Abstract

In order to understand AI, we should understand computation. In order to establish bounds on what we should count as computation, I will describe three kinds of machines that we program.

The first is the classic numerical calculator which will take us from tables of logarithms to Turing to Texas Instruments.

The second is the modern computing system, which includes distributed systems and concurrent programs, web-based infrastructure, wearable devices, unmanned combat air vehicles, sensor networks, virtual reality environments, and their forebears, the programmed controllers such as operating systems and thermostats.

The third is a class of machine that is related to formal games such as chess and bridge, namely human society and its institutions. My conclusion is that AI can emerge from the rule-governed coordination of individuals in a society, not just from well-programmed and well-situated machines. In effect, we have been making ourselves into the machine. Whether this is shocking, alarming, or disturbing depends on whether the machine is making us better or making us worse.

Introduction

For several years, whenever people have asked me how I feel about a future with artificial intelligence, I have managed to surprise them with my answer. I have told them that AI has already been in our lives for a long time, even before electronic computers. It's not a common answer.

It is certainly a common question: What will the world be like when software systems make decisions for us? What will the world be like when it contains devices that are autonomous? What can it mean for us to spend time in virtual worlds cohabited by programs? People are always asking this question or some variation of it. Since my answer is not a common answer, I am happy to have the opportunity to explain it at length. My answer in short is that bureaucracy is more frightening than AI, and they are two of the same kind. Moreover, we have been living with bureaucracies, and rule-governed symbol-processing human institutions generally, for quite a long time.

With minor misgivings, I think we can agree that the popular conception of AI as it has appeared in several decades of sci-fi films is a fair depiction of where AI claims it is headed.

HAL from *2001* and MOTHER from *Alien* do nothing more exotic than help humans control their complex space craft, although both have marvelous natural language abilities and can take a lot of initiative. HAL is proactive, not just intentional. *Star Trek*'s original series MAIN COMPUTER is excellent at statistical inference as well, with degrees of precision that boggle the mind of the careful inductivist.

The later *Star Trek*'s neural net-endowed android, DATA, is indeed a worthy objective for engineers who want to look forward a century or two. Who would have thought, when Jean-Luc Picard took the starship Enterprise's deck in the revived TV show's first season, that the computer's ability to access hundreds of years of trivial human records would foreshadow the internet search engines we consider already mundane today. DATA seems just too good to be possible, so we have *Star Wars*' R2D2 and C3PO as consolation prizes if DATA turns out to be too hard to build. Certainly the automatic targeting assistance that a droid co-pilot provides to an X-wing fighter pilot who has to deliver his torpedo into a Death Star vent is not going to seem outlandish to many of the Boeing engineers who work ten minutes from here on highway I-70.

It is true that AGENT SMITH wants to do bad things to humans, or at least to do bad things to virtual humans, in the *Matrix*. Sometimes science fiction needs monsters, and often it has used up all its big gorillas and Japanese moths and reincarnated raptors (or at least they are reviscerated and rehydrated, if not reincarnated). Many of us who grew up with R2D2 in mind reacted with a mix of glee and horror when the insect-like robot sentinels attack the rebel Morpheus's hover craft, Nebuchadnezzar, and start to twitch with a dreadful entomological obsessive compulsive disorder. "That's what they are trying to build at MIT," we could say to ourselves with a self-assured repugnance. "See what happens when you fund the research of people who want to build automated insects?"

The *Matrix* does not presuppose a very happy future real world, so why should it also suppose AI programs to be friendly in the virtual world? AGENT SMITH may want to punch good people hard, but apparently some other program is responsible for the infinitely more despicable infant harvesting, pod immersion, and subterranean seclusion that afflicts the human race in this movie's vision of the future. Is it plausible as AI? I like to tell the students that I am certain there are lots of day traders on Wall Street who have run into their own version of AGENT SMITH: some adversarial automatic trading program which has had the intention of kicking them out of the system, punching them hard, human trading outmatched by sheer speed and cold calculation, your account balance zeroed, game over. The existence

of an AGENT SMITH in a financial virtual world does not mean that AI is taking over our lives. It does mean that a company like Wall Street Analytics or D. E. Shaw can provide a lot of profit to its customers and hire a lot of programmers and Ph.D.'s instead of the Ivy league ex-athlete pre-M.B.A. brokers to be found at Goldman Sachs and Morgan Stanley.

In the real world, AI today finds its form in autopilots, data mining and information retrieval such as credit card fraud detection and google hit ranking, in intelligent consumer products such as smart Minolta autofocus cameras and Canon scanners, smart traction control Mercedes transmissions and Lexus active suspensions; also it shows up in smart book recommendations from Amazon.com, smart vacuum cleaners from Sharper Image, smart electronic pets from Sony, and smart voice recognition systems when you try to contact companies like Southwestern Bell that don't really want to talk to you. The more complex AI remains in virtual worlds such as electronic interactive games, databases, and ecommerce where detachment from the physical world makes more automated decision making more possible in more quantity. And of course there is complex AI in the research laboratory, where programs that can drive cars at 60 mph can only do so for short periods in perfect conditions, and programs can recognize faces in a crowd only with generous rates of false-positive error, and programs can write their own legal briefs on one or two well-understood and highly routine subjects.

The most impressive AI is in nuclear power plants, national security agencies, large banks, and hospitals. In these places, the AI is impressive not because it makes use of a more advanced technology, but because the software is allowed to advise or even to automate decision making that could be disastrous if it were ever to go badly wrong. Ironically, the AI is usually allowed to support the control of these important systems simply because these systems could not possibly be controlled without the software's help. The Wachowski brothers who wrote the *Matrix* made this point when the mayor of the last human city, Zion, takes Neo, the messianic protagonist, under the city to see the machines that make life underground possible. Without the (good) machines to control the city, the humans could not protect themselves against the siege of the city by the (bad) machines. Here, Neo has his epiphany, that being at war with the machines is futile. Too bad the script staggers to its demise after this lovely irony is revealed.

What makes a program an AI program is usually the fact that it automates something that humans do or that is thought to require intelligence to do. It may be AI on other grounds. Technically, any program that uses an AI technique such as heuristic search or genetic programming will be considered AI, even if it does not actually automate the decision making or the control

that would normally be provided by a human. AI technique on a non-AI problem is one kind of AI. But most of the time, AI techniques are employed to address AI problems. In fact, more often these days, a generic software solution, not an AI technique, is used to automate a task. That is, most AI these days is non-AI technique applied to an AI problem. Many systems are AI because of the problems they solve and the roles they play in our lives, not because of the techniques that they use to play those roles. There is nothing particularly interesting about the software in a thermostat. But no one doubts that the thermostat is doing AI, albeit a particularly mundane and degenerate kind of AI. Some people see a bright line dividing software solutions that employ machine learning and those that do not. Programs that "learn" seem more autonomous than programs that follow static rules. But programming how a program's behavior will evolve is just meta-programming. Once you write one of those "self-modifying" programs, you realize that the ability to learn is not profound. A program with one stored quantity, 0, which changes that quantity as the world passes it by, flipping between 0 and 1, is, after all learning. Big deal.

Mostly what makes AI is the careful embedding of a computational device in the world so that its computations have meanings to the rest of us. If we take the fuel-mixture oxygen sensor out of my 1992 Alfa Romeo and put it in my office aquarium, it has no intelligence. It has lost its proper embedding. It is no longer well situated. The symbol processing that it performs in that underwater setting might as well be random, as its logic has no more meaning to me than to the fish once it is detached from its air intake plenum. Being plugged-in is an important part of artificial intelligent behavior. Animals seem intelligent in this world, but there are worlds in which they wouldn't seem very intelligent because they couldn't plug in very well. Swapping places with dolphins comes to mind.

Daniel Dennett makes some wonderful observations about the "sphexishness" of a particular wasp, the sphex wasp, which seems to have robust routines for searching its environment, and finding little holes, and escaping traps, until we engineer the environment to have concave lips that it cannot get past. I have one of those wasp-collecting plastic jars with the inward-blooming floral surface, designed to eliminate a whole population of wasps within a week, and I feel a bit guilty about opening it at the end of the summer and seeing row after row of wasp bodies, lined up like cigarettes in a carton, doomed from the moment they entered the jar, doomed to wander around and around inside until they exhaust themselves simply because their programming presupposed a world without Klein bottles and they could never think to turn left after turning up. It is my intelligence that permits me to place these things in my environment. But I have limits too. My own body's pro-

grams, I confess, have trouble with daylight savings time, tax rule modifications, cellular phone user interfaces, new releases of the Windows operating system, and whatever it is that the BBC thinks it is doing to the English language.

Thus, most of the AI people you talk to will tell you that AI is not frightening because AI is very sphexish. AI programs depend crucially on being in the right place in order to do their jobs. Turn the ROOMBA vacuum cleaner over, and it is not programmed to right itself. At least a turtle and a beetle will try. Place it on a wet surface and it might destroy itself, until ROOMBA 2.0 is released which undoubtedly will feature a wet vac. We say that systems are "brittle" with respect to assumptions about the problem "domain." In order to represent the environment so that rule-based symbol-processing can be meaningful, we have to assume the environment is something very restricted and specific, like a 4-foot square flat patch of grass, or a perfectly illuminated table with known objects arranged in a line, or a corpus of documents written by judges and typed by legal secretaries, marked flawlessly with the tags of legal XML.

Or else we have to assume that our interest in the environment is very particular, like the narrow and single-minded interest in percent humidity or temperature or the degree to which a punctured card looks like a vote for Al Gore instead of for George Bush or Pat Buchanan. General purpose, robust, self-reflective, self-modifying, and self-conscious intelligence is so far away that we might as well be discussing how we would feel about the next ice age. As the *Blade Runner* fans like to remind us, there are more frightening technological possibilities emerging in the other sciences which may present disturbing mirrors to ourselves, and these sciences are producing other technologies that are on a much nearer horizon. RACHEL in *Blade Runner* is the product of cloning, not AI. Worrying about self-conscious AI, in a networked world where Paypal account theft and international money-laundering is our real problem, is like worrying that the sun is exhausting itself in a world where a ground-impact asteroid tomorrow could make the explosion of the ancient island kingdom Thera look like a happy day in Minoa.

Most technical people who participate in AI would come to you today, to this conference on theological encounters with AI and virtual reality, and they would say that you have nothing to worry about. AI isn't going to happen in a shocking way, change will be gradual, and technological advance will hardly be noticed by those who live through the revolutions, like the arrival of the web, and the refrigerator and the reading glasses and the agricultural plow. About AI we have no worry. It won't take away any job except a job

that was beneath you anyway, and it won't kill you in your sleep. Be happy.

This is not what I have to tell you today. I think AI is shocking, and it provides us with knowledge of ourselves that must cause us to rethink ourselves fundamentally. I think AI makes us look at our rule-governed lives in a different way. It tells us what is precious about us and what we should strive to preserve and protect. And it tells us something about what kind of world we have been making for ourselves for the past several thousand years.

Two Kinds of Computation

In order to proceed, I must explain what I consider to be a program, and what I consider to be a computation. At the end of this diversion, I want to say that society's rules are a kind of program, and that our living the lives we do constitutes a computation. I don't just want to draw an analogy. I do not want people to say that the social computation is a lot like an electronic computation. I want people to realize that it is a computation, as much as anything else we are doing today with computers is a computation. People often want to say that we are programmed by evolution to have certain linguistic or perceptual biases. But they don't mean "programmed" as seriously as I mean it when I say that legislation is a kind of programming, and social interaction is a kind of computation, and social institutions are a kind of AI.

There is a folk understanding of computation which surrounds the use of the word, "computation." Especially when used pejoratively, the connotations of computation are numerous. "One computes the answer" because one is inexorably pushed from start to finish. Computation seems even more automatic than calculation. At least "One calculates the answer" suggests that one could have calculated otherwise, with a different calculus as if one were a polymath, which would entail the possibility of a different calculation in a different calculus. A machine and I could use exactly the same calculus to arrive at an answer, yet most will say that I calculated while the machine computed. If my answer is wrong, my rule-following is suspect. If the machine's answer is wrong, we wonder if the rules themselves are wrong, or if the machine is unwell. There is certainly a flawlessness and slavishness attributed to things that compute.

The classic computation is the calculation of a mathematical function, such as the logarithm. One imagines computers to be larger or faster versions of the calculator. Perhaps the modern computer in the folk mind processes alphabetical symbols, not just numbers. But we all think of the modern computer as essentially a big Texas Instruments chip with Intel stamped on it instead of TI. Even fundamental conceptions of programs and algorithms with-

in the field of computer science attempt to reduce all computer programs to some kind of mathematical function. What matters is the proper relation between the input and the output, nothing else. This is, at least, a century-old way of thinking about computing.

There are other very old biases about computation: A computation is repeatable. It is deterministic. It always produces the same answer when repeated, and it permits no elective choices nor chance intrusions. It is a step-by-step, serial, discrete-time process. It is performed on symbols which have a semantics that is similar to the semantics of numerical systems. It is performed on an electronic device that consumes a lot of electricity. It is specified entirely and fully by the text of its program. The programming language contains imperatives and logical expressions, and possibly ways of moving data from one place to another. It has input. It has output. It stops when it has transformed the input into output. This is the popular conception of computing. The meaning of the English word contains these implicit inferences. Even if we define other kinds of computing in the technical discipline, this is what computing suggests in the language, the picture that appears in our heads.

Hence, AI must be some kind of computation which is restricted to electronic devices that have to be plugged in, which has a kind of banal repetitiveness, and which is capable of numerical precision. This is not true. If there are other kinds of computations, then there may be other things we should be calling AI.

It is instructive here to compare the folk understanding of physics. Physics is a theory that is monolithic and unchanging. It is deterministic and it permits the calculation of future state, if only the initial state could be fully specified. Not true. Physics consists of many mathematical models of different physical phenomena, some of which are still subject to revision, not all of which can be easily reduced to some more fundamental physical theory, not all of which are deterministic or presume to provide a comprehensive representation of initial state from which subsequent state can reliably be derived. We have to remind ourselves when we are doing philosophy that physics is larger and broader and subtler than the subject that high school physics textbooks present.

Computing, according to Alan Turing, could be done on any Turing machine. Since I can draw and operate a Turing machine on the back of a McDonald's napkin, apparently many of the connotations of computing must be wrong. A Turing machine, or any abstract computational device, can be operated by anyone who elects to follow the rules of the machine's program upon the relevant symbols. I can simulate a Turing machine on a chalkboard

or on a sidewalk flat enough for chalk. Simulating a Turing machine is the same as operating a Turing machine. It is just a mathematical abstraction. The same is true of arithmetic calculation. Simulating two plus two is the same as calculating two plus two. Likewise, this simulation of computation is the computation itself. This is an essential property of a formal system. It is exactly why, in a brilliant episode of the *Next Generation Star Trek* involving the HOLODECK where virtual worlds are constructed like novels, the revelation of the password in a simulated reality constitutes an actual revelation of the password in the real world. It does not take an electronic computer to perform a computation. In fact, anything willing and able to perform a computation can turn itself into a computer, at least for the time that it is following the program.

It doesn't matter whether the abstract machine being simulated on a napkin is a Turing machine. It could be a push-down automaton or a finite state machine, a two-counter machine, a RAM machine, a Moore machine. Anyone can simulate a finite state machine with pencil and paper, and thus can be performing a computation. A two-register machine with 8-bit addition and one level of address indirection is also a machine that can be simulated on a piece of paper. There is nothing special about Turing's machine that makes it the only disembodied, abstract, mathematical calculating device, the only computer that does not need to be plugged into an electrical outlet. There are lots of mathematical, abstract models of computation. It is simply that Turing realized quite early that the rule-governed manipulations of symbols in his colleagues' logical systems were things that could be made explicit. Freshmen at MIT used to arrange themselves on their quad in three lines, each holding a 1 or a 0 in front of himself, the group simulating binary addition on n-bit words. Or so it was alleged. In this case, the three lines of 8x1 freshmen allegedly simulated computation. Since simulating a computation is the same as performing the computation, they allegedly computed.

Many computer scientists who are in their mid-forties remember a plastic computer called Digi-comp, which could be programmed using configurations of small rubber bands that were identical to the rubber bands placed by orthodontists on awkward teenagers' braces. Engineers who are at the head of the Baby Boom generation remember the Addometer, made by the Reliable Typewriter and Adding Machine Company of Chicago, which had rotors one could turn with a stylus on a board about the size of a math teacher's small paddle, which functioned like a glorified and user-friendly automotive trip odometer. It is the great discovery of the past decades that the electronic computer can store large and complex programs, process data that has considerable representational complexity, and proceed through a causal chain of clocked events that permits computations to proceed inexor-

ably from start to finish with no human intervention and no mechanical human motivation. We should not underestimate the importance of this discovery, as electronic computers are far better than steam-driven computers might have been. But mechanical computation has been taking place on the fabric loom, for a century, and on the abacus, for a millennium.

There are analog computers, too, not just digital computers. These computers are like the distortion boxes that rock guitarists used to make their outlandish sixties and seventies music, except that an analog computer presumably computes something useful, like the Fourier transform of a waveform. These computers do not work step-by-step, in discrete time. They work in continuous time. One cannot pause the computation at an intermediate step, sit down for an espresso, then resume the computation. They do not have algorithms in the usual sense of sequential transformation and derivation. Analog computers are so different from digital computers that no one, not even computer science professors, considers them to be within the purview of computational method and engineering. Perhaps analog computers still exist in a few electrical engineering departments.

Randomized algorithms do not always produce the same result when they are repeated on identical inputs. Randomized algorithms use chance or pseudo-chance devices in their calculations. The programmer often needs there to be a chance element in order to generate desirable behavior. If two ether-linked devices try to talk at the same time, each backs off for a probabilistically determined time, then tries again. If the algorithm did not contain randomness, the two devices would continue to collide over and over again. So we randomize as part of the program which is called the ethernet transmission protocol.

Here is another example. When simulated annealing is used to produce a heuristically optimal solution to a combinatorial optimization problem, random numbers are used repeatedly in order to generate the behavior. The idea is to simulate a physical system being cooled from a high temperature, which happens to produce very nearly optimal structures, and the best way we know to simulate the behavior of molecules at a given temperature is to have the selection of subroutines governed by probability. The algorithm does not work without the randomness. Some people like to argue that most of these algorithms use pseudo-random numbers, and that pseudo-random number generation is itself an algorithmic computation. This is an irrelevant objection. Some algorithms do not need randomness in the strictest idea of what is random, so we permit pseudo-randomness. There are algorithms that are written to make use of, or even require, input that is genuinely random: randomness that is no less random than the most random things known.

Distributed algorithms are not maximally specific. They usually control the interactions of traditionally programmed subsystems by saying how the subsystems communicate, what they communicate, and when they communicate, as they progress through their own individual programs. A protocol for distributed tree search for example, may require programs to search their own subtree, then report the results of their individual searches when they are moving to the next deeper level in the tree, so that they can each decide what is the best subtree to explore next, based on the accumulated reports from other programs. This protocol does not attempt to control the times at which each program completes its period of individual activity nor even the times of synchronized communication. There is no attempt to synchronize, just to organize.

The program does not attempt to control the alternation of the programs in their reporting, by imposing a strictly alternating or turn-taking regimen, for example. There are many different systems that could implement the distributed algorithm faithfully, some in which all of the processors contributing computation are equally matched in speed, and others in which there is considerable disparity in the speed of individual processes. It is well known that a distributed algorithm is more like the rules of the road than it is like a recipe. A recipe says what to do, step by step. Mix the flour and the butter, then add eggs. A distributed system is different. Drive on the right, pass on the left. A set of constraints for a loosely coordinated or loosely coupled system says that you can do whatever you want, so long as you follow a couple of simple rules.

Actually, most programming languages are not maximally specific about their semantics. They are usually silent about the semantics of performance. There is nothing in the definition of the C language that requires subtraction to take about the same time as addition. Many programming languages actually leave the semantics of certain aspects of the language unspecified, so that the implementer of the language is free to interpret some of the program's meaning. For example, in the programming language I love, called gawk, the semantics of a "getline x" operation are not defined when there is no line to get. In most implementations, one can expect the variable, x, to remain unchanged. But this is not specified by the language, so the programmer has a bit of uncertainty when writing this in a program. The system that computes the results of this program has room for a bit of unconstrained, elective, or free behavior. Most computer science specialists do not think about this when they are programming, but it is important to realize that programmers are specific only about the things they care about, and are silent on many of the things that they do not think about specifying. Many programs are written in such a way that there is no explicit control of the state unless

a specific condition arises. They react to special conditions. They do not say how to progress from some state into a state that satisfies the condition.

Some computations do not have logical start and end points. There are certainly computations which can last as long as the rules can be followed. For example, the program which simply increments a value can run forever. Turing machines can be programmed with some of the most interesting programs, yet they might never stop and produce output. Operating systems are wonderful examples of programs that do not have natural input beginnings and output endings. Any operating system which organizes the machine during boot time is confronted with an initial state which can be considered its initial input. But it certainly receives additional input over time. And it may not always have a natural initial state. The thing on the CD called Red Hat Linux 7.2 confronts the initial state of my machine every time it is run. But if I could upgrade from Red Hat 7.1 to 7.2 without rebooting, it might be hard to say what was the input to the computation that is the operation of Linux 7.2 on my machine. The program evolved into existence during a nondescript state of the machine, so what was the input to the program?

Most people want to say that the operating system is a collection of distinct programs that have beginnings and endings. The Windows operating system runs the program to defragment the disk when you tell it to, for example, just as the Linux kernel calls *kswapd* periodically. But there are several problems with this view. First, there is a program called the operating system, the exact scope of which may be unclear. Does it include the loader or call the loader, or does the loader call it? Does it include *httpd*, or does it run *httpd* as an application every time it is booted? Surely it is possible for an operating system company to make its browser an essential part of the operating system. Second, there are lots of programs called object-oriented programs that are being written these days that look like little operating systems. These programs control how messages are communicated between objects and services, and are less specific about when objects and services pop into existence or go away. The average code-monkey sitting in a cubicle, being mocked by Dilbert and Dogbert, is as likely to be specifying a protocol for the interaction of information processors as to be writing imperatives for a commandable automaton. The programmer is telling school kids how to play nicely, not telling a dog how to lie down, then roll over. Note that the protocol for interaction might suppose nothing about the degree to which the individual processors are automated. Some of the objects or servers might actually be humans.

The conclusion is that the folk conception of computing is flawed. Computations aren't required to be electronic. They can take place with pencil and

paper, or with plastic computers. Humans can certainly perform computations if they elect to do so. When you calculate a sum, you are computing. In that case, you are ironically constraining your natural intelligence to perform a bit of artificial numerical intelligence. Computations aren't necessarily repeatable or deterministic. Programmers love to have their computations flip a coin or roll the electronic dice at a crucial point in the process. Computer programs aren't necessarily fully specified, even with respect to a specific machine model.

Programs can depend on environmental variables, such as the time at which they are started, or the amount of memory on a machine, or the temperature of the processor. Programs don't necessarily transform numerical input to numerical output in discrete time. Sometimes they are simply trying to maintain an admissible state, like the operating system, which is trying its best not to crash. More and more programs and intelligent systems are starting to combine the individual acts of multiple agents into a coordinated whole, so that no matter how each agent behaves, or fails, the system's behavior is desirable or useful. Each of the stereotypes of computation can be defeated. When philosophers, journalists, politicians, fiction writers and movie directors think of computations they think of tables of logarithms and Texas Instruments chips. We have web-based infrastructure, wearable devices, unmanned combat air vehicles, sensor networks, and multi-player virtual reality environments. Computation is clearly a much broader phenomenon.

So AI, which is a kind of computation, can be performed with programs that take random inputs? Certainly. Can AI take place with programs that are distributed across multiple processors, or with programs that are not maximally specific with respect to their semantics in a precise machine model? Certainly. Can AI be performed on an abstract device, such as a chalkboard? Certainly, the program can be simulated and some real-world decision can automatically depend on the result of the simulation.

Surely there are limits to what can be considered a computation. Not just any rule following process counts as a computation. Photosynthesis may follow certain rules, from the point of view of an botanical observer, but it is not a computation. When I digest a hot dog, there are input and output, but it is not a computation. Computation must take place on symbols. Symbols can be identified by the fact that many physical arrangements could substitute for the very same symbol. When I write an "a" on a piece of paper, it is a symbol because it does not matter how large I write the "a", nor how darkly, nor how thickly, nor how deeply. It is an "a" because we agree that it is not a "b" and we can successfully distinguish it from a "b" during the processing of the symbol. I intend it to be the symbol "a", thus it is the symbol "a". Photosyn-

thesis is a processing of actual physical things, not symbols. Digesting hot dogs likewise is not a symbol-processing process. Digesting hot dogs is a hot-dog-processing process.

There must be rule following, some rules followed, in order for there to be a computation. Wittgenstein asks whether one can calculate a sum by accident. Apparently not. The rules must be followed by intention, purposefully, and the rules must be effective, at least some of the time. There may be many ways to arrive at a sum besides calculating the sum by rule following. One may guess well, have associative memory recall or possess a special clairvoyance for sums. Those are ways of finding or arriving at the right answer. Calculating the answer is a different matter. One may calculate the wrong sum, and one still performs the computation of intending to compute a sum. Machines can make mistakes. While writing this essay, my laptop memory failed. It started producing incorrect sums even while it was computing sums. But one cannot calculate a sum if the rules one is following never produce a correct sum.

Can one compute if one is compelled to follow the rules of a program and could not do otherwise? Can one intend to follow a set of rules which one could not actually deviate from, if one did not intend to follow the rules? I think not. I believe there must be some elective component for there to be intention. A Chinese arithmetic savant who calculates the sums of increasing pairs of numbers, and who does so compulsively, and who does not recognize the results of the calculations to be sums, is a busy Chinaman indeed. He is busy with numerical rule following, but is not computing anything. We might query this person at the right time and find that the very next sum he has compulsively computed exactly fits our informational needs. But in this case, we are using the savant's compulsion to help us with our computation.

Apparently there are mollusks that are unaware they are computing golden-mean arithmetic progressions when they grow their shells. Likewise, electronic computers don't actually compute because they cannot intend interpretations of their symbols. Yes, I just said that electronic computers don't actually compute. We, the programmers, intend the data to mean something, so the computation is ours, not the computer's. We use the computer to compute; hopefully the computer does not initiate its own computations and intend its own symbol interpretations. Certainly in the day-to-day parlance, the word "computes" is ambiguous in this respect: the computer computes, but also, we use the computer to do our computation. Despite the ambiguity, the careful philosopher would not want computation to take place on symbols that had no interpreter. The computer computes only because we are using it to compute.

Suppose my computer happens to busy itself by always repeating the last program on the last inputs until it is presented with a new program or new input. Think of this echo computation as an elaborate screen saver. Maybe my processor is like a Formula One racing machine and it is engineered to stay warm. Then it is certainly computing when I am using it to follow my program. After that, when it is simply busying itself with rule-following, I can't be sure it is computing. It might be following someone else's rules at that point, which might have semantics at some meta-programming level, but I am done using it to compute and I can't judge with certainty that it is computing. If indeed, as Douglas Adams has it in the *Hitchhiker's Guide to the Galaxy*, we are all living on this planet in order to provide simulation runs to an extraterrestrial intelligence that is using Earth as its computer, then we are computing only insofar as someone can interpret what it is that we are being used to compute.

One possible exception to the demand that computing be intentional and elective is the class of autonomous robots and other AI systems that are coupled to the real world, which thus ground their symbols in their environments. I say possible exception because there is room for disagreement between insightful people here. In these cases, after a hypothetical many years of operation, symbol-processing systems might have only a tenuous claim to be computing based on the long-lost semantics of their designers; they might have a better claim to be simply aspects of the physical environment, like the trees and the flowers, and quite possibly, the mice in the garden. Not surprisingly, there are many *Star Trek* episodes that contemplate the scenario of a long-situated rule-following machine that outlives its designers, their culture, and their civilization. Programmers who abandon their programs, but leave them in place to continue their interaction with the physical environment, are indeed special kinds of creators. There may be a theological point to be made here. Meanwhile, on a much smaller scale, there may be a question, too, of why the mouse in the garden cannot provide the requisite intentionality for symbol systems and computations.

The Third Kind of Computation

The architecture of the machine seems to be unimportant in the separation of computation from non-computation. So too does the nature of the program seem unimportant. Computation arises from elective (intentional) and purposive (teleological) rule-following on symbols. The test for computation is the existence of a program, and a protocol for interaction is a program. The test for computation is not the existence of electronics, causal determinism, work ethic, predictability or precision.

I claim that many, indeed most, social interactions are computations. They are programmed, and they are often symbol-ingesting, symbol-producing, and purposive. Both the machine and the programmer elect to perform those computations. We, in fact, are the machine. We are also the programmers, though some of us do a lot more rule-making, and others of us have to do a lot of rule-following.

The legislator who is contemplating the proposal of a new set of rules such as a tax incentive for rental property owners, a revised penal code, welfare child care rules, a method for demonopolizing a telecommunications industry, or a procedure for revoking a business license — that legislator is doing a kind of programming. I don't mean he is programming in a shallow sense; there are some very weak, non-technical uses of the word "programming." When people are greeted at the door of a dance concert, they receive a program for the evening. That is not the kind of program I mean. The legislator is trying to achieve some coordinated activity by governing the relationships between the objects in the system; what is required is to conceive of all the possible ways the proposed rules could be followed by people. The legislator is trying to decide whether the possible outcomes are all acceptable, as a function of the inputs. If the outcomes are not all acceptable, then more rules, or fewer rules, or different rules should be proposed. This is a kind of programming in a deeper sense than the dance program, which is merely a scripting. The legislative exercise is exactly the mental exercise that the distributed programmer performs when writing a protocol for the interaction of processes such that no matter how those processes behave individually, their aggregate behavior is admissible. Something like "Freshman activities are programmed during Freshman week" is actually more like the legislator's programming than the choreographer's or playwright's.

In one example, the machine consists of the population of welfare recipients. Who conceives of welfare mothers as a machine? Again, we have to confront the preconception that a machine is something that does what it is told to do, and which can have its entire behavior determined sequentially. The operating system does not consider its peripheral components to be fully controllable, though it must consider those components as part of the machine. An old scanner might be particularly unruly and uninstructable. Still, the code which polls the state of the machine when other software would like to communicate with the scanner treats this electronic free-thinker as if it were a part of the machine. "Programmable machine" originally meant that the behavior can be scripted, its state transitions controlled at a detailed level. "Programmable machine" now means that it generates a desirable behavior no matter how its individual subcomponents behave, so long as they interact according to the rules of the program.

It is not so difficult to conceive of a machine which consists of otherwise free-thinking and freely acting individual people. We often refer to the big city mayor's political machine. We can also think of an eighteenth century European army of infantrymen who are instructed to engage the enemy in lines, and to fall in, so as to fill the line, wherever a comrade has fallen. This was a programmed military machine, and in some countries, it was better programmed than in others. I am not the first one to conceive of social organization this way, of course. We are told to "get with the program," or we can be "in or out of the system." We are advised by teenage rockers to "rage against the machine." If we think of a group as mechanistic, it is because it is governed by rules. It is no accident that Herbert Simon helped found artificial intelligence as a field, then turned to the school of management and founded the field of organizational science.

If a person is particularly good at a task, and can perform that task indefatigably, incessantly, concentrating on the task for long periods of time, achieving results reliably and consistently, we refer to that person as a machine. She is a golfing machine. He is a data entry machine. These are just metaphors that play on the idea that machines do not tire and can attain a kind of syntactic perfection. It is an undeserved stereotype of computation, as we have seen. I have plenty of computing machines in my house that have tired from mere modest use and which do not operate according to specification. Meanwhile, some groups of people are so well organized according to explicit rules, that the group operates not only metaphorically as a machine, but also satisfies my definition of a machine that follows a distributed program or protocol and is therefore performing a distributed computation. A hospital runs like clockwork because its staff is a health-care-providing and patient-processing machine. An intelligence agency is so effective at communicating discoveries and hypotheses between investigative groups that it operates like an information-ingesting machine. Ironically, the intelligence agency is functioning like an artificial intelligence program. The Dallas Cowboys' two minute offense under their legendary quarterback Roger Staubach was so efficient that it was a first-down machine. The synchronized acts of wide receiver Drew Pearson were essential to this machine. In fact, even if the Cowboys had not been efficient on the offense, if they had not been a touchdown-scoring machine, and the explanation for this was mainly a lack of passing and catching talent, but they still possessed impeccable organizational efficiency, then we would still be right to call them a machine. The Princeton University men's basketball offense comes to mind here. Computation is about having mechanism, not exhibiting tireless perfection.

At root in this conception of distributed program and social machine is the idea that game playing is a kind of computation. It seems odd to say that two

chess players compute the outcome of their game, just because they followed the rules of the game. It seems a bit less odd, but still unusual, to say that the stock market rules were followed, so that the legitimate change in the Dow Industrial average, +100 points today, was computed by the day's buyers and sellers. There seems to be no right answer, yet we want to say that the answer was computed. But distributed and probabilistic programs are just like this. We say that the program computed the heuristically optimal solution to be at a point that is valued 10% less than the exact optimal. How did it do this? By having algorithmic subcomponents that played a game according to the rules of heuristic optimization.

They played a genetic programming game, where members of populations were represented by distinct process threads, or they played some asynchronous randomized gradient search game with d-players in d-dimensions. We say that a group of computers elected a new nameserver after the prior nameserver crashed. They computed the result of this election by following the rules of an election game, even if there is no "right answer" to compare to the result they computed. If it helps, consider the chess players to have a purpose. They are playing for a charity, and the donation of their sponsors is based on the number of moves required to achieve checkmate, thirty dollars per move. By achieving checkmate in 42 moves, the players have determined, and computed, that the sponsors owe the charity $1260. I suspect half as many would object if we were to say that the change in the Dow, +100 points, was computed by the traders on the floor, under the instructions of the buyers and sellers they represented that day.

I just separated my AI class into two groups, the UNIX people and the non-UNIX people, and had them stand on separate sides of the room. Then we played a game. We had to pair the students in teams for their programming project. Each student had to look at a student on the opposite side of the room. When they were looking at the person who was looking back, they could sit down. Perhaps this was a bit cruel, but it was very time and space efficient. No one doubts that I programmed this asynchronous interaction, or that the result was a partition on the class membership into pairs. No one doubts that we performed a distributed computation.

Games are all around us. We play tennis a few times a year, and a game of Monopoly with the nephews at Christmas, and enter a couple of lotteries for free mattresses and tickets to baseball games. We play Rock-em-Sock-em robots to let off some steam, and some fantasy football, and do some off-track betting if we aren't good enough at bingo and cribbage. But the real games have much bigger consequences. We are players in national and local election games, in prosecutor-defender jury games, and in investment and insurance

and income and taxation games. We are surrounded by procedures for determining social outcomes, economic distributions, legal responsibilities, and contractual obligations. I recently played the tenure game. I don't mean that I did anything consciously to manipulate the outcome of the tenure process. I mean that there was a process which took my actions to be inputs and which was rule-governed, imposed on me by a social institution. These are not games to be taken lightly. They are serious games, the outcomes of which can affect many people deeply. I am happy not to be a participant in a mandatory formal process of selection for military service, and I do think about the fairness, as well as the honor, of the economic game that binds those who do choose to serve.

A socially constructed reality requires games in order to generate its states. It is logically possible to arrange society so that all things are decreed: you two are now married; you are the duke; you own the property from here to there; you are my vassal; you are a criminal. How simple for the Mayan chief or the Dungeons and Dragons dungeonmaster simply to say what is and is not the case. But people like to think that there is procedural justice, or at least procedural appropriateness, in their social world. Otherwise they "nullify" by rejecting the authority of the decree. They start to construct their own social realities, a counter-culture and revolution, and the end of shared social reality. How strange indeed to have two putative governments claim sovereignty, or two transitional courts claim jurisdiction. Games build their outputs on the inputs of the players. By giving each person a stake in the construction of the outcome, the procedure confers legitimacy upon the outcome.

John Rawls talks about pure procedural justice and the two-stage process of splitting a pie. The purest example of a game that constructs reality, to me, is the game rock-paper-scissors (jon-ken-po), which creates a chance device from the strategic inputs of two players. There are perfectly good examples of tournaments and elections that are strategies for conferring legitimacy on asymmetric pronouncements by starting with symmetric opportunities and investing players in the outcomes. There are lots of interesting examples. Flip a coin to determine who goes first. Arm wrestle for the best seat in front of the TV. Joust for the right to marry the girl. Play a seeded 64-team single elimination tournament for the right to call yourselves NCAA basketball champion.

Ex post non-exchangeability is legitimized by ex-ante exchangeability and the proper stake in a defensible transforming process. The transforming process institutionalizes a meritocracy of some socially recognized value, such as the ability to win tennis games, or to argue clearly, or to amass wealth, or to teach one's children to respect one's parents. Society is governed by formal

symbol-processing games because the alternatives to games are worse. We write social rules to produce outcomes as the result of our collective inputs because we want programs that the social machine will be willing to follow. The right games played the right way makes a right society.

The symbols are things like dollars, championships, inheritance, property rights, adoption papers, confirmations, excommunications, and proclamations. Almost any relationship between two members of society or among multiple members of a society, that is not a physical relationship, is something invented by a social institution and could be written in an official log. These governing rules are therefore symbol-processing rules. Society prescribes computations.

Where, then, is the AI in the virtual worlds of social construction? Where are there systems that are wholly rule-governed, that function as individuals, that have agency and standing on a par with you and me? I have not yet mentioned corporations. They are an obvious class of candidates. They are even granted the status of individuals or persons under our laws. A corporation is easy to see as a socially emergent AI which leeches onto our social computations, which is faceless, sphexish, incessant, emotionless, and more than likely evil. But there are other candidates. Bureaucracies are rule-governed actors in society. Lots of them even have initials like the IRS, FBI, INS, EPA, NIH, FJC, and EEOC.

Actually, every institution is a rule-governed actor in society. Washington University is. If you are trying to hold on to land in the Delmar-Washington or Pershing-Skinker area, Washington University might seem a lot like AGENT SMITH in the real estate investment, land use and rezoning virtual reality game. If you sue Washington University, you will again find yourself toe-to-toe with someone who punches very hard in the virtual world that is called the Missouri civil court system. Even individual positions within an institution, if they are formally specified or hierarchically controlled positions, are an AI to be found in the social computation.

I recently realized that I have spent the past couple of years being mostly unhappy with our Dean, the position, not our Dean, the man. Anyone else occupying that position would have had the obligations and incentives to act exactly as our Dean does. There is a person who sits in the office and acts so as to interface the rest of the University according to the rules of its organization: I have no problem with him. It's the position. The position entails a certain responsibility, so he has to act a certain way. Indeed, our Dean is a kind of AGENT SMITH in the virtual reality constructed by the institutions of our Engineering School.

Not all socially emergent AI is evil. The Red Cross is a well-run organization, but no one fears its improved efficiency. So it is the case with the local food shelter and the Salvation Army. Service First, when the freshmen arrive on campus and are taken by bus to a local public school to paint walls, runs like clockwork. It is so well organized and has such clear traditions that it is an institution. Whoever wrote that program had a pretty good idea how to get the freshman dorm part of the social machine to perform. When I was growing up, the Jerry Lewis Telethon was a fund-raising machine, but that was as much because Jerry Lewis worked non-stop for two days as it was because the Telethon was explicitly rule-governed and symbol-processing. Charitable foundations are all good examples of AI. The whole idea of a charitable foundation that funds NPR or Public TV, or Baylor Medical School, from the estate of a former captain of industry is to mechanize the activity of a social actor after the original actor's passing. What better a comparison to HAL, C3PO and DATA, than the Annenberg, the MacArthur, the William and Flora Hewlett Foundations.

So how should we feel about AI and virtual worlds? Virtual worlds have been with us since we started lifting civilized man from the mud. In fact, any symbol-processing, explicitly rule-following, social (even linguistic) species can construct social worlds. And thank God for those social worlds, because in them, we can expect with some certainty that good things happen, like contractual enforcement and free speech, and bad things don't happen, like theft and libel. As soon as there is sufficient organizational complexity that institutions can emerge as actors, those institutions constitute AI in the social computation. They provide inputs to the social games and process information culminating in decision-making, just like the human actors in the social computation. Their behavior arises from a programming which may be a shallow collection of if-then rules, or which itself may be a distributed program that constructs outputs upon the regulated inputs of human and artificial actors. Maybe the Dutch West India Company was indeed a bigger deal than the Industrial Revolution.

These artificial social actors, too, have been with us for centuries. They have been growing in their complexity and their formality. As our social discourse becomes increasingly constructed, the range of their activities has been increasing. The advent of the electronic computer, which makes it possible to institutionalize a complex set of rules efficiently, also makes it possible to create effective artificial social actors en masse. In the fifties, only corporations like General Motors and AT&T could afford to have offices that interacted with an organizational depth and discipline that inspired the awe and earned the respect of smaller businesses. Now, the mom and pop laundromat not only can program when and from whom it receives faxes, can control its

lights and air conditioners at night and its surveillance cameras to react to motion, but also can network around the world so as to raise its search engine rank position on google and yahoo, the thoroughly modern form of marketing.

The sad reality is that the government and the courts have an expectation that businesses will have an information-processing capacity that will meet the demands of complaints, audits, employee regulations, environmental regulations, licensing, tax reporting, and so on, as demanded by all the other information-devouring organizations that can have a piece of them. The institutional knock falls on your family's door too. It used to be that the emperor's guard would arrive with a writ every few generations asking for an ox or two. Maybe depression-era families had to type a letter a couple of times a month. Now, your daily mail contains information transactions from your credit card company, the auto licensing bureau, an occasional unrecognized law firm, an occasional class action notice, your health insurance company, the company that owns the magazine you subscribe to, the company that wants to provide you with cable TV service, the neighborhood action group, the condo association, and the invoice for the computer parts you needed so you can respond to all of the above. How much of your email has become official correspondence, requiring action, notice, or response? The proliferation of automated social actors has become a lot more frightening than the robot on the floor of the Honda plant in Ohio that paints minivan doors with four coats of burgundy before passing it along for further assembly.

Lewis Mumford wrote about this in *Art and Technics*, even before Marvin Minsky, John McCarthy, and Herbert Simon started talking about AI on computers. He warned about the dehumanizing effect of the aesthetics of machines. We could classify him with other Luddites if his prose were not so presciently focused on information technology and organization. We have been warned about *Organization Man* by William Whyte and his followers, too, who despised the rising class of middle management, surely the main engine of social AI. I am here to renew the concern of these earlier social critics.

I am not so worried that the automatic teller machine can replace the human teller. I am worried that there at one time popped into existence a job called "human teller" a large part of which was so easily replaced by eighty lines of a computer program. I am worried when a company shows me the twenty page "training manual" used to program several hundred people in a call center, where phones ring, people in cubicles answer, and decisions are made according to the rule book, and they say they want to know whether the book can be turned into an expert system. It's not the part about programming the expert system that concerns me. It's the people in the cubicles. Even if they were put in a big park and had laptops and cell phones, their jobs would still stink. The same people who created organizations with information-processing roles for human automatons, the bureaucrats and the bureaucracies of the fifties, are now unleashing upon us actor upon actor of automated social role. Every job has a title, every job has a description, every organization has a mission, every office has automation, and every social AI has just enough standing to engage you in an information-processing dialogue that will kill your creative life with their hundreds of thousands of cuts.

My fire alarm keeps going off in my home, apparently because I have a faulty sensing device. I have never actually gotten angry at it for its false alarms, as the cost of its poor decision-making is just a few hours of sleep here and there. But the alarm monitoring people are truly vexing. They phone us whenever the alarm has tripped, and they insist that the password that we give over the phone is not the password they have written down (apparently, we give a common contracted form of the password, which they don't recognize as a valid variation; so much for the flexibility of humans compared to machines). Only through a mixture of begging, cussing, and threatening termination of the contract are we able to avoid their programmed response, the dreaded phone call to the local fire department which incurs all sorts of fines. It seems that an average of six F-words will gain access to the call-taker's supervisor.

I certainly don't blame the person on the other end of the phone, nor even the caller from the Direct Marketing Association who wanted to sell me something before there was a no-call rule passed and enforced in this state. It is just the world we live in. Dehumanizing, indeed. Not the fact that I live with a dumb electronic device that makes flawed decision-making, no; the real problem is that human beings are employed as devices, cogs in the machine, and have been stripped of the authority to use their own judgment. When they plug in to society, their brains and their hearts get turned off. Some of us have to fight this. We have to avoid using forms when a blank sheet of paper is just as good. We have to invent words when we are being poetic. We have to go places where the computer cannot go. We have to be inefficient when we don't believe in the process. Smell the roses. Drag your feet when the rules are not right. We have to create. We have to dream. We have to speak of grand purposes, sometimes, instead of reducing all things to their details.

As Morpheus tells Neo in the *Matrix*, "most of these people are not ready to be unplugged; they are so hopelessly inured, so hopelessly dependent on the system that they will fight to protect it." There are certainly people who

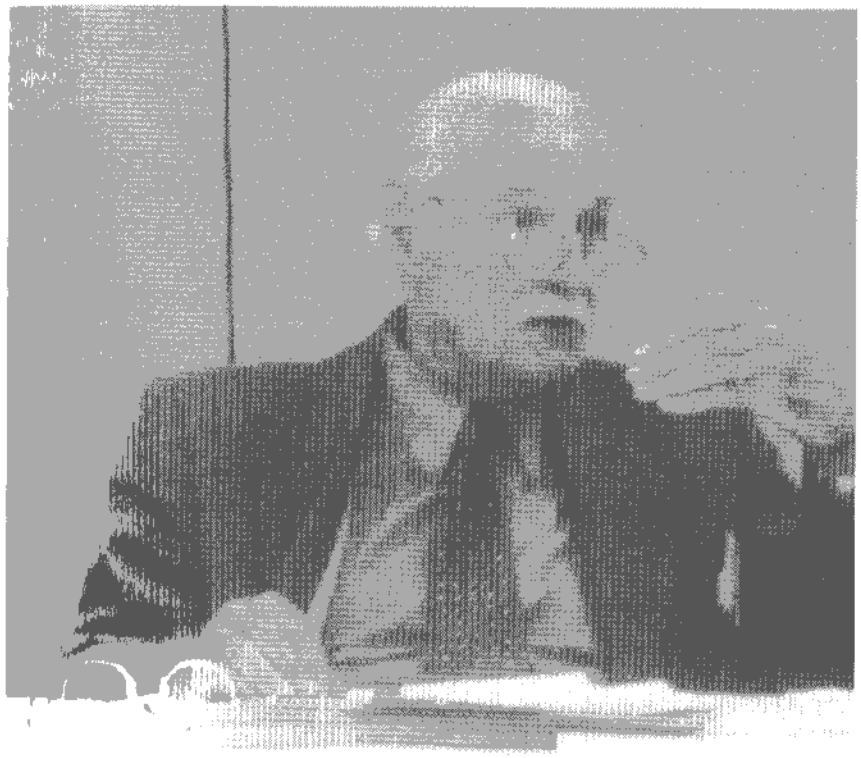
follow the rules just because those are the rules. There are also Neo's among us who are able to rise above the nexus of official regulations and social obligations. Like sixth-stage-Kohlberg moral actors of principled conscience, some of us are able to perceive the purposes of organization and act so as to improve upon its inefficiencies. We nullify the pedestrian no-crossing rules by jaywalking on empty streets. We don't send in payment when the bill is less than five cents, credit report be damned. We are willing to broadcast chain letters through the internet if we can prove that they will terminate within a reasonable bound. We are not engaged in petty civil disobedience, but we can think about the rationales of rules.

Ideally, two people who can perceive better rules for a system will act as if those rules were in place, as if governed by a Kantian imperative for social programs. Those who have the skill of programming systems can see through the imperfections of the current system to its purposes and practical exigencies, its processes and procedures, and through to its revision. Such actors can see that their lack of freedom derives from an inadequate specification of the rules. They take back their freedom to act, confident that under appeal, the correctness of their perceptions will cause the rules to be amended, not their acts to be penalized. Of course, there are people who think they know better than the system who are mistaken. Some automobile drivers don't signal their turns because they don't think it matters and they are wrong.

There is a "St. Louis stop" at a four-way stop, where two cars at right angles are each supposed to stop, then proceed in first-in first-out (FIFO) order. Instead, one driver stops, and the other one drives through the intersection without stopping. These people think they have discovered a dominant solution, where each driver gets through the intersection in less time than the prescribed rules would allow, because it is faster even for the driver who has stopped, for the intersection to clear than it is for that driver to wait for the other driver to come to a full stop. But the FIFO order is a social value that is reenforced by the original system, and lost under the informal revision. Not every potential Neo is The One. People should, however, be encouraged to view the rules as corrigible, to learn all the rationales and cases that govern their institutions, learn the values that their programs are intended to advance, and learn the methods for effectively controlling social interactions.

A better society is one in which people are programmers, not just programmed. If there is going to be AI in the virtual world that is our constructed society, then it might as well be AI that has survived the programming of many well-informed critics. My dog has a better solution. She just never enters the world of our symbol-processing. She doesn't know about

ebay and easements, Anglo-American queuing norms, or even marriage certificates. She has no complaints about AI because she doesn't really know when a program's rules are being followed. The world she occupies is the real world, with dirt and apples, and some lower-case a's that smell differently from other lower-case a's. Our world unfortunately encroaches on her world when we put cars on roads with signal lights that she can't interpret. I think that's a crime, and I am certain that AI is in some way to blame.



Doctor John Cross tells us the fascinating story of the brain.