding of the world than any artificial neural network can. The characteristics and effectiveness of such hillclimbing procedures are discussed in chapter 5.

From this point of view one can easily understand why the AI programs critiqued by Dreyfus and others were so clueless and why computer programs still show no sign of "understanding." The answer is that the creation of these programs involved essentially no compression at all. One can readily get a computer program to parrot answers to a fixed set of questions by simply programming in the list of answers. Tell the computer if it receives question A, give answer *a*, and if it receives question B, give answer *b*. But a list of answers is not compressed; it is a program as long as the number of answers it can give. In contrast to the process of training a neural net, this storing of answers requires extremely little computation. And it is not at all constrained: one can program in any list of answers one chooses. A parrot may impress for a few seconds by speaking some complicated phrase, but it has no understanding of what it is saying. A computer program that is not compressed is not much better than a parrot and will be tripped up when it gets to a novel question.

Of course, many AI programs are more sophisticated than simple lists, but even so, they have not been produced by a computationally intensive optimization process. Rather, they have simply been written down by people. People are smart, but they don't seem to have the capability of doing enough hard optimization to produce truly compressed programs. Constructing extremely compressed programs that extract the structure of complex data is a very hard optimization problem requiring extensive computation such as is done in training neural networks or by evolution. We are no match for our computers at solving hard optimization problems and very far indeed from the computational capabilities that evolutionary history brought to bear on the problem. Human-created programs, for example, the AI programs called expert systems, thus do not reflect Occam's razor in the way human thought does, and so do not display understanding.

Another problem with standard AI and neural net approaches is that they typically throw out much of the structure of the world before they start. To understand language, for example, one must understand how language is about the world, but the academic divisions within computer science treat language as divorced from vision, as divorced from planning, as divorced from the world generally. Language is usually treated as pure syntax before one begins, so there is little hope of recovering the semantics. Similarly, planning is often treated in the scientific literature as independent of the specific knowledge about a particular domain for which one wants to plan, and in particular as independent of human knowledge about topology and geometry. By dividing up the world into academic subdisciplines such as language, planning, and vision, computer scientists are throwing out the relationships that the compressed program of the human mind exploits. I believe that any approach aiming to achieve understanding must be much more holistic and must evolve a compact computer code that compresses much experience about the world.

This picture of compression as understanding also readily explains why the guts of heavily compressed programs, for example, the internal representations of some trained neural networks, are inherently hard to understand. The argument, as further elaborated in chapter 6, is that understanding comes from having a very compressed description. But an understanding of the guts of a compressed program would then be an even more compressed description. Such will not generally exist.

Now, it is a huge step to go from simple curves, or even complex curves such as neural networks, to the kind of thought, understanding, and consciousness that we observe in human beings. In fact, neither neural networks nor any other function class that only classifies presented examples can model the mind well. Rather, we need to talk in terms of powerful computer programs. The program of mind does not simply represent or mirror the world; rather, it knows how to do complex computations about the world. It can do things like output algorithms to address whole new problem classes it has never seen before. It does not just mirror the compact description of the world; it exploits this structure to plan and to compute.

Finding a compact description of the world is already a hard task. But given a compact description of some computational problem, computing how to solve the problem is a separate impressive feat. Nonetheless, this exploitation of structure has arisen through program evolution. Evolution has produced minds that not only mirror the structure of the world but do amazing calculations about it.

What does it mean to exploit compact structure, and how can programs evolve to exploit compact structure? A first comment is that the mind does not arise from the kind of input-output classification training discussed, for example, in the neural network literature but rather from a process more like what the computer science

literature calls *reinforcement learning*. In reinforcement learning a robot interacts with its environment and is rewarded when it behaves correctly. Thus, the robot must learn to sense the appropriate features of the world, to compute what to do, and then to act. In a complex world it cannot simply sense and react through a simple function. It is rewarded only when it correctly decides what to do, so it is trained directly not only to reflect structure but to exploit it. I argue that our ancestors were trained by evolution to sense and think the right thoughts and take the right actions. We are not just reactive systems but have learned to do the right computations as well.

Three approaches to reinforcement learning are discussed in chapter 7. The first approach, which is the most widely studied in the literature, essentially consists of memorizing what to do in every state; it serves in this book as another example of a program that does *not* shed much light on the mind. Such memorization does not involve any compression and hence does not invoke Occam's razor. Memorization of this type can work only in very simple regimes because memory by itself can never tell you what to do for situations you have never encountered before. Memorizing and understanding are at opposite extremes. My point of view throughout is that it is Occam's razor—the finding of very compressed representations—that leads to understanding.

The next approach is using neural nets to do compression in reinforcement learning and to achieve generalization to environments never before seen. This builds on the formal Occam's razor results, and hence, for simple enough classes of nets, is relatively well understood from a formal perspective and does succeed empirically to a certain extent in learning and generalization. However, I critique this well-studied approach, arguing that in many environments it is conceptually flawed, that it will never get sufficient compression in complex environments, and that it will never handle reflective thought because the simple kinds of neural nets that computer scientists know how to train are essentially reactive. The neural representations people typically study are just not powerful enough to represent the kind of processes that are going on in the mind.

It seems very likely that the kinds of neural circuits studied provide a good model of certain brain functions such as early vision, but it seems unlikely that they are a good model of higher mental processes. Although it is true that more general classes of neural nets could simulate the actual neural circuitry of the brain, it does not follow that this is a fruitful way to talk about thought. To talk about thought fruitfully, at least along the line of attack in this book, one must be able to discuss the compactness in the algorithm. The compactness in the program of mind lies in the DNA and in the process by which the neural circuitry is constructed. The neural circuitry

itself is rather large, at least if one simply counts adjustable weights, but it has an underlying compact description.

When one writes a computer program, what is written is called source code. This is typically relatively compact. A computer analyzes the source code and produces an executable—very detailed instructions that say exactly what the computer should do at each step. This is typically much more voluminous than the source code. People would not typically look at an executable and try to understand it—it's too long and messy for easy human understanding.[3] The neural circuitry is, in my view, akin to an executable. The DNA is more like the source code. Looking at the neural circuitry is not, I suggest, the best way to intuit how the program works any more than we would look at the executable of an application like Microsoft Word. This is the more so here because the focus is on Occam's razor as a source of the mind's power, and the focus should therefore be on the most constrained part of the process.

A third approach to reinforcement learning is program evolution. I propose to study the possibility and nature of an evolved, extremely compact program, consistent with vast amounts of data, that exploits the deep structure of the world. In principle, such a program can exemplify Occam's razor in a very strong way. I hypothesize that if one evolves a very compact program that acts well in the world, then that program essentially understands the world and that program's syntax has gained semantics. I hypothesize that this is what the mind is, and that the difference between mind and ordinary computer programs comes because the mind reflects such an extraordinarily compressed program, which not only reflects the structure of the world but exploits it in complex ways (see chapter 8). Much of this book is devoted to explaining how such programs can be produced, how understanding arises, what the program of mind looks like and how it arose, and how it exploits the structure of the world and understands.

To get some intuition into what it means to exploit structure, consider how you know that the number 98667500989443658 is evenly divisible by 2. The answer is, roughly speaking, that while there are an infinite number of integers, they are all defined by just a few axioms. Thus, they have a very compact structure—there is a very small description that defines them all. You know tricks that exploit this structure to do various calculations, such as rapidly deciding whether large numbers are even. This is an example of how structure can be exploited for computation, and how you can have a simple subroutine or module in your mind that exploits that structure.

Similarly, the world has an enormously compact underlying structure. Considered as simply a collection of states with no structure, the world would be unimaginably

vast. But in fact, a quite compact program—it may be much smaller than the source code for Microsoft Office—is capable of dealing with the world's complexity.

This view addresses a number of questions, for example, the nature of objects in the world, a question hotly debated among philosophers for millennia. What are objects? Are they just in your mind, or is there an outside physical reality to them? For instance, if you go to a friend's house and sit at her table, how do you identify as a cup a collection of atoms that you have never before seen and that may have a different color, shape, and size than any other collection of atoms you have ever seen? In what sense is a newspaper one object? Are there really electrons in the world, even though nobody has ever seen one? Is there such a thing as the Platonic ideal of a circle?

The answer to these questions, I suggest, is that the mind is an evolved program that exploits the compact underlying structure of the world. The world has potentially a vast number of states but is compactly described in terms of objects, and the mind does calculations exploiting the compact description. Your mind's program "knows," for example, that it can poke your finger through the handle on the cup and tip it up so you can drink from it. That is to say, it contains some computer code, a small subroutine, that can execute this operation. It comprises many routines that exploit the fact that the world is compactly describable in terms of interacting objects. The objects really exist, or the compact description would not, and they are essentially defined by the mind's code. The mind's code corresponds to reality, or it would not make so many correct predictions.

Analogously, electrons were posited precisely because they arise in an extremely compact description of a vast number of physical phenomena. Physicists claim that they can ultimately describe all the phenomena in the universe in terms of a handful of equations, and electrons appear in a fundamental way in this compact description. The fact that electrons appear in such a compact description implies that there is reality corresponding to them, and the representation of electrons in the program of mind that results is essentially the definition of electrons in the world. The reality of electrons and your knowledge of electrons are thus directly analogous to the reality of cups and your knowledge about cups. Both arise from Occam's razor.

Of course, a better scientific theory might be found with a slightly different notion of electrons. That theory would have to be consistent with current theory inasmuch as current theory describes phenomena so well, just as Einstein's theory of gravity is consistent with the predictions of Newton. Electrons (and cups) exist in the sense that they are features of the program of mind that understands the world, and in the sense that they accurately describe the world, but not necessarily in the sense that the description is perfect.

To develop more intuition about what it means to exploit structure for computation as well as about program evolution and how an evolved program might come to understand, I discuss the solution by people, by AI programs, and by evolutionary programming of some standard benchmark problems, such as Rubik's cube and the simple stacking of children's blocks. Such problems are of course much more limited than a general analysis of concepts, but for this reason they can be discussed in concrete terms to help us get a firmer grasp on these concepts.

Evolving a program to understand is a hard problem, and in order to make progress on this I had to take some inspiration from observations regarding the qualitative nature of the program of mind. So before talking about structure exploitation and program evolution further, I pause to make some remarks regarding the nature of the program of mind.

For a large number of reasons, compact code for dealing with complex phenomena invariably has a modular structure. One doesn't sit down and write a program as one integral whole. Rather, one divides the problem up into subproblems and writes *subroutines*, sometimes called *objects*, *modules*, or *functions*, to solve the pieces. The whole program then is a complex society, composed of multiple modules that refer to each other, with each module charged with some particular task, achieving a division of labor.

It is impossible to construct code, or to evolve it, or to debug it, unless it is modular. If every part of the program interacts with every other part, then any change breaks so many things that it is impossible to make progress in constructing the code. On the other hand, if the program can be constructed module by module, it may be relatively straightforward to continue to improve it and add new functionality because at each step it is necessary to modify or produce only a relatively well-contained module. This can be expected to be true not only for human authors but for evolution as well.

Modularity is a powerful way to get compact code that exploits structure. If, as I walk, a module in my mind positioning my ankle is independent of a module contemplating my future plans, then each of these modules can be compact. In other words, code dealing with the world will be modular because the world is modular, and exploiting the modularity gives a compact program that can calculate how to interact with the world. By having modularity, one gains combinatorially in dealing with the myriad possible states of the world. A relatively small number of modules can interact to represent and deal with an exponentially huge number of world states. Moreover, the code is compact to the extent that modules can be reused many times and preferably in many contexts. A large part of the reason that standard neural

networks are often an inappropriate model for mind is that they don't readily lend themselves to describing such a modular structure.

Researchers in many different disciplines—computer science, cognitive science, neuroscience, evolutionary psychology, and others—have reached a separate consensus within their several fields that the mind is modular. The different viewpoints give interesting aspects of what the modules look like and how they interact and are coordinated. A number of these different pictures are examined in chapter 9—cognitive deficits coming from localized damage, psychophysical experiments that indicate the existence of a dedicated module for reasoning about social obligations, and so on. But as always, I take the point of view that the mind is a program. Hence these modules represent modules in the program. The upshot is that you have, I believe, modules representing different concepts, representing different objects, representing how the objects act, what their properties are, how you deal with them, for example, how you go about lifting a cup to your lips and what to expect when it gets there. These modules call submodules that are reused in many different ways. Many of the submodules that you use for dealing with cups are also used for dealing with forks or cars or computers.

I want to mention here two additional points about modules. The first is that the metaphoric structure of language gives a window on the modular structure of the mind's code. Lakoff and Johnson (1980) have pointed out that our language is deeply metaphoric. For example, time is money, in that we spend time, borrow time, waste time, lose time, save time, and so on. Such built-in metaphors are incredibly pervasive in language. I believe this provides a picture of code reuse. Time is money because we have a module for valuable resource management that we reuse to deal with time. From this point of view we have many modules that call one another in complex ways. This massive code reuse yields a very compact program that understands and deals with the world.

A second point is that we can learn whole new modules. The main example, discussed in chapters 8 and 9, comes from computer science. I've said repeatedly that the mind exploits the compact structure of the world to solve problems rapidly. In fact, exploiting structure to solve problems rapidly is the central problem of computer science. The mind is in many ways better at this than computer scientists because the mind has access to many modules discovered over evolutionary time that exploit the structure of the world. Computer scientists have, however, developed a whole bag of tricks for exploiting structure. Examples of such tricks are methods called divide and conquer, recursion, branch-and-bound, dynamic programming, gradient descent, and many others. Each of these is a trick for exploiting structure

in problems. Using these tricks, computer scientists can program computers to solve much bigger problems than could be solved without them, problems that are vastly too hard for the unaided mind to solve. A typical research paper in computer science consists of applying one or more of these tricks to a novel problem. A breakthrough research paper finds a new trick.

I suggest that when you learn at university about one of these tricks, say, recursion, you are really building a module in your mind that knows how to search for recursive solutions to new problems. This is a fairly sophisticated module, calling a lot of previous modules you have in your mind, but it is pretty evidently distinct from what you knew before. It clearly is something that you explicitly learn. In fact, I hope readers who do not already know what recursion is will gain some idea of what it is from my description in chapter 8. If the mind is a program, which after all is the central premise of this book, then this recursion ability is presumably a new module added to that program.

I expect that a large part of the way our powerful minds have been formed is by accretion of new modules built on the existing structure, giving it new abilities. Each module may call previous ones. Constructing a program as complex as mind is an incredibly complex task, but it is much easier if it is done incrementally. Most of the discovery of such new tricks for exploiting structure has, I expect, happened over evolutionary time. But the example of these new modules shows that some of it happens in our lifetimes as well. Human thought is so powerful because we can discover new modules and pass them on. It only took one computer scientist to discover a trick like recursion, and he could add it to the human repertoire. I discuss in later paragraphs the relation between modules built by evolution and modules discovered or learned during life.

To recap the argument to this point, I have proposed that the mind is an evolutionary program. Because it has a very compact description, the syntax of the program corresponds to real semantics in the world. Because the world has structure, and because the program of mind has evolved to exploit that structure, it is able rapidly to compute and output algorithms for addressing problems in the world. Understanding comes from the compactness and the ability to exploit structure for computation. The program has a modular structure, with modules corresponding to concepts calling other such modules. I say concepts because they can be seen as having semantic meaning, which has arisen during the production of compact code capable of dealing with vast numbers of situations.

To illustrate what it means to exploit structure for computation, I discuss the solution of some standard AI benchmark problems, such as Rubik's cube and the

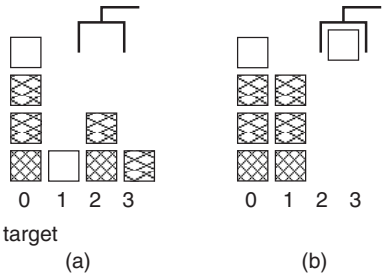


Figure 1.1
In block-stacking problems one is presented with four stacks of colored blocks. The last three stacks taken together contain the same number of blocks of each color as the first stack. The solver controls a hand that can lift blocks off the last three stacks and place them on the last three stacks but that cannot disturb the first stack. The hand can hold at most one block at a time. The goal is to pile the blocks in the last three stacks onto stack 1 in the same order as the blocks are presented in stack 0. The picture shows a particular instance of this problem with four blocks and three colors: *a*, the initial state; *b*, the position just before solution. When the hand drops the white block on stack 1, the instance will be solved.

simple stacking of children's blocks (figure 1.1). Such problems are more limited than a general analysis of concepts, but for this reason they can be discussed in more concrete terms.

Block stacking as presented in this book is not a problem but rather a class of problems. The class contains infinitely many problems because infinitely many blocks can be arranged in infinitely many ways, but the class has a very compact overall description, namely, a paragraph of text that describes what the block-stacking problems all look like (see the caption of figure 1.1). Block-stacking problems are thus a little like the integers: there are an infinite number of them, but a short description shows they have much structure in common that can be exploited to do computations of various sorts.

A person looking at these problems can in short order produce an algorithm capable of solving arbitrary block-stacking problems rapidly, by exploiting the structure of the problem. That is, you can describe an algorithm that works for problems having this particular short description, just as you can rapidly tell whether any long number is divisible by 2. Think about how you would solve any problem in this class, if you haven't already.

By contrast, standard AI planning approaches don't exploit the structure of the problem at all; instead, they do a vast brute-force search. I mentioned before that the planning literature more or less abstracts the planning problem to the point of planning without using special knowledge about the particular domain, such as the domain of stacking blocks. Instead, standard planning algorithms simply search over

all possible configurations, in this case, all possible block stackings, looking for one that works. They thus treat the problem as simply a huge list of all possible action sequences, ignoring the fact that the problem domain has a short description, an exploitable structure. Since there are a vast number of possible block stackings, this approach fails for any problem of interesting size.

You, a human being, bring to bear understanding of topology, geometry, and goals. You had, I expect, modules in your mind that know about these concepts before I presented you with these block-stacking problems. So, you don't search at all over block stackings; you simply output an effective procedure that solves the problem. If you search at all, it is over different approaches, different algorithms. You are thus working in an entirely different space than the AI approaches, and on a very different problem.

A trained computer scientist has yet another module in her mind: a module for recursion. This allows her to construct a yet more efficient solution to the problem. That is, this module exploits the structure of these block-stacking problems to output an algorithm that solves them very rapidly.

Chapter 10 discusses computational experiments in which programs are evolved, starting from pure random computer code, to solve a few well-known problems such as Rubik's cube (which is only partially solved) and the class of block-stacking problems.

Evolving a compact program is extremely difficult for at least two reasons. First, the space of possible programs is enormous and thus hard to search. Second, small changes in a program, even single-line changes, can completely disrupt its behavior. This makes it hard to figure out how to improve a candidate solution. It leads evolutionary approaches to get stuck in local optima where any small change doesn't improve performance and where one has to somehow discover a larger change in order to progress. In fact, it is a fundamental mystery how evolution has succeeded in evolving the program for building us, even given the 4 billion years it has been working. In order to evolve these compact programs using only a few days on a desktop computer, my colleagues and I employed the following ideas.

A powerful way to exploit structure and produce a compact program is to produce a modular program that exploits natural modular structure in the world. Ideally, small interacting modules can be found that can be reused for different problems, contributing to compactness. Moreover, if the problem can be factored, that is, if a product structure can be identified, one can hope to search only for modules rather than having to find the whole program at once, which makes the search much smaller. Our picture of the mind is of a complex adaptive system composed of modules that interact in complex ways, forming a very compact program in total.

We were inspired to use the metaphor of the economy to automatically divide and conquer hard problems down into a collection of interacting modules. The real economy is just this: a system where millions of individuals interact under simple rules, and which has the property of coordinating activity among the individuals in a meaningful way. Consider, for example, the common event of going into a stationery store to buy a pencil. How did the pencil come to be in the store? There is no one person on earth who knows all about how to make that pencil. Lumberjacks, skilled at cutting trees, produce wood for it. Chemists have knowledge used to produce the yellow paint. Miners find the graphite in Ceylon. Smelters know how to produce the brass ring that holds the eraser on. Farmers in Indonesia grow a special rape seed for oil processed into the eraser (Read 1958). These people have no common language or purpose. They don't even know of each others' existence. Adam Smith's "invisible hand" organizes widely distributed knowledge to perform a computation, the mass production and distribution of pencils, that would be called cognitive if a human being were even capable of it.

Motivated by this intuition, we constructed an artificial economy of agents, where each agent is a computer program with an associated wealth. We imposed an economic framework that distributes "money" among the collection of agents in such a way that the agents are motivated to collaborate on solutions to problems. We then were able to get our systems to evolve, starting from literally random code, to collections of modules that collaborate to solve hard problems. We have empirically evolved systems that solve arbitrary block-stacking problems of the type discussed, systems that halfway unscramble arbitrarily scrambled Rubik's cubes, and systems that solve several other classes of problems, including a real-world application to Web crawling. These compact programs thus exploit the underlying structure of huge problems in the world with which they are presented. The solution represents the interaction of many modules, each of which performs some small task. The action of the modules evolves careful coordination, resulting in the overall algorithm. Although these tasks are much simpler than many faced by the mind, and although the programs we've evolved are of course vastly less complex or capable than either the United States economy or the mind, I hope this example will give some insight into how complex modular systems can evolve to exploit the structure of the world and into the nature of understanding.

The key to getting our artificial economies to work turns out to be imposing simple notions of property rights and "conservation of money." When we impose these, we get a system that self-organizes in a natural way for simple, intuitive reasons. Each agent, maximizing its own money, learns to perform some small task that contributes to the performance of the whole system. If property rights or conservation of money

are broken, however, the system evolves great complexity but does not evolve compact programs that solve problems. I discuss in chapter 10 the evolution of cooperation generally, and identify generic phenomena that afflict a wide variety of complex adaptive systems and interfere with evolution of cooperation.

At this point, I step back to discuss in more detail questions arising from the branch of computer science known as complexity theory. How can computations as complex as those performed by mind and by evolution in creating mind have been performed so rapidly?

As I've said, exploiting structure to solve problems rapidly is the central problem of computer science. One of the things computer scientists believe they have discovered about this problem is that it is not generally solvable. Given a compact description of some problem class, it does not follow that there is any fast way to exploit the structure to solve the problems rapidly. Roughly speaking, computer scientists call a problem class *NP-hard* or *NP-complete* if it is compactly describable but they don't believe there is any possible algorithm that solves it rapidly (see chapter 11). The different block-stacking problems are all rapidly solvable because they have a structure in common that can be used to solve them. They all look pretty much alike, at least as viewed by a person. By contrast, any NP-complete problem class provably contains such a diversity of problems that in an important sense most of them look utterly unlike the others. Hence there is no approach that works for them all.

The phenomenon of NP-hardness provides a conundrum for understanding mind, for two reasons. First, computer scientists' intuition is that almost all problems are NP-hard. If this is so, one would expect the problem of exploiting the structure of the world to be NP-hard. If so, it is a mystery how evolution solved it. Computer scientists expect that NP-hard problems, roughly speaking, can in general only be solved by searching through all possibilities. But the space of possible programs is so immense that even given the billions of years and vast numbers of trials that evolution has used, it cannot have explored even the most infinitesimal fraction of the space. Second, this worry is reinforced because the AI literature is full of results showing that problems solved by people, such as reasoning, planning, solving certain classes of block-stacking problems, and so on, are NP-hard, which is a puzzle because people routinely solve them, if only approximately, and indeed solve them rapidly. Since computer scientists essentially believe that NP-hard problems are not rapidly solvable, it is a conundrum how human intelligence can exist. This problem is, if anything, compounded by the view emphasized in this book (which is not often discussed in the literature) that a lot of what human thought does is to output algorithms. When you solve Blocks World (see chapter 8), you output an algorithm for

the problem. But the space of possible algorithms is so immense that it is amazing you can do this rapidly.

I do not offer a definitive answer to this puzzle but discuss a number of interesting possibilities. Our evolutionary computer experiments did succeed in outputting some interesting algorithms, albeit in a relatively restricted example. Computer scientists have typically thrown out the useful structure that the mind employs before formulating hard questions. Computer scientists apply their understanding to real-world problems, extracting already compressed problems with too little remaining structure to be rapidly solved. For example, the fact that subproblems of intelligence are hard is not in itself alarming but may simply arise because the academic division of problems into subproblems throws out the useful structure.

But the main suggestion I make, in keeping with the overall thrust of this book, is that the overconstrained nature of the world again comes to the rescue. Occam's razor says that when there are a lot of data to explain, and thus many constraints on a compact solution, the compact solution must reflect the underlying simple structure of the world. Something similar turns out to happen in at least some instances of NP-hard problems. When there are many constraints on the solution to a problem, there is usually only one way to extend a partial solution. Then you can extend it rapidly. I believe this may describe a critical and common feature of thought. Our understanding of the world is so compact and thus so constrained, that only one or a few possible ways exist to extend it that make sense at any given time. Thus, thought flows forward, each step following more or less inexorably from the last. We can understand speech because, applying all the constraints that come from our understanding of the world and from grammar, only one or a few ways of understanding each sentence make sense. We can play chess because, given our understanding of the position, only a few lines of play must be searched. These human thought processes are in stark contrast to standard computer science methods for analyzing speech or playing chess, which are not based on such a compact understanding and thus must search a vast number of possible extensions.

To state this in a slightly different way, human thought is fast because we search only possibilities that make sense. That is, our thought is organized to search semantically meaningful possibilities only, as units. Semantics arises because the world is highly overconstrained, and by finding code that knows how to deal with the world but is extremely compact, evolution has captured and learned to exploit those constraints.

Evolution built creatures that had to do the right thing and do it fast. It built from simple behaviors toward more complex computations but at each step stuck to a domain where each step followed from the last without much search. We are built

with an understanding based on an enormously complex collection of modules computing different concepts, and we evolved to coordinate their actions purposefully. Many of these modules predate human beings: we are built on concepts that have been useful to understand the world for eons just as we are built out of enzymes that have been useful for eons. The structure of our program is so constrained by the world and so similar from person to person that we can understand each other and even communicate whole new concepts. Evolution has discovered these modules one step at a time, at each step figuring out how to add some new, useful concept. And people have added to the store of program modules over time, at each step discovering a new module by making a small enough step so that it can be discovered without impossibly hard search. The reason we are able to discover and add new useful modules to the program is perhaps that the understanding of the world we inherited from evolution already contained so many useful modules, interacting in such a coordinated fashion, that we are quite constrained in how we can add new modules, new thoughts.

Evolution itself has learned to search in semantically meaningful directions, for example, using computational units that have acquired semantics. Evolution long ago discovered genetic regulatory circuits coding for development that involve a small set of toolkit genes, such as Hox genes. These genes came to have semantically sensible meanings, so that, for example, by expressing one specific gene on a fly's wing it is possible to grow a well-formed eye there. Evolution then proceeded to experiment with new body designs by building programs out of such semantically meaningful modules. Evolution is manipulating a highly compressed representation of the world, and thus in a sense it too is applying understanding to the process of discovery. I suggest that this in large measure is how evolution has produced such amazingly powerful programs in only 4 billion years.

Understanding the evolution of compact programs is critical to understanding mind because it is evident as a matter of historical record that a process of training a compact program did in fact generate our minds. The compact program was encoded as DNA, and the training process was evolution. This process is analogous to the reinforcement learning model by which we trained our artificial economy. Evolution trained on data for 4 billion years, involving (as I estimate) perhaps 10^{35} or more separate learning trials, and in a sense distilled all this learning into a compact expression in the DNA. The DNA program is quite compact compared to this massive training.

A bit is the minimal unit of information, the amount communicated by a single symbol that can take the value 0 or 1, in other words, the amount of information in a single choice between two alternatives. A byte is a word of eight bits and thus can

specify a choice among 2^8, or 256, alternatives. I estimate that the DNA program has effective information content of roughly 10 million bytes. To put this in perspective, this book is a string of text (omitting pictures) long enough to allow specification of about 1 million bytes. Thus, the 10 megabyte estimate for the information in the DNA is approximately equivalent to what could be specified in ten volumes of text the size of this one. For all its massive capabilities and extensive training, the effective content of the DNA program is a fraction of the size of the source code for programs like Microsoft Office. But the computing machine it specifies (for instance, you) looks like it understands the world and can be expected to act as a human being would in new situations.

Schrödinger (1944) considered the question of the nature of mind but felt it to be unsolvable. He wrote,

> [The fact that life is produced by simple mechanical interactions of proteins reflecting genetically stored information] is a marvel—than which only one is greater; one that, if intimately connected with it, yet lies on a different plane. I mean the fact that we, whose total being is entirely based on a marvelous interplay of this very kind, yet possess the power of acquiring considerable knowledge about it. I think it possible that knowledge may advance to little short of a complete understanding—of the first marvel [life]. The second [mind] may well be beyond human understanding. (31)

Yet I am proposing that the analogy to Schrödinger's explanation of life is exact. The answer to the mystery of life and the answer to the mystery of mind (thought) are one and the same. It is given by the information flow, which in each case is provided (largely) by the genome. In the case relevant here, understanding thought, the genome encodes a compact expression that gives rise to understanding in the mind. This genome is quite a compact expression, which grows out (interacting with the world) into an immense flowering—the mind—much as the genome grows out (interacting with the world) into the body.

Since Schrödinger's day, our understanding of life has improved considerably. We have unraveled the structure of DNA and are beginning to understand the complex series of reactions, regulatory networks, and chemical cascades by which it goes through the algorithmic process that leads to the body. Few scientists today would suggest that there is anything mystical, anything not ultimately explainable by science, in life. It is a goal of this book to further the process of understanding the algorithmic process that leads to the mind.

The preliminary sequencing of the human genome has produced a surprise: human beings are now thought to have only 30,000 or so genes, perhaps one third as many as had been once expected. The 100,000 previously predicted had already been thought to be a small number. Many creatures seemingly less impressive than ourselves have

as many genes as we do. Yet, from the point of view of this book, the low number of genes making up the human genome should not be surprising. Our focus is on compression inducing Occam's razor. A more impressive intellect can result from a more compressed genome, not necessarily just a longer one.

Many computer and cognitive scientists, particularly neural net practitioners, believe that the key factor in producing the mind is learning during life, not structure imparted in our DNA. They argue that we are born in a state that is largely *tabula rasa*, knowing nothing but possessing a general learning algorithm capable of learning about the world (maybe any conceivable world). The computational learning literature is largely oblivious to the interaction of learning during evolution and learning during life, and many in the field find the notion of special-purpose modules built into the mind by evolution to be extremely controversial.

Nonetheless, in chapter 12, I discuss how modern computational learning theory has shown that inductive bias, that is, the built-in predisposition to learn one thing rather than another, is crucial in being able to learn in complex environments. I have emphasized from the beginning that Occam's razor—finding a short explanation—is crucial in learning. But finding a short explanation involves first fixing on a language for describing explanations (a way of describing programs) and then being able to seek an explanation rapidly, that is, a learning algorithm. These things constitute inductive bias. Without these, you cannot learn. Once you have specified a language and a learning algorithm, however, what you can learn is not fully general—you will learn some things much more readily than others. And the explanation you will come to is more or less predetermined. The same algorithm and language will repeatably lead to a similar explanation when confronted with the same world. Your DNA can reliably be counted on to produce something with a mind rather like yours in the important ways if it is executed in contact with any roughly similar environment. The guts of learning during life are thus in the DNA.

There are a number of arguments to make clear the importance of the DNA code in predisposing learning. Some of these are the following. First, the DNA is where the compactness is: the DNA code is compact, but the brain is not. For example, the brain has at least 10 million times as many neural connections as there are bits of information in the DNA code. So, if one believes in Occam's razor, if one believes that it is the compact specification that leads to the mind's power, one should focus on the DNA. Second, finding the DNA is where almost all the computation has gone. Learning is a computationally hard problem, requiring vast computational resources. Vast computation has gone into evolving the genome, and only a relatively small amount goes into learning during life, which we mostly do in real time. We are predisposed to learn so rapidly that there is no time for computation; how we

fit new facts into our improving program is so constrained by what we already know and by built-in algorithms that it is fast and largely predetermined. Third, if we examine the learning abilities of various creatures, we see numerous examples of explicit inductive bias: animals are predisposed to learn certain types of things. They are not general-purpose learners at all. People, too, are predisposed to learn certain types of things. Chapter 12 describes a case study providing compelling evidence that we are wired to learn grammar rapidly.

Fourth, the nature of evolutionary computing is such that we could not have helped but evolve to be predisposed to learn. We often think about development (the growth of an embryo into an adult) as distinct from learning, but actually development takes place in contact with the world and must generalize over variations in its environment in a way quite analogous to how learning must generalize flexibly to unseen data. Development of the brain takes place in contact with sensory input, and the nature of evolution is such that the brain naturally comes to depend on this sensory input to develop correctly. The DNA codes for a program that is evaluated in the presence of an environment, resulting in a human being. In evolution the DNA is tweaked, the resulting creature develops in contact with the environment, and successful creatures propagate. This process leads to DNA coding for development, in contact with the environment, of the right structure and the right behavior.

This same evolutionary process, however, has led to a large expansion of the complexity of program execution through use of external storage in the form of culture. Learning the right things during life, that is, having an algorithm for development that interacts with the environment and develops the right behavior, is a crucial part of behaving the right way and thus greatly affects evolutionary fitness. Once parents became a part of the environment in which the child developed, the development process naturally evolved (as the DNA was tweaked and what worked was kept) to exploit passage of knowledge (programs) from parent to child. Culture came to interact with development and to affect the evolution of the genome. Such an effect began long before there were human beings, but it became more important once the development of language permitted them to pass on much more programming.

This book argues at some length that the learning and thought we do during life are heavily biased in that they are built on top of semantically meaningful modules that evolution long ago wired into the DNA. But human beings have built a major structure on top of these modules, much as mathematicians have discovered many theorems of number theory by working out the implications of a small set of axioms. The theorems are implied by the axioms, but it required massive computation over generations of mathematicians to work them out. Similarly, humankind discovered a

massive hierarchic structure of new modules over history, building these programs out of modules that had acquired semantics through Occam's razor and passing on these programs through language. As discussed in chapters 13 and 14, we judge the value of the new modules we create using an internal evaluation function wired into the DNA.

It is intuitively straightforward to translate folk wisdom regarding language into a model of mind as an evolved program composed of modules that compute semantically meaningful concepts. Presumably, words are labels for modules, and grammar indicates how the modules are composed into larger computations. By speaking, we can describe semantically meaningful computations (thoughts) going on in our minds. Such a mapping was implicitly assumed when I discussed metaphor. If one accepts that language consists of attaching labels to preexisting computational modules, many of them already present in honeybees and dogs, then it is relatively easy to understand how children learn words so effortlessly. All that is needed is to attach a label to a computational module, and the particular module indicated will often be quite salient, because we share inherited inductive biases in the form of modular structures.

As I've mentioned, code discovery is hard. The process of finding new, useful, semantically meaningful computational modules building upon previous ones is computationally very hard, requiring massive computation. It can only advance by a series of small steps. But humans have added something to this that no animal before us has. Because we speak, we can guide other humans to rapidly construct in their minds new modules that we discover. For this reason, the computation of the computer code of human minds has extended over tens of billions of individual minds. I estimate that the total number of computing cycles brought to bear this way may be comparable to the total number of trials that evolution brought to bear on producing a chimpanzee.[4] This extended computation affects not only what is usually thought of as technology but also things usually thought of as more basic to our minds, such as our ability to reason about how others see things. Sometimes called a theory of mind, this is an ability in which we are far in advance of other primates and which is informed, for example, by our bedtime stories and our reading of fiction.

A number of authors have argued that human minds must be qualitatively different from those of other creatures because we have evolved such superior abilities, for example, a new ability to handle symbols. Exactly what this means has never, to my knowledge, been made explicit. I argue that no such qualitatively new (and poorly understood, at least by me) ability need be posited. Such an advance may exist, but it need not be posited to explain human abilities. From the viewpoint of this book, all our abilities can be more simply explained by language purely as a communicative

medium rather than as a symptom of some new symbolic ability. Once one posits that the mind is a program that exploits the structure of the world, realizes that the discovery of this program requires vast computation, and understands that the program can be built incrementally with new modules building upon what went before, the difference between animals and humans is immediately explainable. Because we have cumulatively, over centuries and millennia, brought so much more computation to bear on the hard problem of producing a program of mind that exploits the structure of the world, we have built a much more powerful program, adding module upon module to what came before language. Thus, almost all the difference between a human being and a chimpanzee can potentially be explained in terms of much more powerful and effective nurture.

It took billions of years to build the conceptual structure but only tens or at most hundreds of thousands of years for language to evolve from apes to its modern power. If all that is necessary is to apply labels to existing modules, it seems relatively easy to understand how language, once started, would rapidly improve. But why did it take so long to get started? The evolution of language must have been held up in some local optimum. Several possibilities for what this sticking point might be are discussed in chapter 13, including interesting proposals by Martin Nowak and collaborators that the bottleneck follows directly from Claude Shannon's information theory. However, I also posit the following potential bottleneck suggested by experience with evolutionary programming experiments.

Many experimenters have found empirically that it is very hard to evolve programs that communicate usefully between modules (or even between a program and the computer's memory) because one has a chicken-and-egg problem. The evolution cannot discover that it is useful to speak (or write) a word until some agent knows how to understand it and make profitable use of the knowledge, and it cannot produce an agent that can understand the word until someone is speaking it. So, evolution can't get started utilizing communication without a big leap. According to this picture, we have the concepts before we learn the words, so learning the words is easy, but only as long as we realize that we should be learning words. Experiments show in fact that animals can learn words if a person makes a concerted effort to teach them. But it's still a big step for evolution to discover teaching behavior in an environment where no words are taught or learned. It thus seems possible that the crucial discovery in evolving language was made by some pair of protohuman Einsteins who first realized that one should try to communicate through words.

Consciousness in all its many manifestations, sensation, and what we call free will are also natural and necessary products of the evolution of mind. Evolution dis-

covered that it was important to do computation in real time as the creature behaves. If everything were programmed purely reactively, a creature could not adjust its behavior to any change in its environment over time scales much less than a generation. It is far more powerful and thus evolutionarily fit to create a creature that can plan and reflect, and to effectively program this creature to try to maximize the propagation of its genes. Chapter 13 describes a succession of increasingly sophisticated programs: the minds of *E. coli*, the wasp, the bee hive, and the dog.

Evolution thus designed the mind for the purpose of making decisions leading to propagation of the DNA. But to accomplish this effectively, the DNA has to program the mind to make decisions, both short-range decisions and long-range plans, that favor the DNA's interest. The DNA has to build a program that makes sophisticated decisions based on local conditions and sensory information, but equally important, it has to control this system so that it does the right thing, that is, favors the interests of the genes rather than wandering off to some other calculation. In other words, it has to build in decision-making capability and, effectively, build in an internal reward function that the creature will seek to maximize (see chapter 14). Evolution thus leads to creatures that are essentially reinforcement learners with an innate, programmed-in reward system: avoid pain, eat when hungry but not when full, desire parental approval, and react to stop whatever causes your children to cry. These urges are detailed and complex—not all orgasms are identical—and moreover are not automatic—creatures must weigh one against the other, weigh long-range payoff versus short-range payoff in a sophisticated fashion. Reinforcement learning is all about calculating how to maximize reward over the long term, disdaining short-term rewards where necessary to achieve greater long-term ones. Even very simple creatures are no doubt vastly more sophisticated than the reinforcement learners in computer science simulations, which already weigh long-term versus short-term gain.

Thus, the minds of creatures have evolved to be something I'll call *sovereign agents*, goal-driven decision makers that strive purposefully toward internally generated goals. We look around us, and we look at ourselves, and this is what we see: sovereign agents. We need modules in our minds for interacting with all these creatures, including ourselves, in order to make decisions about what actions to take. To maximize long-term rewards we have to predict what other creatures will do and what we ourselves will do in the future. The way we understand this is by a computational module or modules that we call consciousness—that is, we attribute free will to ourselves and others, we attribute consciousness to ourselves and others.

After all, what it means for us to have any thought, for example, the thought that we are conscious, is that we are executing computational modules in our mind. The

attribution-of-consciousness module is a very simple, compact way of understanding the world. It is a very useful module, naturally arising through Occam's razor, applying widely to a host of phenomena.

Looking at mind from this evolutionary point of view makes natural something that some authors defending strong AI seem skeptical about: the unity of self. Daniel Dennett and Marvin Minsky, for example, have emphasized that the mind is a huge, multimodule program with lots of stuff going on in parallel. They doubt there is any single individual, any single interest; rather, they see a cacophony of competing agents. But, as we found in our economic simulations, the coordination of agents is crucial in exploiting structure. The program of mind was designed for one end: to propagate the genome. The mind is indeed a complex parallel program for reinforcement learning, but it comes equipped with a single internal reward function: representing the interest of the genes. Thus, the mind is like a huge law office with hundreds of attorneys running around and filing briefs but with a single client, the self. Because we are designed for complex and long-range planning—representing our genes' interests over generations and in widely different circumstances—exactly what the interests of the self are differs from individual to individual and over time. Suicide bombers, mothers, and capitalists are all striving to advance the interests of their genes as their respective minds compute those interests. But for all the modules in the mind and all the many computations going on in parallel, there is one central self focusing all the computation—one central reward being optimized—the resultant of the interests of the genes.

To make decisions that effectively favor our genes' interests involves an enormous computation. We run this evolved program, and it analyzes the world using its compressed description, as discussed throughout the book. For example, it understands the world in terms of interacting objects like cups and fluids and modules for valuable resource management. Analyzing the world like this involves discarding vast amounts of information. When we appreciate a collection of atoms as a cup, we abstract away little details about the shape and color that we could otherwise attend to. But evolution has told us these details are less important to making decisions, and so we engage in massive calculations of which we are unaware that discard all the unimportant information and pass on a processed picture of the world to other modules that output actions, for example, that speak appropriate words.

The portions of our minds that directly control our mouths have no access to all these earlier computations. That is what it means that we are unaware of these computations. We cannot report them. But, at the same time, the later modules do evidently understand the world in terms of the processed picture. At the upper levels of this complex computation, we have modules looking at the outputs of lower-level

modules and effectively understanding the world in terms of the processed picture. We can report on this because these modules directly control our words. This is what it means that we are aware of events in the world. We can report these events, we report them to others or to ourselves. Our awareness is thus nothing more or less than the higher portions of this evolved computation.

I suggest that this picture will, when we are done examining it, qualitatively explain everything about our consciousness and our experience of living. It will explain the nature of our words and of our thoughts. Nonetheless, I'm sure some readers will be unsatisfied with this straightforward mechanistic view of the nature of experience. How a physical system made of meat can possibly give rise to a subjective experience of being is "the hard problem," according to the philosopher David Chalmers. The philosopher Frank C. Jackson (1982), arguing that the way experience feels to us cannot possibly be explained from a physicalist perspective, wrote,

> Tell me everything physical there is to tell about what is going on in a living brain, and . . . you won't have told me about the hurtfulness of pains, the itchiness of itches, pangs of jealousy, or about the characteristic experience of tasting a lemon, smelling a rose, hearing a loud noise or seeing the sky. (127)

But I argue that the view advanced in this book explains everything, and does so economically. I specifically take on each of the challenges quoted from Jackson. He picked these, no doubt, because they are intense experiences, but they are intense experiences because it was important to evolution to build them into mind strongly, long before human beings evolved, and for that very reason it is straightforward to explain them qualitatively. It is hard to imagine, for example, how the subjective experience of the "hurtfulness of pains" could possibly be changed to make any clearer that it was built in by evolution to guide the behavior of the program that will advance the interests of the genes.

This will still not convince all readers because the nature of experience is so gripping. However, these remaining doubts are a failure of the imagination analogous to many that have occurred in the history of science. I believe I have presented in this book a simple, straightforward, mechanistic, self-consistent theory that explains all observations, including introspective "experience." It starts by positing (and justifying) the axiom that any thought is an execution of computer code and from this compactly explains every observation, subjective or objective. It explains what you say about your experience to me, and what you think about it to yourself, and what it means to think about it to yourself. To those who still say they don't understand this explanation of their sensation of experience, I reply, Our explanation addresses directly what it means to understand and can explicitly explain why you don't

understand. Thus, we have a self-consistent theory that explains everything, including the doubts of the dubious. By Occam's razor, it should be accepted unless a simpler alternative can be found.

Finally, in chapter 15, I consider the prospects for producing intelligence and understanding in a computer. If the exponential increase in hardware speed of computers continues for a few more decades, we will have machines that can compete in raw computational power with the human brain. Many authors thus predict that we will have smart machines. However, we will never in the foreseeable future have the computational resources to compete directly with evolution, and I argue throughout this book that most of the computation that went into producing our intelligence was done by evolution. Thus, I am a pessimist on the possibility of producing mind. On the other hand, chapter 10 gives some arguments that indicate evolution is very far from optimal at evolving intelligence. If this is so, then better algorithms, together with improvements in hardware speed over the coming decades, might plausibly allow the development of smart machines. One reason why I have engaged in this project is to think through the directions that might lead us there.

References

Abelson, H., G. J. Sussman, and J. Sussman. 1996. *Structure and interpretation of computer programs. 2d ed*. Cambridge, Mass.: MIT Press.

Ackley, D., and M. Littman. 1991. Interactions between learning and evolution. In *Artificial life II*, ed. C. G. Langton, C. E. Taylor, J. D. Farmer, and S. Rasmussen, 487–509. Redwood City, Calif.: Addison-Wesley.

Applegate, D., R. Bixby, V. Chvátal, and W. Cook. 1998. On the solution of traveling salesman problems. *Documenta Mathematica Journal der Deutschen Mathematiker-Vereinigung* ICM 3: 645–656.

Armstrong, D. 1999. The nation's dirty big secret. *Boston Globe*, November 14. ⟨http://www.boston.com/globe/nation/packages/pollution/day1.htm⟩.

Aserinsky, E., and N. Kleitman. 1953. Regularly occurring periods of eye motility, and concomitant phenomena, during sleep. *Science* 118: 273–274.

Ban, A. 2002. Interview with Amir Ban, ⟨http://www.chessbase.com/shop/product.asp?pid=94&user=⟩.

Banzhaf, W., P. Nordin, R. E. Keller, and F. D. Francone. 1998. *Genetic programming: An introduction*. San Francisco: Morgan Kaufmann.

Barkow, J. H., L. Cosmides, and J. Tooby, eds. 1992. *The adapted mind*. New York: Oxford University Press.

Baum, E. B. 1989. A proposal for more powerful learning algorithms. *Neural Computation* 1: 201–227.

Baum, E. B., D. Boneh, and C. Garrett. 2001. Where genetic algorithms excel. *Evolutionary Computation* 9 (1): 93–124.

Baum, E. B., and I. Durdanovic. 2000a. An artificial economy of Post production systems. In *Advances in learning classifier systems: Third international workshop, IWLCS 2000*, ed. P. L. Lanzi, W. Stoltzmann, and S. M. Wilson, 3–21. Berlin: Springer-Verlag.

———. 2000b. Evolution of cooperative problem solving in an artificial economy. *Neural Computation* 12 (12): 2743–2775.

Baum, E. B., and D. Haussler. 1989. What size net gives valid generalization? *Neural Computation* 1: 148.

Baum, E. B., E. Kruus, I. Durdanovic, and J. Hainsworth. 2002. Focused web crawling using an auction-based economy. ⟨http://www.neci.nec.com/homepages/eric/papertn.pdf⟩.

Baxt, W. G. 1990. Use of an artificial neural network for data analysis in clinical decision making: The diagnosis of acute coronary occlusion. *Neural Computation* 2 (4): 480–489.

Baxter, J., A. Tridgell, and L. Weaver. 1998. Knightcap: A chess program that learns by combining TD(λ) with game-tree search. In *Proceedings of the International Conference on Machine Learning*, 28–36.

Berlekamp, E., and D. Wolfe. 1994. *Mathematical Go: Chilling gets the last point*. Wellesley, Mass.: A. K. Peters.

Berliner, H. 1974. Chess as problem solving: The development of a tactics analyzer. Ph.D. diss., Carnegie-Mellon University.

Bickerton, D. 1981. *The roots of language*. Ann Arbor, Mich.: Karoma.

———. 1995. *Language and human behavior*. Seattle: University of Washington Press.

Bisiach, E., and C. Luzatti. 1978. Unilateral neglect of representational space. *Cortex* 14: 129–133.

Black, D. L. 1998. Splicing in the inner ear: A familiar tune, but what are the instruments? *Neuron* 20: 165–168.

Blakemore, C., and G. F. Cooper. 1970. Development of the brain depends on the visual environment. *Nature* 228: 477–478.

Blakeslee, S. 2000. Rewired ferrets contradict popular theories of brain's growth. *New York Times*, April 25, F1.

Bloom, P. 2000. *How children learn the meanings of words*. Cambridge, Mass.: MIT Press.

Blum, A. 1994. New approximation algorithms for graph coloring. *Journal of the ACM* 41 (3): 470–516.

Blum, A., and M. Furst. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90: 281.

Blumer, A., A. Ehrenfeucht, D. Haussler, and M. Warmuth. 1989. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM* 36: 929–965.

Branden, C., and J. Tooze. 1999. *Introduction to protein structure*. New York: Garland.

Carroll, S. B., J. K. Grenier, and S. D. Weatherbee. 2001. *From DNA to diversity: Molecular genetics and the evolution of animal design*. Malden, Mass.: Blackwell Science.

Caruana, R., S. Lawrence, and C. L. Giles. 2001. Overfitting in neural networks: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems 13*, ed. T. K. Leen, T. G. Dietterich, and V. Tresp, 402–408. Cambridge, Mass.: MIT Press.

Center for Wildlife Information. 2003. Bears of North America. ⟨www.bebearaware.org/bearstocomparenfnf.htm⟩.

Chalmers, D. 1996. *The conscious mind*. New York: Oxford University Press.

Chenn, A., and C. A. Walsh. 2002. Regulation of cerebral cortical size by control of cell cycle exit in neural precursors. *Science* 297: 365–369.

Chomsky, N. 1965. *Aspects of the Theory of Syntax*. Cambridge, Mass.: MIT Press.

Coleman, S., and J. Mandula. 1967. All possible symmetries of the *S* matrix. *Physical Review* 159: 1251.

Conway, J. H. 1976. *On numbers and games*. New York: Academic Press.

Cook, S. A. 1971. The complexity of theorem-proving procedures. In *Proceedings of the 3d Annual ACM Symposium on Theory of Computing*, 151–158.

Copeland, J. 1993. *Artificial intelligence: A philosophical introduction*. Oxford: Blackwell.

Cormen, T. H., C. E. Leiserson, and R. L. Rivest. 1990. *Introduction to algorithms*. Cambridge, Mass.: MIT Press.

Cover, T. M., and J. A. Thomas. 1991. *Elements of information theory*. New York: Wiley.

Crick, F. 1994. *The astonishing hypothesis*. New York: Scribners.

Crick, F., and C. Koch. 1998. Consciousness and neuroscience. *Cerebral Cortex* 8: 97–107.

———. 2000. The unconscious homunculus. In *Neural correlates of consciousness*, ed. T. Metzinger, 103–110.

Crick, F., and G. Mitchison. 1983. The function of dream sleep. *Nature* 304: 111–114.

Damasio, A. R. 1994. *Descartes' error: Emotion, reason, and the human brain*. New York: Putnam.

———. 1999. *The Feeling of what happens: Body and emotion in the making of consciousness*. New York: Harcourt Brace.

———. 2000. A neurobiology for consciousness. In *Neural correlates of consciousness*, ed. T. Metzinger, 111–120.

Davidson, E. H., et al. 2002. A genomic regulatory network for development. *Science* 295: 1669–1678.

Dawkins, R. 1989. *The selfish gene*. New York: Oxford University Press.

De Souza, S. J., et al. 1998. Toward a resolution of the introns early/late debate: Only phase zero introns are correlated with the structure of ancient proteins. *Proceedings of the National Academy of Sciences USA* 95: 5094–5099.

Deacon, T. W. 1997. *The symbolic species*. New York: Norton.

Dennett, D. C. 1988. *The intentional stance*. Cambridge, Mass.: MIT Press.

———. 1991. *Consciousness explained*. Boston: Little, Brown.

Devlin, K. 2000. *The math gene*. New York: Basic Books.

Diligenti, M., F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. 2000. Focused crawling using context graphs. In *26th International Conference on Very Large Databases, VLDB 2000*, 527–534.

Donald, M. 1991. *Origins of the modern mind*. Cambridge, Mass.: Harvard University Press.

Dorn, B. H. 2003. Katmai National Park editorial. ⟨www.brucehamiltondorn.com/pages/b_muse_02.html⟩.

Drescher, G. 1991. *Made-up minds*. Cambridge, Mass.: MIT Press.

Dretske, F. I. 1981. *Knowledge and the flow of information*. Cambridge, Mass.: MIT Press.

Dreyfus, H. L. 1972. *What computers can't do*. Cambridge, Mass.: MIT Press.

———. 1993. *What computers still can't do: a critique of artificial reason*. Cambridge, Mass.: MIT Press.

Ehrenfeucht, A., D. Haussler, M. Kearns, and L. Valiant. 1989. A general lower bound on the number of examples needed for learning. *Information and Computation* 82 (3): 247–261.

Ellington, A. D., and J. W. Szostak. 1990. In vitro selection of RNA molecules that bind specific ligands. *Nature* 346: 818–822.

Erol, K., D. S. Nau, and V. S. Subrahmanian. 1992. On the complexity of domain-independent planning. In *Proceedings of the AAAI National Conference*, 381–386.

Esch, H. E., S. Zhang, M. V. Srinivasan, and J. Tautz. 2001. Honeybee dances communicate distances measured by optic flow. *Nature* 411: 581–583.

Feist, M. 2002, Interview with Mathias Feist, ⟨http://www.karlonline.org/402_2.htm⟩.

Fodor, J. A. 1987. *Psychosemantics*. Cambridge, Mass.: MIT Press.

Fong, S. 1999. Parallel principle-based parsing. In *Sixth International Workshop on Natural Language Understanding and Logic Programming, Interational Conference on Logic Programming, Las Cruces, New Mexico*.

Fong, S., and R. C. Berwick. 1985. New approaches to parsing conjunctions using Prolog. In *International Joint Conference on Artificial Intelligence, August 1985, Los Angeles* (870–876). San Mateo: Morgan Kaufmann.

Franklin, S. 1995. *Artificial minds*. Cambridge, Mass.: MIT Press.

Freud, S. 1900. *The interpretation of dreams*, trans. A. A. Brill. New York: Random House, 1950.

Fukushima, K. 1979. Neural network model for a mechanism of pattern recognition unaffected by shift in position—neocognitron. *Transactions of the IECE Japan* 62-A (10): 658–665.

Garey, M. R., and D. S. Johnson. 1979. *Computers and intractability: A guide to the theory of NP-completeness*. New York: Freeman.

Garey, M. R., D. S. Johnson, and L. Stockmeyer. 1978. Some simplified NP-complete graph problems. *Theoretical Computer Science* 1: 237–267.

Gazzaniga, M. S. 1998. *The mind's past*. Berkeley: University of California Press.

Gibson, E., and K. Wexler. 1994. Triggers. *Linguistic Inquiry* 25 (3): 407–454.

Gilbert, W., S. J. De Souza, and M. Long. 1997. Origin of genes. *Proceedings of the National Academy of Sciences USA* 94: 7698–7703.

Giurfa, M., S. Zhang, A. Jenett, R. Menzel, and M. V. Srinivasan. 2001. The concepts of ''sameness'' and ''difference'' in an insect. *Nature* 410: 930–933.

Gould, J. L., and C. G. Gould. 1994. *The animal mind*. New York: Scientific American Library.

Gur, C. R., and H. A. Sackheim. 1979. Self-deception: A concept in search of a phenomenon. *Journal of Personality and Social Psychology* 37: 147–169.

Harman, G. 1973. *Thought*. Princeton, N.J.: Princeton University Press.

Hartley, D. 1791. *Observations on man, his frame, his duty and his expectations*. London: Johnson.

Hauser, M. D. 2000. *Wild minds*. New York: Henry Holt.

Hauser, M. D., D. Weiss, and G. Marcus. 2002. Rule learning by cotton-top tamarins. *Cognition* 86: B15–B22.

Haussler, D. 1988. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence* 36 (2): 177–221.

Hazlett, T. W. 1997. Looking for results: An interview with Ronald Coase. *Reason*, January. ⟨http://reason.com/9701/int.coase.shtml⟩.

Hillis, W. D. 1990. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D* 42: 228–234.

Hinton, G., and S. Nowlan. 1987. How learning can guide evolution. *Complex Systems* 1: 495–502.

Holland, J. H. 1986. Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In *Machine Learning*, vol. 2, ed. R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, 593. Los Altos, Calif.: Morgan Kauffman.

Hong, W., and J. Slotine. 1995. Experiments in hand-eye coordination using active vision. In *Proceedings of the 4th International Symposium on Experimental Robotics, ISER'95*.

Hopfield, J. J. 1982. Neural networks and physical systems with collective computational abilities. *Proceedings of National Academy of Sciences USA* 79: 2554–2558.

Hopfield, J. J., D. I. Feinstein, and R. G. Palmer. 1983. "Unlearning" has a stabilizing effect in collective memories. *Nature* 304: 158–159.

Hubel, D., and T. Wiesel. 1962. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physics* 160: 106–154.

Hutchinson, A. 1994. *Algorithmic learning*. Oxford: Clarendon Press.

Jackson, F. C. 1982. Epiphenomenal qualia. *Philosophical Quarterly* 32: 127–136.

Kandel, E. R. 2001. The molecular biology of memory storage: A dialogue between genes and synapses. *Science* 294: 1030–1038.

Karp, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations*, ed. R. E. Miller and J. W. Thatcher, 85–103. New York: Plenum Press.

Kauffman, S. A. 1993. *The origins of order*. New York: Oxford University Press.

Keefe, A. D., and J. W. Szostak. 2001. Functional proteins from a random sequence library. *Nature* 410: 715.

Khardon, R., and D. Roth. 1994. Learning to reason. In *Proceedings of the 12th National Conference on Artificial Intelligence*, 682–687.

Koehler, J. 1998. Solving complex planning tasks through extraction of subproblems. In *Proceedings of the 4th International Conference on Artificial Intelligence Planning Systems*, 62–69.

Kolers, P. A., and M. von Grünau. 1976. Shape and color in apparent motion. *Vision Research* 16: 329–335.

Kotov, A. 1971. *Think like a grandmaster*. London: Batsford Books.

Koza, J. 1992. *Genetic programming*. Cambridge, Mass.: MIT Press.

Kurzweil, R. 1999. *The age of spiritual machines: When computers exceed human intelligence*. New York: Viking.

Lakatos, I. 1977. *Proofs and refutations*. Cambridge: Cambridge University Press.

Lakoff, G. 1996. *Moral politics: What conservatives know that liberals don't*. Chicago: University of Chicago Press.

Lakoff, G., and M. Johnson. 1980. *Metaphors we live by*. Chicago: University of Chicago Press.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86 (11): 2278–2324.

Lenat, D. B. 1985. EURISKO: A program that learns new heuristics and domain concepts. *Artificial Intelligence* 21: 61–98.

———. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38: 11.

Leopold, D. A., and N. K. Logothetis. 1996. Activity changes in early visual cortex reflect monkeys' percepts during binocular rivalry. *Nature* 384: 549–553.

Lin, S., and B. W. Kernighan. 1973. An effective heuristic algorithm for the traveling salesman problem. *Operations Research* 21: 498–516.

Logothetis, N., and J. Schall. 1989. Neuronal correlates of subjective visual perception. *Science* 245: 761–763.

MacKay, D.J.C. 1992. Bayesian interpolation. *Neural Computation* 4: 415–472.

———. 1999. Rate of information acquisition by a species subjected to natural selection. ⟨http://www.inference.phy.cam.ac.uk/mackay/abstracts/gene.html⟩.

Maes, P. 1990. How to do the right thing. *Connection Science* 1 (3): 291–323.

Makalowski, W. 2000. Genomic scrap yard: How genomes utilize all that junk. *Gene* 259: 61–67.

Maquet, P. 2001. The role of sleep in learning and memory. *Science* 294: 1048–1052.

Mattick, J. S., and M. J. Gagen. 2001. The evolution of controlled multitasked gene networks: The role of introns and other noncoding RNAs in the development of complex organisms. *Molecular Biology and Evolution* 18 (9): 1611–1630.

Maynard Smith, J. 1978. *The evolution of sex*. Cambridge: Cambridge University Press.

Maynard Smith, J., and E. Szathmary. 1999. *The origins of life*. Oxford: Oxford University Press.

McCulloch, W. S., and W. H. Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5: 115–133.

McDermott, D. V. 1981. Artificial intelligence meets natural stupidity. In *Mind design*, ed. J. Haugeland, 133. Cambridge, Mass.: MIT Press.

———. 1995. [STAR] Penrose is wrong. *Psyche* 2 (17). ⟨http://psyche.cs.monash.edu.au/v2/pyche-2-17-mcdermott.html⟩.

———. 2001. *Mind and mechanism*. Cambridge, Mass.: MIT Press.

McGurk, H., and J. MacDonald. 1976. Hearing lips and seeing voices. *Nature* 264: 746–748.

Melchner, L. V., S. L. Pallas, and M. Sur. 2000. Visual behaviour mediated by retinal projections directed to the auditory pathway. *Nature* 404: 871–876.

Merkle, R. C. 1989. Energy limits to the computational power of the human brain. *Foresight Update* 6. ⟨http://www.merkle.com/merkledir/papers.html⟩.

Metzinger, T., ed. 2000. *Neural correlates of consciousness*. Cambridge, Mass.: MIT Press.

Michal, G., ed. 1998. *Biochemical pathways: An atlas of biochemistry and molecular biology*. New York: Wiley. Wall charts of biochemical pathways available at ⟨http://biochem.boehringer_mannheim.com/fst/products.htm?/techserv/metmap/htm⟩.

Miller, G. A. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review* 63: 81–97.

Miller, M. S., and K. E. Drexler. 1988a. Comparative ecology: A computational perspective. In *The ecology of computation*, ed. B. A. Huberman, 51. New York: Elsevier.

———. 1988b. Markets and computation: Agoric open systems. In *The ecology of computation*, ed. B. A. Huberman, 133.

Millikan, R. 1984. *Language, thought, and other biological categories*. Cambridge, Mass.: MIT Press.

Milner, D., and M. Goodale. 1995. *The visual brain in action*. Oxford: Oxford University Press.

Mineka, S., R. Keir, and V. Price. 1980. Fear of snakes in wild- and laboratory-reared rhesus monkeys. *Animal Learning and Behavior* 8: 653–663.

Minsky, M. 1967. *Computation: Finite and infinite machines*. Englewood Cliffs, N.J.: Prentice-Hall.

———. 1986. *The society of mind*. New York: Simon and Schuster.

Mittler, J. E., M. Markowitz, D. D. Ho, and A. S. Perelson. 1999. Improved estimates for HIV-1 clearance rate and intracellular delay. *AIDS* 13 (11): 1415–1417.

Montana, D. J. 1995. Strongly typed genetic programming. *Evolutionary Computation* 3(2): 199–230.

Myhrvold, N. 1994. Roadkill on the information highway. Distinguished Lecture Series: Industry Leaders in Computer Science and Electrical Engineering. Stanford, Calif.: University Video Communications.

Nelson, R. R., and S. G. Winter. 1982. *An evolutionary theory of economic change*. Cambridge, Mass.: Harvard University Press.

Nesse, R. M., and G. C. Williams. 1994. *Why we get sick: The new science of Darwinian medicine*. New York: Vintage Books.

Newell, A. 1990. *Unified theories of cognition*. Cambridge, Mass.: Harvard University Press.

Nolfi, S., and D. Floreano. 2000. *Evolutionary robotics.* Cambridge, Mass.: MIT Press.

Nowak, M. A., and D. C. Krakauer. 1999. The evolution of language. *Proceedings of the National Academy of Sciences USA* 96: 8028–8033.

Penrose, R. 1989. *The emperor's new mind.* New York: Oxford University Press.

———. 1994. *Shadows of the mind.* New York: Oxford University Press.

Perelson, A. S., A. Neumann, M. Markowitz, J. Leonard, and D. D. Ho. 1996. HIV-1 dynamics in vivo: Virion clearance rate, infected cell life-span, and viral generation time. *Science* 271: 1582–1586.

Pfeifer, R., C. Scheier, and Y. Kuniyoshi. 1998. Embedded neural networks: Exploiting constraints. *Neural Networks* 11: 1551–1569.

Pinker, S. 1994. *The language instinct*. New York: Morrow.

Pirsig, R. M. 1984. *Zen and the art of motorcycle maintenance: An inquiry into values*. Bantam Books.

Pollan, M. 2002. *The botany of desire: A plant's-eye view of the world*. New York: Random House.

Post, E. L. 1943. Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics* 52: 264–268.

Putnam, H. 1995. Review of Penrose's *Shadows of the Mind. Bulletin of the American Mathematical Society* 32 (3): 370–373.

Quartz, S., and T. J. Sejnowski. 1994. Beyond modularity: Neural evidence for constructivist principles in development. *Behavioral and Brain Sciences* 17: 725–727.

———. 1997. The neural basis of cognitive development: A constructivist manifesto. *Behavioral and Brain Sciences* 20 (4): 537–596.

Quine, W. V. 1960. *Word and object*. Cambridge, Mass.: MIT Press.

Ramachandran, V. S. 1995. Anosognosia in parietal lobe syndrome. *Consciousness and Cognition* 4: 22–51.

Ramachandran, V. S., and S. Blakeslee. 1998. *Phantoms in the brain: Probing the mysteries of the human mind*. Morrow.

Read, L. 1958. I, pencil: My family tree as told to Leonard E. Read. *Freeman*, December. ⟨http://www.econlib.org/library/essays/rdpncl1.html⟩.

Refenes, A.-P., ed. 1994. *Neural networks in the capital market*. New York: Wiley.

Ridley, M. 1994. *The Red Queen: Sex and the evolution of human nature*. New York: Macmillan.

———. 1996. *The origins of virtue*. New York: Penguin.

———. 1999. *Genome: The autobiography of a species in 23 chapters.* New York: HarperCollins.

Rissanen, J. 1989. *Stochastic complexity in statistical inquiry*. Singapore: World Scientific.

Robertson, D. S. 1999. Algorithmic information theory, free will, and the Turing test. *Complexity* 4 (3): 25–34.

Roth, G. 2000. The evolution and ontogeny of consciousness. In *Neural correlates of consciousness*, ed. T. Metzinger, 77–97.

Rumelhart, D., G. E. Hinton, and R. J. Williams. 1986. Learning internal representations by error propagation. In *Parallel distributed processing*, ed. D. Rumelhart and J. McClelland. Cambridge, Mass.: MIT Press.

Russell, S., and P. Norvig. 1995. *Artificial intelligence: A modern approach*. Upper Saddle River, N.J.: Prentice-Hall.

Sandberg, R., et al. 2000. Regional and strain-specific gene expression mapping in the adult mouse brain. *Proceedings of the National Academy of Sciences USA* 97 (20): 11038–11043.

Scheinberg, D. L., and N. K. Logothetis. 1997. The role of temporal cortical areas in perceptual organization. *Proceedings of the National Academy of Sciences USA* 94: 3408–3413.

Schell, T., A. E. Kulozik, and M. W. Hentze. 2002. Integration of splicing, transport, and translation to achieve mRNA quality control by the nonsense-mediated decay pathway. *Genome Biology* 3 (3): 1006.1–1006.6.

Schrödinger, E. 1944. *What is life?* Cambridge: Cambridge University Press.

Searle, J. R. 1990. Is the brain's mind a computer program? *Scientific American* 262 (1): 26–31.

———. 1997. *The mystery of consciousness*. New York: New York Review of Books.

———. 2001. Free will as a problem in neurobiology. ⟨http://www.theunityofknowledge.org/the_free_will_fiction/searle.htm⟩.

Seay, B. M., and H. F. Harlow. 1965. Maternal separation in the rhesus monkey. *Journal of Nervous and Mental Disease* 140: 434–441.

Selfridge, O. 1959. Pandemonium: A paradigm for learning. In *Proceedings of the Symposium on Mechanisation of Thought Process*. National Physics Laboratory.

Shannon, C. E. 1956. A universal Turing machine with two internal states. In *Automata Studies*, ed. C. E. Shannon and J. McCarthy, 157–165. Princeton, N.J.: Princeton University Press.

Shoemaker, D. D., and P. S. Linsley. 2002. Recent developments in DNA microarrays. *Current Opinion in Microbiology* 5 (3): 334–337.

Shor, P. W. 1994. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, 124–134.

Siegel, J. M. 2001. The REM sleep–memory consolidation hypothesis. *Science* 294: 1058–1063.

Siegelmann, H. T. 1995. Computation beyond the Turing limit. *Science* 268: 545–548.

Siskind, J. M. 1994. Lexical acquisition in the presence of noise and homonymy. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 760–766.

Skyrms, B. 1996. *Evolution of the social contract.* New York: Cambridge University Press.

Slate, D. J., and L. R. Atkin. 1983. Chess 4.5: The Northwestern University chess program. In *Chess skill in man and machine*, ed. P. Frey.

Smith, B. C. 1996. *On the origin of objects*. Cambridge, Mass.: MIT Press.

Smith, W. D. 1993. Church's thesis meets the N-body problem. NEC Research Institute Technical Report. ⟨http://www.neci.nj.nec.com/homepages/wds/works.html⟩.

———. 1999. Church's thesis meets quantum mechanics. NEC Research Institute Technical Report. ⟨http://www.neci.nj.nec.com/homepages/wds/works.html⟩.

Smolin, L. 1997. *The life of the cosmos*. Oxford: Oxford University Press.

Sokolowski, R. 2000. *Introduction to phenomenology*. Cambridge: Cambridge University Press.

Stickgold, R., J. A. Hobson, R. Fosse, and M. Fosse. 2001. Sleep, learning, and dreams: Off-line memory reprocessing. *Science* 294: 1052–1057.

Stock, J., and M. Levit. 2000. Signal transduction: Hair brains in bacterial chemotaxis. *Current Biology* 10: R11–R14.

Sugihara, K. 1982. Mathematical structures of line drawings of polyhedrons: Toward man-machine communication by means of line drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-4 (5): 458–469.

Surette, M. G., M. B. Miller, and B. L. Bassler. 1999. Quorum sensing in *Escherichia coli, Salmonella typhimurium*, and *Vibrio harveyi*: A new family of genes responsible for autoinducer production. *Proceedings of the National Academy of Sciences USA* 96: 1639–1644.

Sutton, R. S., and A. G. Barto. 1998. *Reinforcement learning: An introduction*. Cambridge, Mass.: MIT Press.

Tesar, B., and P. Smolensky. 2000. *Learnability in optimality theory*. Cambridge, Mass.: MIT Press.

Tesauro, G. 1995. Temporal difference learning and TD-gammon. *Communications of the ACM* 38 (3): 58.

Theron, C. 2001. An online encounter with Christophe Theron, ⟨http://www.computerschach.de/sprechstunde/archiv/theron1e.htm⟩.

Toga, A., and J. Mazziotta. 1996. *Brain mapping: The methods*. San Diego: Academic Press.

Trivers, R. 1991. Deceit and self-deception: The relationship between communication and consciousness. In *Man and beast revisited*, ed. M. Robinson and L. Tiger, 175–191. Washington D.C.: Smithsonian Institution Press.

Trueswell, J. C., M. K. Tanenhaus, and S. M. Garnsey. 1994. Semantic influences on parsing: Use of thematic role information in syntactic ambiguity resolution. *Journal of Memory and Language* 33: 285–318.

Turing, A. M. 1937. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society* (ser. 2) 42: 230–265. ⟨http://www.abelard.org/turpap2/turpap2.htm⟩.

———. 1950. Computing machinery and intelligence. *MIND* 59 (236): 433–460. ⟨http://www.abelard.org/turpap/turpap.htm⟩.

Valiant, L. G. 1984. A theory of the learnable. *Communications of the ACM* 27 (11): 1134–1142.

———. 1994. *Circuits of the mind*. New York: Oxford University Press.

———. 1995. Rationality. In *Proceedings of the 8th Annual Conference on Computational Learning Theory*, 3–14.

Vapnik, V. 1995. *The nature of statistical learning theory*. New York: Springer-Verlag.

Varian, H. P. 1996. *Intermediate microeconomics: A modern approach*. New York: W.W. Norton.

Vergis, A., K. Steiglitz, and B. Dickinson. 1986. The complexity of analog computation. *Math and Computers in Simulation* 28: 91–113.

Visick, K., and M. J. McFall-Ngai. 2000. An exclusive contract: Specificity in the *Vibrio fischeri–Euprymna scolopes* partnership. *Journal of Bacteriology* 182: 1779–1787.

von Neumann, J. 1956. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In *Automata Studies*, ed. C. E. Shannon and J. McCarthy, 329–378. Princeton, N.J.: Princeton University Press.

———. 1966. *Theory of self-reproducing automata*. Urbana: University of Illinois Press.

Waltz, D. 1975. Understanding line drawings of scenes with shadows. In *The psychology of computer vision*, ed. P. H. Winston. New York: McGraw-Hill.

Warland, D., P. Reinagel, and M. Meister. 1997. Decoding visual information from a population of retinal ganglion cells. *Journal of Neurophysiology* 78: 2336–2350.

Wexler, K. 1999. Acquisition of syntax. In *MIT Encyclopedia of the Cognitive Sciences*. Cambridge, Mass.: MIT Press.

Wiener, J. 1994. *The beak of the finch: A story of evolution in our time*. New York: Knopf.

Wilcox, B. 1977. Instant Go [part 1]. *American Go Journal* 12 (5). American Go Association.

———. 1979. Instant Go [part 2]. *American Go Journal* 14 (5/6). American Go Association.

———. 1985. Reflections on building two Go programs. *SIGART Newsletter* 94: 29–43.

Wilczek, F. 1994. A call for a new physics. *Science* 266: 1737–1738.

Wilkins, D. 1980. Using patterns and plans in chess. *Artificial Intelligence* 14: 165–203.

Wilson, E. O. 1978. *On Human Nature*. Cambridge, Mass.: Harvard University Press.

———. 1998. *Consilience: The unity of knowledge*. New York: Vintage.

Wilson, S. 1994. ZCS: a zeroth level classifier system. *Evolutionary Computation* 2 (1): 1–18.

Wittgenstein, L. 1953. *Philosophical investigations*. Macmillan: London.

Young, H. P. 1998. *Individual strategy and social structure: An evolutionary theory of institutions*. Princeton, N.J.: Princeton University Press.

Zhao, X., et al. 2001. Transcriptional profiling reveals strict boundaries between hippocampal subregions. *Journal of Comparative Neurology* 441: 187–196.

Zhu, J., et al. 2002. Quorum-sensing regulators control virulence gene expression in *Vibrio cholerae. Proceedings of the National Academy of Sciences USA* 99 (5): 3129–3134.