THE NEW ATLANTIS
A JOURNAL OF TECHNOLOGY & SOCIETY

Why Minds Are Not Like Computers

Ari N. Schulman

When the blackbird flew out of sight,
It marked the edge
Of one of many circles.
–Wallace Stevens

People who believe that the mind can be replicated on a computer tend to explain the mind in terms of a computer. When theorizing about the mind, especially to outsiders but also to one another, defenders of artificial intelligence (AI) often rely on computational concepts. They regularly describe the mind and brain as the "software and hardware" of thinking, the mind as a "pattern" and the brain as a "substrate," senses as "inputs" and behaviors as "outputs," neurons as "processing units" and synapses as "circuitry," to give just a few common examples.

Those who employ this analogy tend to do so with casual presumption. They rarely justify it by reference to the actual workings of computers, and they misuse and abuse terms that have clear and established definitions in computer science—established not merely because they are well understood, but because they in fact are products of human engineering. An examination of what this usage means and whether it is correct reveals a great deal about the history and present state of artificial intelligence research. And it highlights the aspirations of some of the luminaries of AI—researchers, writers, and advocates for whom the metaphor of mind-as-machine is dogma rather than discipline.

Conceptions of the Computer

Before any useful discussion about artificial intelligence can proceed, it is important to first clarify some basic concepts. When the mind is compared to a computer, just what is it being compared to? How does a computer work?

Broadly speaking, a computer is a machine that can perform many different procedures rather than just one or a few. In computer parlance, a procedure is known as an *algorithm*—a set of distinct, well-defined steps.

Ari N. Schulman *is assistant editor of* The New Atlantis.

Suppose, for example, that you work in an office and your boss asks you to alphabetize the books on his shelf. There are many ways you could do it. For example, one approach would be to look through all of the books and find the first alphabetically (say, *Aesop's Fables*), and swap it with the first book on the shelf. Then look through the remaining unsorted books again, find the next highest, and swap it with the book after *Aesop's Fables*. Keep going until you have no unsorted books left. This procedure is known as "selection sort" because the approach is to select the highest unsorted book and put it with the sorted books.

The algorithmic approach, as this example shows, is to break up a problem into a series of simple steps, each of which requires little thought or effort. In this particular procedure, the number of specified steps is fairly small—but when you actually perform a selection sort to organize a bookshelf, the number of steps you execute will be much larger, because most of the steps are repeated for each book. The heart of most useful algorithms is repetition; selection sort accomplishes a task with one basic operation that, when performed over and over, completes the whole task. An algorithm doesn't necessarily have to involve repetition, but any task performed on a large set of data usually will use such repeated steps, known as "loops." Selection sort also has a well-defined start state (the unsorted shelf) and end state (the sorted shelf), which can be referred to as its input and output. Algorithms have a well-defined set of steps for transforming input to output, so anyone who executes an algorithm will perform the same steps, and an algorithm's output for a given input will be the same every time it is executed (even so-called "randomized" algorithms are deterministic in practice).

Algorithms involve several forms of abstraction. First, an algorithm consists of clear specifications for *what* should be performed in each step, but not necessarily clear specifications for *how*. In essence, an algorithm takes a problem specifying *what* should be achieved and breaks it into smaller problems with simpler requirements for *what* should be achieved. An algorithm should specify steps simple enough that *what* becomes identical to *how* as far as the person or machine executing the algorithm is concerned. How specific the steps need to be in order for this identity to occur depends on the intelligence of the executor. Returning to the example of sorting your boss's books, the step in which you select the highest unsorted book is more complex than, say, the one in which you pull that book off the shelf and swap it with another. For a highly intelligent sorter, how to execute this step may be self-evident; a less intelligent sorter may need the details spelled out (perhaps like this: write down the

first unsorted title; for each remaining book, check its title; if it's higher, cross it off and write it down instead, along with where it is on the shelf so you can quickly find it again). This routine can be considered a sub-procedure of the original algorithm.

Suppose you wanted to pay someone else to organize the books for you using selection sort. You could simply write the original few steps on a pad of paper in the level of detail at which they were first described. But when you include the step about "selecting the highest unsorted book," since another person might not know how to do it, you could include the note "see page 2 for instructions on how to do this," and then list the steps of this sub-routine on page 2. The intelligence of the sorter would lead you to specify more or fewer detailed sub-routines, depending on what steps the sorter already knows how to do. The tasks that an executor can perform in which the *what* can be specified without the *how* are known as "primitive" operations.

There is also an abstraction in the description of the objects involved in the algorithm. Certain assumptions are made about their nature. In our example, the books have titles composed of known characters, allowing for alphabetization; the shelf has an ordering (beginning to end, or left to right); the books are objects that can fit onto the shelf and be moved about; and so on. These characteristics may seem rather obvious—so much so that they are inextricable from the concepts of "book" and "shelf"—but what is important is that *only* these few properties are relevant for the purposes of the algorithm. You, as the sorter, need know nothing about the full nature of a book in order to execute the algorithm—you need only have knowledge of shelf positions, titles, and how titles are ordered relative to one another. This abstraction is useful because the objects involved in the algorithm can easily be represented by symbols that describe only these relevant properties.

These two forms of abstraction are at the core of what enables the execution of procedures on a computer. At the level of its basic operations, a computer is both extremely fast and exceedingly stupid, meaning that the type of task it can perform in which the *what* is the same as the *how* is very simple. For a computer to perform the selection sort algorithm, for example, it would have to be described in terms of much simpler primitive steps than the version offered here. The type of steps a computer can perform are usually about as complex as "tell me if this number is greater than that number" and "add these two numbers and tell me the result." The power of the computer derives not from its ability to perform complex operations, but from its ability to perform many simple operations

very quickly. Any complex procedure that a computer performs must be reduced to the primitive operations that a computer can execute, which may require many levels at which the procedure is broken down into simpler and still simpler steps.

Manipulating Symbols

Imagine that you have a computer with three useful abilities: it has a large number of memory slots in which you can store numbers; you can tell it to move existing numbers from one slot to another; and it can compare the numbers in any two slots, telling you which is greater. You can give the computer a sequential list of instructions to execute, some examples of which could be, "Store the number '25' in slot 93," "Copy the number from slot 76 into slot 41," "Tell me whether the numbers in slots 17 and 58 are equal," and "If the last two numbers compared were equal, jump back four instructions, otherwise keep going." Could you use such instructions to perform your book-sorting task?

To do so, you must be able to represent the problem in terms that the computer can understand—but the computer only knows what numbers and memory slots are, not titles or shelves. The solution is to recognize that there is a correspondence between the objects that the computer understands and the relevant properties of the objects involved in the algorithm: for example, numbers and titles both have a definite order. You can use the concepts that the computer understands to *symbolize* the concepts of your problem: assign each letter to a number so that they will sort in the same way (1 for A, 26 for Z), and write a title as a list of letters represented by numbers; the shelf is in turn represented by a list of titles. You can then reduce the steps of your sorting job into steps at the level of simplicity of the computer's basic operations. If you do this correctly, the computer can execute your algorithm by performing a series of arithmetical operations. (Of course, getting the computer to physically move your boss's books is another matter, but it can give you a list ordered the way your boss wanted.)

This is why the computer is sometimes called a "symbol-manipulation machine": what the computer does is manipulate symbols (numbers) according to instructions that we give it. The physical computer can thus solve problems in the limited sense that we imbue what it does with a meaning that represents our problem.

It is worth dwelling for a moment on the dualistic nature of this symbolism. Symbolic systems have two sides: the abstract concepts of the symbols

themselves, and an instantiation of those symbols in a physical object. This dualism means that symbolic systems and their physical instantiations are separable in two important (and mirrored) ways. First, a physical object is independent of the symbols it represents: Any object that represents one set of symbols can also represent countless other symbols. A physical object and a symbolic system are *only* meaningfully related to each other through a particular encoding scheme. Thus it is only partially correct to say that a computer performs arithmetic calculations. As a physical object, the computer does no such thing—no more than a ball performs physics calculations when you drop it. It is only when we consider the computer through the symbolic system of arithmetic, and the way we have encoded it in the computer, that we can say it performs arithmetic.

Second, a symbolic system is independent of its representation, so it can be encoded in many different ways. Again, this means not just that it is independent of any particular representation, but of any particular *method* of representation—much as an audio recording can exist in any number of formats (LP, CD, MP3, etc.). The same is true for programs, in which higher-level concepts may be represented in any number of different ways.

This is a crucial property of algorithms and programs—another way of stating that an algorithm specifies *what* should be done, but not necessarily *how* to do it. This separation of *what* and *how* allows for a division of knowledge and labor that is essential to modern computing. Computer users know that most popular programs (say, Microsoft Word or Mozilla Firefox) work the same way no matter what computer they're running on. You, as a user, don't need to know that the instructions a Windows machine uses to run the program are entirely different from those used by an Apple machine. This view of the interaction between user and program is known to software engineers as a "black box," because the user can see everything on the outside of the box—*what* it does—but nothing on the inside—*how* it does it.

Black boxes pervade every aspect of computer design because they employ three distinct abstractions, each offering tremendous advantages for programmers and users. The first has already been described: a user needs to know only what a program does, so he need not repeat the programmer's labor of understanding how it does it. The same is true for programmers themselves, who need to know only what operations the computer is capable of performing, and don't need to concern themselves with how it performs them. Second, black box programming allows for simple machines to be easily combined to create more complex machines; this is called *modular* programming, as each black box functions as a module that

can be fitted to other modules. The final abstraction of modular programming is perhaps its greatest advantage: the *how* can be changed without affecting the *what*. This allows the programmer to conceive of new ways to increase the efficiency of the program without changing its input-output behavior. More importantly, it allows for the same program to be executed on a wide variety of different machines. Most modern computer processors offer the same set of instructions that have been used by processors for decades, but execute them in such a dramatically different way that they are performed millions of times faster than they were in the past.

Computers as Black Boxes

Let's return to the hypothetical task of sorting your boss's bookshelves. Suppose that your employer has specified *what* you should do, but not *how*—in other words, suppose he is concerned only with transforming the start state of the shelf to a desired end state. You might sort the shelf a number of different ways—selection sort is just one option, and not always a very good one, since it is exceedingly slow to perform for a large number of books. You might instead decide to sort the books a different way: first pick a book at random, and then move all the books that alphabetically precede it to its left, and all the books that alphabetically follow to its right; then sort each of the two smaller sections of books in the same way. You'll notice that the operations you use are quite different, but your employer, if he notices any change at all, will only note that you completed the task faster than last time. (Called "quicksort," this is in fact the fastest known sorting algorithm.)

Or, as suggested, you might pay a friend to sort the books—then potentially *you* would not even know how the sorting was performed. Or you could hire several friends, and assign to each of them one of the simpler parts of the task; you would then have been responsible for taking a complex task and breaking it into more simple tasks, but you would not have been responsible for how the simpler tasks themselves were performed. Black box programming creates hierarchies of tasks in this way. Each level of the hierarchy typically corresponds to a differing degree of complexity in the instructions it uses. In the sorting example, the highest level of the hierarchy is the instruction "sort the bookshelf," while the lowest is a collection of simple instructions that might each look something like "compare these two numbers."

Computers, then, have engineered layers of abstraction, each deriving its capabilities from joining together simpler instructions at a lower layer of abstraction. But each layer uses its own distinct concepts, and

each layer is causally closed—meaning that it is possible to understand the behavior of one layer without recourse to the behavior of a higher or lower layer. For instance, think about your home or office computer. It has many abstraction layers, typically including (from highest to lowest): the user interface, a high-level programming language, a machine language running on the processor, the processor microarchitecture, Boolean logic gates, and transistors. Most computers will have many more layers than this, sitting between the ones listed. The higher and lower layers will likely be the most familiar to laymen: the user interface creates what you see on the screen when you interact with the computer, while Boolean logic gates and transistors give rise to the common description of the computer as "just ones and zeroes."

The use of layers of abstraction in the computer unifies several essential aspects of programming—symbolic representation, the divide-and-conquer approach of algorithms, and black box encapsulation. Each layer of a computer is designed to be separate and closed, but dependent upon some lower layer to execute its basic operations. A higher level must be translated into a lower level in order to be executed, just as selection sort must be translated into lower-level instructions, which must be translated into instructions at a still lower level.

The hierarchy of a computer is not turtles all the way down: there is a lowest layer that is not translated into something lower, but instead is implemented physically. In modern computers this layer is composed of transistors, miniscule electronic switches with properties corresponding to basic Boolean logic. As layers are translated into other layers, symbolic systems can thus be represented using other symbols, or using physical representations. The perceived hierarchy derives partially from the fact that one layer is represented physically, thus making its relationship to the physical computer the easiest to understand.

But it would be incorrect to take the notion of a hierarchy to mean that the lowest layer—or any particular layer—can better explain the computer's behavior than higher layers. Suppose that you open a file sitting on your computer's desktop. The statement "when I clicked the mouse, the file opened" is causally equivalent to a description of the series of state changes that occurred in the transistors of your computer when you opened the file. Each is an equally correct way of interpreting what the computer does, as each imposes a distinct set of symbolic representations and properties onto the same physical computer, corresponding to two different layers of abstraction. The executing computer cannot be said to be *just* ones and zeroes, or *just* a series of machine-level instruc-

tions, or *just* an arithmetic calculator, or *just* opening a file, because it is in fact a physical object that embodies the unity of *all* of these symbolic interpretations. Any description of the computer that is not solely physical must admit the equivalent significance of each layer of description.

The concept of the computer thus seems to be based on a deep contradiction between dualism and unity. A program is independent of the hardware that executes it; it could run just as well on many other pieces of hardware that work in very different ways. But a program is dependent on *some* physical representation in order to execute—and in any given computer, the seemingly independent layers do not just exist simultaneously, but are in fact identical, in that they are each equivalent ways of describing the same physical system.

More importantly, a description at a lower level may be practically impossible to translate back into an original higher-level description. Returning again to our sorting example, suppose now that a friend hires *you* to do some task that *his* boss asked him to perform. All he gives you is a list of instructions, each of which is about as simple as "decide if these two numbers are equal." When you follow these instructions, you will perform the task exactly as your friend has specified, but you may have no idea what task you are performing beyond comparing lots of numbers. Even if you are able to figure out that, say, you are also doing some kind of sort, it could be impossible to know whether you are sorting books rather than addresses or names. The steps you execute still clearly embody the higher-level concepts designed by your friend and intended by his boss, but simply knowing those steps may not be sufficient to allow you to deduce those original concepts. In the computer, then, a low-level description of a program does provide a causally closed description of its behavior, but it obscures the higher-level concepts originally used to create the program. One may very likely, then, be unable to deduce the intended purpose and design of a program, or its internal structure, simply from its lower-level behavior.

The Mind as Black Box

Since the inception of the AI project, the use of computer analogies to try to describe, understand, and replicate mental processes has led to their widespread abuse. Typically, an exponent of AI will not just use a computer metaphor to describe the mind, but will also assert that such a description is a sufficient understanding of the mind—indeed, that mental processes can be understood entirely in computational terms. One of the most pervasive abuses has been the purely functional description of

mental processes. In the black box view of programming, the internal processes that give rise to a behavior are irrelevant; only a full knowledge of the input-output behavior is necessary to completely understand a module. Because humans have "input" in the form of the senses, and "output" in the form of speech and actions, it has become an AI creed that a convincing mimicry of human input-output behavior amounts to actually achieving true human qualities in computers.

The embrace of input-output mimicry as a standard traces back to Alan Turing's famous "imitation game," in which a computer program engages in a text-based conversation with a human interrogator, attempting to fool the person into believing that it, too, is human. The game, now popularly known as the Turing Test, is above all a statement of epistemological limitation—an admission of the impossibility of knowing with certainty that any other being is thinking, and an acknowledgement that conversation is one of the most important ways to assess a person's intelligence. Thus Turing said that a computer that passes the test would be *regarded* as thinking, not that it actually *is* thinking, or that passing the test constitutes thinking. In fact, Turing specified at the outset that he devised the test because the "question 'Can machines think?' I believe to be too meaningless to deserve discussion." But it is precisely this claim—that passing the Turing Test constitutes thinking—that has become not just a primary standard of success for artificial intelligence research, but a philosophical precept of the project itself.

This precept is based on a crucial misunderstanding of why computers work the way they do. The implicit idea of the Turing Test is that the mind is a program and a program can be described purely in terms of its input-output behavior. To be sure, some programs *can* be defined by what output they return for a particular input. For example, our sorting program would always return a sorted shelf when given an unsorted shelf. You don't need to know how the sorting program works, just what it does. However, many other computer programs cannot be described without referring to how they work. Given a program you did not create, attempting to completely explain its behavior without referring to its internal structure is exceedingly difficult (and likely impossible) unless the designer has provided you with its specification. A description of a program solely in terms of its "black box" input-output behavior can only completely describe the program when that description is supplied by the person who designed it.

Over the past decade, the "mind as black box" standard has led to the creation of a wide array of "socialized" robots that mimic humans and animals. AI partisans have suggested that robotic pets might serve as

substitutes for their messy flesh-and-blood counterparts, and that computer programs could replace expensive human psychotherapists. These suggestions reveal a troublingly low standard for our interactions with other beings, as the robots were created not so much to *be social* as to *elicit socialized responses* from humans. But more than believing that their mimicry makes them sufficient human companions, the makers of socialized robots often state that their creations actually possess human traits. Robots that mimic facial expressions are said to experience genuine emotions—and for more than half a century, researchers have commonly claimed that programs that deliver "intelligent" results are actually thinking.

Such statements reveal more than just questionable ethics—they indicate crucial errors in AI researchers' understanding of both computers and minds. Suppose that the mind *is* in fact a computer program. Would it then be possible to conclude that what's inside the mind is irrelevant, as is supposed by some interpretations of the Turing Test? If we have some computer program whose behavior can be completely described as if it were a black box, such a description does not mean that the box is empty, so to speak. The program must still contain some internal structures and properties. They may not be necessary for understanding the program's external behavior, but they still exist. So even if we possessed a correct account of human mental processes in purely input-output terms (which we do not), such an external description by definition could not describe first-person experience. The Turing Test is not a definition of thinking, but an admission of ignorance—an admission that it is impossible to ever empirically verify the consciousness of any being but yourself. It is only ever possible to gain some level of confidence that another being is thinking or intelligent. So we are stuck measuring correlates of thinking and intelligence, and the Turing Test provides a standard for measuring one type of correlate. Similarly, although social interaction requires communication in the form of such "input-output" as speech and hearing, it also requires two or more agents who experience that interaction: A teddy bear may provide a child with the same comfort and companionship—and elicit from the child the same responses—that another human being would, but we cannot say that it loves.

AI proponents understand that communication is possibly the most important way of demonstrating intelligence, but by denying the importance of each agent's internal comprehension, they ironically deny that any real meaning is conveyed through communication, thus ridding it of any connection to intelligence. While AI partisans continue to argue that the existence of thinking and social interaction in programs is demonstrated

by their mimicry of observed human input-output behavior, they have merely shifted the burden of proof from the first-person experience of the programs themselves to the first-person experiences of the people who interact with them. So although behaviorists and functionalists have long sought to render irrelevant the truth of Descartes' *cogito*, the canonization of the Turing Test has merely transformed *I think therefore I am* into *I think you think therefore you are.*

What We Talk About When We Talk About AI

Much artificial intelligence research has been based on the assumption that the mind has layers comparable to those of the computer. Under this assumption, the physical world, including the mind, is not merely understandable through sciences at increasing levels of complexity—physics, chemistry, biology, neurology, and psychology—but is actually organized into these levels. Moreover, much work in AI has assumed that the layers of the mind and brain are separable from each other in the same manner that the computer is organized into many layers of abstraction, so that each layer can be understood on its own terms without recourse to the principles of lower levels. These assumptions underlie the notion that the mind is a "pattern" and the brain is its "substrate."

If this notion is true, then the processes that give rise to the mind must consist of some basic rules and procedures implemented in the brain. The mind, then, is a program, and the brain is but a computer upon which the mind is running. In this understanding, the brain must contain some basic functional unit whose operations enable the implementation of the procedures of the mind. For those AI researchers interested in actually replicating the human mind, the two guiding questions have thus been (1) What organizational layer of the mind embodies its program? and (2) At what organizational layer of the brain will we find the basic functional unit necessary to run the mind-program? Especially for researchers who believe "strong AI" is tenable—that is, those who believe that computers can be programmed to be intelligent, conscious, and emotional—their aims and methods can be understood as a progression of attempts to answer these two questions. But when closely examined, the history of their efforts is revealed to be a sort of regression, as the layer targeted for replication has moved lower and lower.

The earliest AI efforts aimed for the highest level, attempting to replicate the rules underlying reason itself. In 1955, Alan Newell and Herbert Simon created Logic Theorist, a program that modeled human

problem-solving methods in order to prove mathematical theorems. When it successfully solved many of the theorems in Bertrand Russell's *Principia Mathematica*, Simon famously told students, "Over Christmas, Al Newell and I invented a thinking machine." The next year, leaders in the burgeoning field held the first AI conference in Dartmouth; their conference proposal asserted the foundational creed of AI: that "every aspect of learning or any other feature of intelligence can be so precisely described that a machine can be made to simulate it." Implicit in this statement are answers to the fundamental questions: (1) Reason itself, the highest level of the mind, embodies the mind's program, and (2) The basic functional unit of the brain is irrelevant.

The approach of "precisely describing" intelligence and learning was applied to a wide array of simple problems, meeting with great success until the late 1970s, at which time researchers began to realize that many seemingly simple problems—such as recognizing objects in an image, navigating a robot around an obstacle course, or understanding a children's story—could not be solved using methods that "precisely described" intelligence and learning. (Later research in image recognition has met with more success, while children's stories are still too difficult for computers, despite decades of effort.)

Seldom now do researchers attempt to replicate a basic mental task by tackling the salient features of the task on its own terms—that is, by "precisely describing" aspects of learning and intelligence. Instead, they apply generalized techniques to specific problems; those techniques have often proved successful as they have become more sophisticated, as more training data has become available, and as raw computing power has increased. But these techniques are usually based on concepts and terms that bear only the vaguest resemblance to the problems they are employed to solve. For example, facial recognition technology has been achieved using a variety of generalized pattern-matching techniques. Such programs are fed large sets of training data, giving them pictures specified to contain faces; they then measure scores of properties of the images and apply mathematical models to determine if any properties can be used to predict the presence of the same features in new images. While effective, this approach is distinctively more low-level than that of early AI methods. The implicit answers to the foundational questions have thus become (1) The basic program of the mind is a generalized, unconscious process that is able to learn many different sorts of tasks, and (2) The basic functional unit of the brain is still mostly irrelevant, but it must be something with access to the brain's vast storage space and immense computational power.

Most AI research today is conducted under these working assumptions, albeit with much room for variation among researchers. Few researchers now have the goal of devising a single method that will by itself give rise to a thinking machine; instead, typical research projects attempt to tackle small subsystems of intelligence. The hypothesis, again, is that the separability of the mind into layers implies that each layer, like a computer system, is composed of distinct modules that can be studied and replicated independently. Among those researchers whose ultimate goal is still to create a truly thinking machine, the hope is that, when the subsystems become sufficiently advanced, they can be joined together to create a mind.

The majority of AI researchers consider that ultimate goal to be far off. Today, asserting one's goal to "precisely describe every aspect of learning or any other feature of intelligence" would be a professional embarrassment; most researchers rather would describe their own work as primarily attempting to solve sophisticated classes of problems. But in recent years, the attention of some strong-AI proponents has shifted from problem-solving methods back to the original goal of creating a thinking, feeling computer. The hope of quickly and directly replicating some essential process of the mind has waned—but no matter, for now the prospect of replicating the brain itself appears, at least to some (as we shall see), to be achievable. The brain could be replicated on the computer, and supposedly, along with it, the mind—thus achieving artificial intelligence without understanding intelligence at all. In this regression to the lowest level yet of the hierarchy of the mind, the answers to the foundational questions become explicit: (1) The brain itself embodies the program of the mind, and (2) The basic functional unit is the neuron.

This regression has gone practically unremarked in all the reporting and scrutiny of the successes and failures of AI. It goes a long way toward explaining why participants in the AI debate who share the same knowledge of computers nonetheless arrive at conflicting and seemingly incommensurable conclusions about the possibility of strong AI: even though they may be employing similar arguments and even making identical statements, they are often referring to entirely different levels of the hierarchies of computer and mind.

On the one hand, arguments *against* strong AI, both moral and technical, typically describe the highest levels of the mind—consciousness, emotion, and intelligence—in order to argue its non-mechanical nature. This type of argument is everywhere; an eloquent example appears in the 1976 book *Computer Power and Human Reason* by the late computer scientist Joseph Weizenbaum:

> The theories…that have hypnotized the artificial intelligentsia, and large segments of the general public as well, have long ago determined that life is what is computable and only that.…Sometimes when my children were still little, my wife and I would stand over them as they lay sleeping in their beds. We spoke to each other in silence, rehearsing a scene as old as mankind itself. It is as [playwright Eugène] Ionesco told his journal: "Not everything is unsayable in words, only the living truth."

The implication is that the essence of human nature, and thus of the mind, is profound and unknowable; this belief underlies Weizenbaum's extensive argument that the mind cannot be described in procedural or computational terms.

On the other hand, arguments *for* strong AI typically describe the lowest levels of the mind in order to assert its mechanical nature. The rhetoric of mechanism pervades the writing of AI believers, who claim again and again that the brain is a machine. In his 2002 book *Flesh and Machines: How Robots Will Change Us*, roboticist Rodney Brooks declares that "the body, this mass of biomolecules, is a machine that acts according to a set of specifiable rules," and hence that "we, all of us, overanthropomorphize humans, who are after all mere machines." The mind, then, must also be a machine, and thus must be describable in computational terms just as the brain supposedly is.

Both these positions fail to acknowledge that the mind may be simultaneously like and unlike a machine, depending on the level at which it is being described. That is, perhaps it is the case that the highest levels of mentation cannot be described in computational terms, but some lower level can. At the very least, it should be acknowledged that, in computers, conceptual frameworks used to describe different levels of the same system may be entirely different, and the same will be true of the mind if it is also a computer—so it should be argued rather than assumed that a true claim about the attributes of a high level is also a true claim about the attributes of a low level, or vice versa. Although there are indeed genuine and important fundamental disagreements in the AI debate, a great many of them are obscured by this type of confusion.

Confusion in the Chinese Room

An instructive example of this confusing conceptual gap can be found in the heated debate surrounding one of the most influential articles in the history of computer science. In a 1980 paper, the philosopher John R.

Searle sketched out a thought experiment in which a man who speaks only English sits in a room with a batch of Chinese symbols and a set of instructions written in English. Interrogators outside of the room slip questions written in Chinese in through the door; the man inside understands no Chinese, but based solely on the English instructions and the shape of the Chinese characters, he constructs answers, which he then slips back out through the door. Even though the interrogators might believe that they are interacting with a person who understands Chinese, we know that the man inside the room does not understand Chinese. Searle's scenario is, of course, designed to be analogous to how an operating AI program works, and is thus supposedly a disproof of the claim that a computer operating a similar program could be said to "understand" Chinese or any other language—or indeed, anything at all. Some defenders of strong AI have replied that understanding *is* taking place, if not by the man, then by the room as a whole.

Searle and his critics have typically debated his argument as though it were a unified whole, but it is actually possible to separate it into at least three distinct claims, rarely distinguished among by either his defenders or detractors. The first and simplest claim is that even if human beings sometimes use formal systems while reasoning, merely replicating those formalisms on a computer is not enough to make the computer understand in the same way that a human solving a problem using the same formalisms would. Graphing calculators, for instance, can solve complex equations using the same formal reasoning as people, and air-conditioning systems decide when to switch on and off according to a simple set of formal rules—but few people would argue that their calculator understands math, or that their air conditioner literally "knows" that it turns on because the air is too hot.

Following from this first seemingly uncontroversial claim is a second one that is stronger and more general: that a program could appear to possess understanding without actually understanding. This is a direct refutation of the basic hypothesis of strong AI exemplified in the Turing Test. Searle is essentially claiming that the appearance of human mental states is not sufficient to establish their existence—that imitation is not the same as replication.

From this claim, and the assumption that his thought experiment is a generalized description of any program that exhibits some degree of intelligence, Searle makes a third and substantially stronger claim: that no computer program could account for mental states—and thus that no machine which consists of a program running on a computer can think.

But the Chinese Room scenario is *not* such a generalized description, and so the jump to this third claim relies on a basic logical error, namely that demonstrating that *one* particular program falsely appears to have mental content does not demonstrate the same for *all* potential programs.

Indeed, it is worth remembering that Searle's thought experiment was not even originally intended as a general analogy of all computer programs. As first formulated, the Chinese Room scenario was a direct analogy of a program created by computer scientist Roger C. Schank. Schank's program answered basic questions about simple stories; Searle sought to demonstrate that, despite its correct answers, Schank's program could not be said to understand the stories in the same way that a human being could. But both Searle himself and the respondents to his paper over the years have mostly taken his thought experiment to be a description of any program that could pass the Turing Test—that is, of a program that could participate in a sustained and convincingly human conversation with a human being. Indeed, Searle's later reformulations of the scenario in the paper describe exactly such a situation. But there is a key difference between these two scenarios, which should be obvious from the fact that the original and simpler formulation sought to analogize a program that already existed in 1980, whereas the latter formulation describes a program so immensely complex that it still does not exist. Which of these scenarios one has in mind when discussing the argument is extremely important.

Some of Searle's critics have interpreted his broad third claim to be about the impossibility of thinking arising at a high level from processes that are unthinking at a low level. They essentially accuse Searle of making an error of layers, falsely equating the properties of one layer of a system with that of another. Thus the most common rebuttals to the Chinese Room thought experiment invoke, in some way, the "systems reply": although the man in the room does not understand Chinese, the whole system—the combination of the man, the instructions, and the room—indeed does understand Chinese. In other words, they argue that the understanding occurs at the highest level of the system, not at the lowest level.

Searle's response to this argument—that the "systems reply simply begs the question by insisting without argument that the system must understand Chinese"—is surely correct. But Searle himself, as AI enthusiast Ray Kurzweil put it in his 2005 book *The Singularity Is Near*, similarly just declares "*ipso facto* that [the room] isn't conscious and that his conclusion is obvious." Kurzweil is also correct, for the truth is somewhere in between: we cannot be sure that the system does or does not understand

Chinese or possess consciousness. Both Searle and his critics largely sidestep the central questions, which are about how the hierarchy of layers of the mind compares to the hierarchy of the computer.

Searle in particular leaves unanswered some crucial questions about how he believes computer hierarchies and physical hierarchies work, and how these two conceptions are similar and different. On the one hand, he asserts that no formal model of a computer program "will ever be sufficient by itself" for mental properties—such as the ability of thoughts to be *about* something—because formalisms "have by themselves no causal powers except the power, when instantiated, to produce the next stage of the formalism when the machine is running." Searle claims this as the reason, demonstrated through his thought experiment, that a computer could not think. Considered alongside this claim, one of the most befuddling sections of his 1980 paper is this:

> "OK, but could a digital computer think?"
>
> If by "digital computer" we mean anything at all that has a level of description where it can correctly be described as the instantiation of a computer program, then again the answer is, of course, yes, since we are the instantiations of any number of computer programs, and we can think.

More so even than the casual assertion that people are computer programs, this section of Searle's paper is surprising in its contradiction of his own claim that computers cannot think. On Searle's account, then, can computers think or not? The answer reveals just how confused is the common understanding of computer systems.

The interpretation of Searle's argument hinges on what exactly he considers a "correct" description to entail, but this matter is not clear from his paper. In a computer system, a "correct" description at some level is also a *complete* causal description, meaning that it accounts for all behavior at that level. But as the preceding discussion should make clear, a complete description at one level of a computer system is equivalent to a complete description at any other level. Recall again the scenario in which your boss asks you to sort his books. It is correct to describe the scenario either (1) by saying that the books are sorted, or (2) by listing each and every detailed step you executed. Both descriptions are also complete, because they can each be used to explain why any event in the system occurred. However, the descriptions will employ distinct concepts and properties to explain the same physical system—so the question, "Why is *Aesop's*

Fables first on this shelf?" could be answered with a long list of arithmetic instructions that, while a correct answer, make no reference to important concepts like "titles," "shelves," and "sorting." Possessing a description of one layer of a computer system does not necessarily give you a description of another layer of the system, even though the two are equivalent.

So Searle's statement that no computer program "will ever be sufficient by itself" for mental properties may be true if he means that a computer program could not *describe* mental properties themselves. Searle seems to take it, that is, that some running programs *can* think, but they cannot think simply *because* they are programs that think when executing. But if (as he supposes) there is some such program that can run the mind, then the program and a mental description would both be correct layers of description of the same physical system. Any layer of that system, including the computational, *must* be sufficient to *replicate* and *give rise to* any higher layers, such as the mental—even if it does not describe them. The important questions, then, are about whether the mind really is a computer system—and if not, whether or not the physical world in which minds reside is like a computer system.

Procedures, Layers, and the Mind

The Chinese Room debate points to the salient questions underlying the common discussions about whether a computer can be made to think. It is incumbent upon participants in these discussions to make more responsible and clear use of their knowledge of the nature of computers, and, more importantly, to explicitly argue for their implicit claims about the ways in which the mind is or is not like the computer. Above all, these arguments must address the crucial distinction that the mind is a natural system while the computer is engineered, and so they must demonstrate that the design principles of computers also apply to natural systems. The most crucial questions about the mind arise from our profound ignorance of just how it is that its characteristic abilities—intelligence, consciousness, intentionality, and emotion—arise from processes that have nothing resembling these traits. The crucial questions about the possibility of creating a thinking computer, then, must address this seeming paradox, and participants in that debate must begin with an acknowledgment that the very fact of the mind's improbable existence means that the answers are likely to defy our intuitions.

Properly understood, the first question underlying the AI debate is: *Can the properties of the mind be completely described on their own terms as an*

algorithm? Recall that an algorithm has a definite start and end state and consists of a set of well-defined rules for transitioning from start state to end state. As we have already seen, it was the explicit early claim of AI proponents that the answer to this question was yes: the properties of the mind, they believed, could be expressed algorithmically (or "procedurally," to use a more general term). But the AI project has thus far failed to prove this answer, and AI researchers seem to have understood this failure without acknowledging it. The founding goal of AI has been all but rejected, a rejection that carries great significance for the central presumption of the project but that has gone largely unremarked. As an empirical hypothesis, the question of whether the mind can be completely described procedurally remains open (as all empirical hypotheses must), but it should be acknowledged that the failure thus far to achieve this goal suggests that the answer to the question is no—and the longer such a failure persists, the greater our confidence must be in that answer.

Once the unlikelihood of procedurally describing the mind at a high level is accepted, the issue becomes whether the mind can be replicated at some lower level in order to recreate the high level, raising the next important question: *Are the layers of physical systems, and thus the layers of the mind and brain, separable in the same way as the layers of the computer?* A physical system is any portion of the physical world examined in isolation—for instance, a basketball bouncing on a gym floor, an oak tree, or a hot stone tossed into a cold pond. No portion of the physical world, of course, is ever completely in isolation from the rest of the world, but it is easier and can be helpfully accurate to use such an abstraction for the purposes of analysis. In the example of the stone in the pond, one need consider only the properties and matter of the stone, and how they interact with the world—say, by leaking heat into the surrounding water. As applied to this example, then, the question becomes, *Can I consider the heat properties of the stone without examining the molecules that compose it, and vice-versa?* The question, of course, becomes much more complicated when the physical system is a living or thinking being.

As explained above, it is correct to explain computers in terms of separable layers, since that is how they are designed. Physical systems, on the other hand, are not designed at all. They exist prior to human intent; we separate them into layers as a method of understanding their behavior. Psychology and physics, for example, can each be used to answer a distinct set of questions about a single physical system—the brain. We rely on hierarchies to explain physical systems, but we actually engineer hierarchies into computers.

The layers of a computer are separable because the behavior of any single level can be explained without recourse to some higher or lower level. You can study a computer system at any level of description and explain the entire behavior of that level solely in terms of causes from that level. This is why a computer processor does not need to know that the instructions it is executing are for, say, a Web browser, and why the person using that browser doesn't need to know what those instructions are in order to understand what the browser does. If physical systems work in the same way, then we should be able to do biology without knowing chemistry, chemistry without knowing physics, and so on. That is, we should be able to *completely* describe the behavior of, for example, a system of chemicals without any recourse to physics. Of course, this is impossible. When we describe physical systems at one layer, we can abstract away the properties of the lower layer to simplify our explanation—but a complete description must take into account the lower-level properties.

This raises the related question that the AI project has implicitly posed since its inception: *What is the basic functional unit of the mind?* If the mind were a computer, it would be possible to completely describe its behavior as a procedure. That procedure would have to use certain basic operations, which are executed by some functional unit of the brain. The early hypothesis of AI was that this question was essentially irrelevant since the mind's operations could be described at such a high level that the underlying hardware was inconsequential. Researchers today eschew such a large-scale approach, instead working under the assumption that the mind, like a computer program, might be a collection of modules, and so we can replicate the modules and eventually piece them back together—which is why research projects today focus on very specific subsystems of intelligence and learning.

As the high-level AI project has failed to meet its original goal, some attention has returned to the study of the brain itself under the belief that, if nothing else, we might make a computer think by simply copying the brain onto it. The unit of the mind typically targeted for replication is the neuron, and the assumption has thus been that the neuron is the basic functional unit of the mind. The neuron has been considered a prime candidate because it appears to have much in common with modules of computer systems: It has an electrical signal, and it has input and output for that signal in the form of dendrites and axons. It seems like it would be relatively straightforward, then, to replicate the neuron's input-output function on a computer, scan the electrical signals of a person's brain, and boot up that person's mind on a suitably powerful computer. One

proponent of this possibility, Zenon Pylyshyn, a professor at the Rutgers Center for Cognitive Science, describes the "rather astonishing view" of what would happen if replicating the neuron's external behavior were *not* sufficient to replicate the mind:

> If more and more of the cells in your brain were to be replaced by integrated circuit chips, programmed in such a way as to keep the input-output *function* of each unit identical to that of the unit being replaced, you would in all likelihood just keep right on speaking exactly as you are doing now except that you would eventually stop *meaning* anything by it. [His emphasis.]

If Pylyshyn is right that the neuron's role in the mind is entirely a matter of its electrical input-output specification—that is, that the neuron is a black box—then he is clearly correct that the conclusion of his thought experiment is astonishing and false.

But what if the neuron is not a black box? Then Pylyshyn's thought experiment would be a description of the slow death of the brain and the mind, in the same manner as if you were to slowly kill a person's brain cells, one by one. The task in defending the soundness of Pylyshyn's argument, then, is first to demonstrate that the neuron is a black box, and second, to demonstrate that its input-output specification can be attained with complete fidelity. But since the neuron has no designer who can supply us with such a specification, we can only attempt to replicate it through observation. This approach, however, can never provide us with a specification of which we can be *completely* confident, and so if this task is to be undertaken in real life (as some researchers and activists are seriously suggesting that it should be) then a crucial question to consider is *what degree of fidelity is good enough?* Suppose we were to replicate a computer by duplicating its processor; would it be sufficient to have the duplicate correctly reproduce its operations, say, 95 percent of the time? If not, then 99.9 percent? When running a program containing billions of instructions, what would be the effect of regular errors at the low level?

That the neuron even has such a specification is hardly a foregone conclusion. It has been shown that mental processes are strongly affected by anatomical changes in the brain: a person's brain structure and chemical composition change over the course of his lifetime, chemical imbalances and disorders of the neurons can cause mental disorders, and so on. The persistence of the mind despite the replacement of the particles of the brain clearly indicates some causal autonomy at the high level of the mind, but the fact that the mind is affected by structural changes in

the brain also indicates some causal autonomy at the low level of the brain. So while transhumanists may join Ray Kurzweil in arguing that "we should not associate our fundamental identity with a specific set of particles, but rather the pattern of matter and energy that we represent," we must remember that this supposed separation of particles and pattern is false: Every indication is that, rather than a neatly separable hierarchy like a computer, the mind is a tangled hierarchy of organization and causation. Changes in the mind cause changes in the brain, and vice versa. To successfully replicate the brain in order to simulate the mind, it will be necessary to replicate *every* level of the brain that affects and is affected by the mind.

Some defenders of the brain-replication project acknowledge this problem, and include in their speculation the likelihood that some structure lower than the level of the neuron may have to be included in a simulation. According to Kurzweil, the level at which the functional unit resides is a rather unimportant detail; if it is lower than commonly supposed, this may delay the project by only a decade or two, until the requisite scanning and computing power is available. Pylyshyn similarly asserts, "Let Searle name the level, and it can be simulated perfectly well."

So where and when should we expect to find the functional unit of the mind, and how far removed will it be from the mind itself? We may have to keep going further and further down the rabbit-hole, perhaps until we reach elementary particles—or perhaps the fundamental unit of the mind can only be found at the quantum level, which is decidedly nondeterministic and nonprocedural. We may ultimately come face to face with the most fundamental question of modern science: *Is nature itself procedural?* If physicists can indeed construct a "Theory of Everything," will it show the universe to consist of particles with definite positions and deterministic rules for transitioning from one state to the next? The outcome of the AI project may depend on this deepest of inquiries.

The Future of the AI Project

The questions presented here suggest an approach for further exploration of the project of strong AI. Attempts to prove or disprove the viability of this task would benefit from efforts to address these questions. But even if these questions are left unaddressed, participants in the AI debate should seek to avoid error and confusion by beginning with a clear understanding of computer principles and, when discussing properties of the mind, being specific about which level they are referring to.

Intriguingly, some involved in the AI project have begun to theorize about replicating the mind not on digital computers but on some yet-to-be-invented machines. As Ray Kurzweil wrote in *The Singularity is Near*:

> Computers do not have to use only zero and one.... The nature of computing is not limited to manipulating logical symbols. Something is going on in the human brain, and there is nothing that prevents these biological processes from being reverse engineered and replicated in nonbiological entities.

In principle, Kurzweil is correct: we have as yet no positive proof that his vision is impossible. But it must be acknowledged that the project he describes is entirely different from the original task of strong AI to replicate the mind on a *digital* computer. When the task shifts from dealing with the stuff of minds and computers to the stuff of brains and matter—and when the instruments used to achieve AI are thus altogether different from those of the digital computer—then all of the work undertaken thus far to make a computer into a mind will have had no relevance to the task of AI other than to disprove its own methods. The fact that the mind is a machine just as much as anything else in the universe is a machine tells us nothing interesting about the mind. If the strong AI project is to be redefined as the task of duplicating the mind at a very low level, it may indeed prove possible—but the result will be something far short of the original goal of AI.

If we achieve artificial intelligence without really understanding anything about intelligence itself—without separating it into layers, decomposing it into modules and subsystems—then we will have no idea how to control it. We will not be able to shape it, improve upon it, or apply it to anything useful. Without having understood and replicated specific mental properties on their own terms, we will not be able to abstract them away—or, as the transhumanists hope, to digitize abilities and experiences, and thus upload and download them in order to transfer our consciousnesses into virtual worlds and enter into the minds of others. If the future of artificial intelligence is based on the notion that the mind is really *not* a computer system, then this must be acknowledged as a radical rejection of the project thus far. It is a future in which the goal of creating intelligence artificially may succeed, but the grandest aspirations of the AI project will fade into obscurity.