ABACUS

THE MAGAZINE FOR THE COMPUTER PROFESSIONAL

VOL 3 NO 4 SUMMER 1986

PHILOSOPHY, AI, AND THE CHINESE ROOM

STRING-PROCESSING LANGUAGES

SPREADSHEETS

SPRINGER-VERLAG NEW YORK BERLIN HEIDELBERG TOKYO $15.95

CONTENTS

VOL 3 NO 4 SUMMER 1986

EDITORIAL

ARTICLES

DEPARTMENTS

REPORTS FROM CORRESPONDENTS

COVER PICTURE. This illustration by Ed Soyka represents a thought experiment proposed by John Searle in his attack on artificial intelligence. A man unfamiliar with the Chinese language is locked in a room alone. Someone passes him a slip of paper with a Chinese question on it, and with the help of an elaborate algorithm supplied to him in English, he is able to write a completely convincing Chinese answer. Can he be said to *understand* Chinese? Searle says no; but in the article beginning on page 7, William Rapaport offers an opposing view.

Philosophy, Artificial Intelligence, and the Chinese-Room Argument

Using the theory of abstract data types to contradict John Searle's argument.

by William J. Rapaport

The theory of artificial intelligence (AI) has been an object of criticism since its earliest days. It has been attacked by philosophers, as in the book *What Computers Can't Do*, by the phenomenologist† Hubert Dreyfus. Even within the AI community itself, there have been cautionary remarks from Joseph Weizenbaum, in *Computer Power and Human Reason*. (See the quotations in the box on the next page.)

Because I am a philosopher by training, as well as a practicing member of what Weizenbaum has called the "artificial intelligentsia," I see this criticism as being on the whole a good thing. In the words attributed to Socrates in Plato's dialogue *Apology*, "The unexamined life is not worth living." It is equally true that in the realm of science, the unchallenged theory is not worth pursuing—or defending. Indeed, as the philosopher of science Karl Popper might express it, a theory that cannot be challenged is not even worthy of being considered scientific.

Philosophers have always been gadflies, making nuisances of themselves and subjecting theories to criticism—constructive or destructive. But they do this to clarify the theories in the hope of arriving at the truth. This criticism may make the path to the truth harder to follow, and the destination is often found to be further away than expected. But philosophical criticism forces scientists and others to think hard about their theories in order, one hopes, to make them better.

The Chinese-Room Argument

In 1980, John R. Searle, a philosopher at the University of California at Berkeley, published an attack on AI in the prestigious journal *The Brain and Behavioral Sciences*. This journal practices what it calls "open peer commentary"; consequently, Searle's article was accompanied by critiques from twenty-eight psychologists, philosophers, neurophysiologists, computer scientists, and cognitive scientists, followed by Searle's replies to each of them. A cottage industry exploring, defending, and attacking Searle's arguments has since developed in professional journals and in the popular press; most recently, Searle has presented his views as the 1984 BBC Reith Lecturer (published as *Minds, Brains and Science*).

Searle's argument is based on a thought experiment that has come to be called the Chinese-Room Argument (see panel on page 9). In this experiment, Searle, who knows neither written nor spoken Chinese, asks you to imagine that he is locked in a room and supplied with an elaborate algorithm (written in English) that tells him

William J. Rapaport is an assistant professor of computer science and a member of the Graduate Group in Cognitive Science at SUNY Buffalo. He received a Ph.D. in philosophy from Indiana University in 1976, and was an associate professor of philosophy at SUNY Fredonia before taking an M.S. in computer science and joining the Buffalo faculty. He has published articles in philosophy of mind, philosophy of language, and computational linguistics, and is coauthor of the text, *Logic: A Computer Approach* (McGraw-Hill, 1985). He is currently doing research on the logical foundations of belief representation, under a grant from NSF.

† Throughout this text, terms marked with a dagger are defined in the glossary on pages 16–17.

Attacks on AI

From phenomenological philosophy:

"Artificial intelligence is a field in which the rhetorical presentation of results often substitutes for research. . . ."

"Descriptive or phenomenological evidence . . . suggests that nonprogrammable human capacities are involved in all forms of intelligent behavior. Moreover, . . . no contrary empirical evidence stands up to methodological scrutiny. Thus, insofar as the question whether artificial intelligence is possible is an empirical question, the answer seems to be that further significant progress in Cognitive Simulation or in Artificial Intelligence is extremely unlikely."

"Is an exhaustive analysis of human reason into rule-governed operations on discrete, determinate, context-free elements [that is, symbol manipulation] possible? Is an approximation to this goal of artificial reason even probable? The answer to both these questions appears to be, No."

—Hubert L. Dreyfus,
What Computers Can't Do

From within the AI community:

"The deepest and most grandiose fantasy that motivates work on artificial intelligence . . . is nothing less than to build a machine on the model of man, a robot that is to have its childhood, to learn language as a child does, to gain its knowledge of the world by sensing the world through its own organs, and ultimately to contemplate the whole domain of human thought. . . . An entirely too simplistic notion of intelligence has dominated both popular and scientific thought, and . . . this notion is, in part, responsible for permitting artificial intelligence's perverse grand fantasy to grow."

"The achievements of the artificial intelligentsia are mainly triumphs of technique. They have contributed little either to cognitive psychology or to practical problem solving."

—Joseph Weizenbaum,
Computer Power and Human Reason

how to write Chinese characters in response to other Chinese characters. Native Chinese speakers are stationed outside the room, and pass pieces of paper with questions written in Chinese characters into the room. Though these symbols are meaningless to him, Searle uses them as input and—following only the algorithm—produces, as output, answers written in Chinese characters. He passes these back outside to the native speakers, who find his answers "absolutely indistinguishable from those of native Chinese speakers." On this basis, Searle argues (in a 1982 article, "The Myth of the Computer"):

> [I] still don't understand a word of Chinese and neither does any other digital computer because all the computer has is what I have: a formal program that attaches no meaning, interpretation, or content to any of the symbols. . . . No formal program by itself is sufficient for understanding. . . .

Syntax vs. Semantics. The Chinese-Room Argument is a variation on Turing's Imitation Game (more commonly known as the Turing Test; see the panel on the facing page). According to this test, we should be willing to say that a machine can think if it can fool us into believing that it is a human. If the Chinese-language program passes the Turing Test, then it *does* understand Chinese. And indeed it does pass the test, according to the very criteria Searle sets up. So how can Searle conclude that it doesn't understand Chinese? One explanation he offers is that the program doesn't understand because it doesn't "know" what the words and sentences *mean*. As Searle puts it in his Reith lectures,

> The reason that no computer program can ever be a mind is simply that a computer program is only syntactical, and minds are more than syntactical. Minds are semantical, in the sense that they have more than a formal structure, they have a content.

That is, meaning—"semantics"†—is something over and above mere symbol manipulation—"syntax."† Meaning is a relation between symbols and the things in the world they are supposed to represent or be about. The creation of "aboutness," or *intentionality*†, is often cited by philosophers as a feature that only minds possess. So, if AI programs cannot exhibit intentionality, they cannot be said to think or understand in any way.

But there are a variety of ways to provide the links between a program's symbols and things in the world. One way is by means of sensor and effector organs: one of my colleagues, AI researcher Stuart C. Shapiro, once suggested that all that is needed is a camera and a pointing finger. If the computer running the Chinese-language program (plus image-processing and robotic-manipulation programs) can "see" and "point" to what it's talking about, then surely it has all it needs to "attach meaning" to its symbols.

Searle calls this response to his argument "the Robot Reply." He objects that if he were processing all of this new information along with the Chinese-language program, he still wouldn't "know what's going on," because now he would just have more symbols to manipulate: he still would have

Searle's Chinese-Room Thought Experiment

Source: John Searle, "Minds, Brains, and Programs," *Behavioral and Brain Sciences* 3 (1980): 417–8. © 1980 by Cambridge University Press; reprinted with permission.

Suppose that I'm locked in a room and given a large batch of Chinese writing. Suppose furthermore . . . that I know no Chinese. . . . To me, Chinese writing is just so many meaningless squiggles. Now suppose further that after this first batch of Chinese writing I am given a second batch of Chinese script together with a set of rules for correlating the second batch with the first batch. The rules are in English, and I understand these rules as well as any other native speaker of English. They enable me to correlate one set of formal symbols with another set of formal symbols, and all that "formal" means here is that I can identify the symbols entirely by their shapes. Now suppose that I am given a third batch of Chinese symbols together with some instructions, again in English, that enable me to correlate elements of this third batch with the first two batches, and these rules instruct me how to give back certain Chinese symbols with certain sorts of shapes in response to certain sorts of shapes given me in the third batch. Unknown to me, the people who are giving me all of these symbols call the first batch "a script," they call the second batch a "story," and they call the third batch "questions." Furthermore, they call the symbols I give them back in response to the third batch "answers to the questions," and the set of rules in English that they give me, they call "the program." . . . Imagine that these people also give me stories in English, which I understand, and they then ask me questions in English about these stories, and I give them back answers in English. Suppose also that after a while I get so good at following the instructions for manipulating the Chinese symbols and the programmers get so good at writing the programs that from the external point of view—that is, from the point of view of somebody outside the room in which I am locked—my answers to the questions are absolutely indistinguishable from those of native Chinese speakers. . . . Let us also suppose that my answers to the English questions are . . . indistinguishable from those of other native English speakers. . . . From the external point of view—from the point of view of someone reading my "answers"—the answers to the Chinese questions and the English questions are equally good. But in the Chinese case, unlike the English case, I produce the answers by manipulating uninterpreted formal symbols. As far as the Chinese is concerned, I simply behave like a computer; I perform computational operations on formally specified elements. For the purposes of the Chinese, I am simply an instantiation of the computer program.

Now the claims made by strong AI are that the programmed computer understands the stories and that the program in some sense explains human understanding. . . .

It seems to me quite obvious in the example that I do not understand a word of the Chinese stories. I have inputs and outputs that are indistinguishable from those of the native Chinese speaker, and I can have any formal program you like, but I still understand nothing. For the same reasons, . . . [a] computer understands nothing of any stories. . . .

We can see that the computer and its program do not provide sufficient conditions of understanding since the computer and the program are functioning, and there is no understanding. But does it even provide a necessary condition . . .? One of the claims made by the supporters of strong AI is that when I understand a story in English, what I am doing is exactly the same . . . as what I was doing in manipulating the Chinese symbols. . . . I have not demonstrated that this claim is false. . . . As long as the program is defined in terms of computational operations on purely formally defined elements, what the example suggests is that these by themselves have no interesting connection with understanding. . . . Whatever purely formal principles you put into the computer, they will not be sufficient for understanding, since a human will be able to follow the formal principles without understanding anything.

no direct access to the external world. Indeed, some thinkers see this lack of access as universal; the philosophical term for this view is *solipsism.*† Solipsism is generally regarded as a self-defeating notion: it can't be refuted, but it seems useless as a description of the way the world is.

But there is another way to provide the link between symbols and things in the world: even if the system has sensor and effector organs, it must still have *internal representations* of the external objects, and the relations between these and its other symbols constitute meaning for *it.* (The box on page 11 provides a formalism for such representations.) Searle's view seems to be that semantics must link the internal symbols with the outside world, and that this is something that cannot be programmed. But if this is what semantics must do, this has to apply for human beings, too, and we might as well wonder how the link could possibly be forged for us. Either the link between internal representations and the outside world *can* be made for both humans and computers, or else semantics is more usefully treated as linking one set of internal symbolic representations with another. On this view, semantics does indeed turn out to be just more symbol manipulation. Views like this have been called *methodological solipsism.*† Methodological solipsism need not be regarded as self-defeating, since it doesn't deny the existence of, or access to, an outside world; it merely ignores the outside world where it's not needed.

Here is Searle's objection to the Robot Reply:

I see no reason in principle why we

Turing's Imitation Game

Source: A. M. Turing, "Computing Machinery and Intelligence," *Mind* 59 (1950).

I propose to consider the question "Can machines think?" This should begin with definitions of the meaning of the terms "machine" and "think." The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous. If the meaning of the words "machine" and "think" are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, "Can machines think?" is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and expressed in relatively unambiguous words.

The new form of the problem can be described in terms of a game which we call the "imitation game." It is played with three people, a man (A), a woman (B), and an interrogator (C) who may be of either sex. The interrogator stays in a room apart from the other two. The object of the game for the interrogator is to determine which of the other two is the man and which is the woman. He knows them by labels X and Y, and at the end of the game he says either "X is A and Y is B" or "X is B and Y is A." The interrogator is allowed to put questions to A and B thus:

C: Will X please tell me the length of his or her hair?

Now suppose X is actually A, then A must answer. It is A's object in the game to try to cause C to make the wrong identification. His answer might therefore be

"My hair is shingled, and the longest strands are about nine inches long."

In order that tones of voice may not help the interrogator the answers should be written, or better still, typewritten. The ideal arrangement is to have a teleprinter communicating between the two rooms. Alternatively the question and answers can be repeated by an intermediary. The object of the game for the third player (B) is to help the interrogator. The best strategy for her is probably to give truthful answers. She can add such things as "I am the woman, don't listen to him!" to her answers, but it will avail nothing as the man can make similar remarks.

We now ask the question, "What will happen when a machine takes the part of A in this game?" Will the interrogator decide wrongly as often when the game is played like this as he does when the game is played between a man and a woman? These questions replace our original, "Can machines think?"

NOTE: *The question, "Can machines think?" is "replaced" by these new questions because there is no clear way to answer the former, whereas Turing's Imitation Game* can *be played, and thus there is a clear way to answer the new questions. —WJR*

couldn't give a machine the capacity to understand English or Chinese, since in an important sense our bodies with our brains are precisely such machines. But . . . we could not give such a thing to a machine . . . [whose] operation . . . is defined solely in terms of computational processes over formally defined elements.

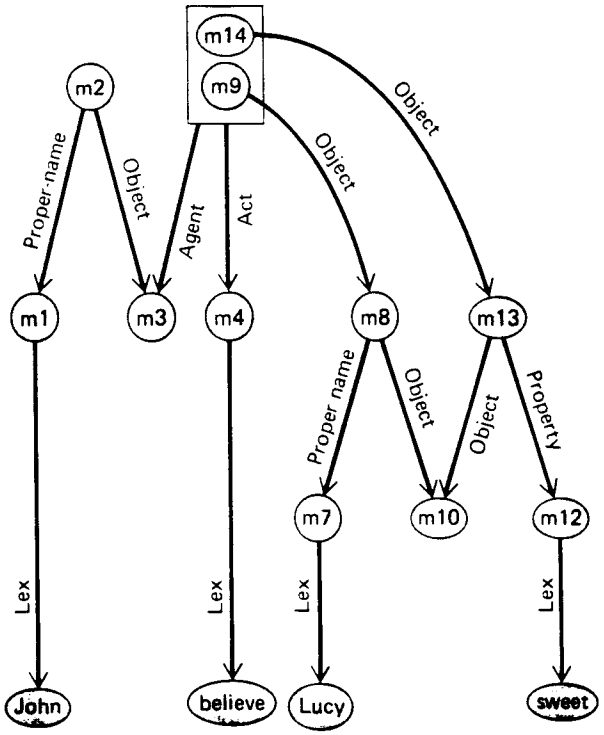
"Computational processes over formally defined elements" is a more precise phrase for symbol manipulation. In claiming that a machine that just manipulates symbols cannot understand a natural language, Searle reasons that "only something having the same causal powers[†] as brains can have intentionality." What, then, are these "causal powers"? All Searle tells us is that they are due to the (human) brain's "biological (that is, chemical and physical) structure." But he does not specify precisely what these causal powers are, and this is the biggest gap in his argument.

Causation and Realization. In his book, *Intentionality*, Searle tells us a bit more. He says that "mental states are both *caused by* the operations of the brain and *realized in* the structure of the brain." As I shall explain below, a careful analysis of these two notions reveals that the requisite "causal powers" are not really causal at all. What's more, Searle's claim about mental states is misleading because he does not distinguish "abstract" mental states from "implemented" ones.

To put it simply, it is misleading to say that we are dealing with only one kind of thing—mental states—and that they "are both *caused by* the operations of the brain and *realized in* the structure of the brain." Rather, it is better to recognize that there are actually two kinds of things: *implemented* mental states, which are caused by the operations of the brain, and *abstract* mental states, which are realized in the structure of the brain. An implemented mental state bears the same relationship to an abstract mental state as an implemented data structure bears to an abstract data type. But I am getting ahead of myself.

It is important to bear in mind Searle's essentially biological stance. In "The Myth of the Computer," he says that even a simulated human brain "made entirely of old beer cans . . . rigged up to levers and powered by windmills" would not really exhibit intentionality, whatever it might appear to do. Thus, the age-old philosophical distinction between appearance and reality is at the heart of the issue: Do machines that *appear* to think—or that *simulate* thinking—*really* think? Searle contends that the answer is no, while I believe the answer is yes: intentional phenomena are among a class of phenomena for

Interconnecting Networks of Meanings



One method for giving meaning to internal symbols without direct links to the external world is by using semantic networks. For instance, beliefs are prime examples of mental states because of their intentionality or "aboutness": beliefs are always about something "outside" of themselves—they have an object, typically a proposition. Thus, John's belief that Lucy is sweet is about Lucy's being sweet.

This can be represented in a semantic network formalism—as in the figure at left, where the formalism used is that of Stuart C. Shapiro's SNePS semantic network processing system. Node *m14* represents the fact that John believes that Lucy is sweet; the OBJECT arc points to the system's representation of the object of John's belief. The object of John's belief is represented by node *m13*. Its meaning is represented by the arcs that point to the node representing Lucy (*m10*) and the node representing the property of being sweet (*m12*).

By itself, of course, there is no reason to think that *m12* represents being sweet rather than, say, being rich—as Searle would be quick to point out. But in a full natural-language-understanding system, each of these nodes would be connected to many others, some of which might be direct representations of external stimuli. For instance, node *m12* might be one of these, with the node labeled "sweet" being the actual property of sweetness. But the internal meaning of any node would merely be its location and connections in the entire network.

which appearance *is* reality. In other words, Searle claims that computers might be said to "think" in a weak sense at most, while I hold that they are capable of thinking in a strong sense. (See the panel on the next page, *Strong and Weak AI*.)

My theory, in rough outline, is this. Consider Searle's beer-can-and-windmill simulation of a human brain, programmed to simulate thirst. Searle says that it is *not* thirsty. We might reply that perhaps it feels *simulated* thirst; and we might then go on to wonder if simulated thirst *is* thirst. But an even better reply is to say that it *simulates* the feeling of *simulated* thirst. Similarly, the Chinese computer system simulates the understanding of simulated Chinese. But, says my theory, the simulated feeling of simulated thirst *is* thirst, and simulated understanding *is* understanding. The differences between the artificial thing and the "real" thing—or, more to the point, the human thing—lie in their physical make-up. The thirst that is implemented in the beer-can-and-windmill computer may not "feel" the way that thirst implemented in a human feels, but they are both *thirst*. And similarly for understanding.

Data Abstraction and Implementation

I said before that abstract mental states were like abstract data types. What, then, is the theory of abstract data types, and how can it be used to counter Searle's argument?

Computer programs can be thought of as describing *actions* to be performed on *objects*. The actions are expressed in terms of the operations of the programming language used, and the objects are represented by *data structures*, each of which must be constructed out of the data structures available in that programming language.

Data structures can be classified into different data *types*. An *abstract data type* is a formal (that is, a mathematical or abstract) data structure, together with various characteristic operations that can be performed on it. An *implementation* of an abstract data type is usually an actual data structure in a program, that *plays the role* of the abstract data type. This characterization is admittedly rough, but it will do for now. The notion of role-playing will also prove to be useful later on.

An example should help. A *stack* is an abstract data type consisting of potentially infinitely many items of information (data) arranged or "structured" in such a way that new items are added only to the "top" of the stack, and

Strong and Weak AI

The claim of "strong AI" is—in Searle's words—that "the appropriately programmed computer really *is* a mind, in the sense that computers given the right programs can be literally said to *understand*." "Weak AI," on the other hand, sees the computer only as a "very powerful tool" for studying the mind. Searle thinks that strong AI "has the consequence that there is nothing essentially biological about the human mind," and that therefore there is something wrong with strong AI. But one of the advantages of looking at the problem through the lenses of the computational theory of abstract data types is that we will be able to see that this is *not* what strong AI says. Instead, I would say that, while the physical implementing medium of humans does turn out to be essential to *human* minds and *human* intelligence, the (different) physical medium of computers is just as essential to *computer* minds and *computer* intelligence. Thus, both human and computers can (or will be able to) be said to be intelligent. Or at least to understand Chinese!

an item can be retrieved only if it is on the top. Picture a stack of cafeteria trays—the last item put on the stack is the first to come off. The programming language Pascal does not have stacks as a built-in data type, but they can be implemented in Pascal by arrays, which *are* built in. Unlike a stack, an array is structured so that an item can be added to or removed from any of its cells. But if one steadfastly refuses to do that and always treats an array in a last-in/first-out manner, then it is, for all practical purposes, a stack. Indeed, real stacks of cafeteria trays are more like arrays than stacks, since people often take a tray from the middle. (See Figure 1.)

The specific implementation is to the abstraction as the individual is to the species—but this analogy is flawed.

Aristotle's Philosophy and Abstract Data Types. Sometimes, the relation between an abstract data type and an implementation of it reminds people of a comparison that dates back to Aristotle's time, one that in its modern form can be found in taxonomic biology: the relation between a genus or species (such as humankind) and an individual (such as Searle, or you, or I) belonging to that genus or species. According to Aristotle, an individual has *essential* or "defining" properties—namely, those of the genus or species to which it belongs. Its nonessential or *accidental* properties are those that differentiate it from other individuals of its species. So, for example, I am essentially a rational animal (as Aristotle might have put it), but only accidentally a computer scientist; that differentiates me from, say, Aristotle or Searle, who are also essentially rational animals but who are not computer scientists.

So we have an analogy: the specific implementation is to the abstraction as the individual is to the species. However, this analogy is flawed. Having a top is essential to a stack, but not to an array that implements it. And two implementations of a stack can differ in essential ways (not merely in accidental ways); arrays are essentially different from linked lists, yet both can implement stacks. And stacks are essentially different from queues, yet arrays can implement them both.

These flaws in the analogy arise from several facts. Not all properties of an abstract data type need be "inherited" by an implementation. Nor does the abstract data type need to have all the essential properties of the device that implements it. For instance, an abstract stack can be infinite or dynamically sized; arrays in Pascal are finite and of fixed size. Arrays that implement stacks *can* be accessed in the middle, even if they shouldn't be; stacks cannot. (Modern programming languages that are designed to take advantage of data abstraction, such as Ada or CLU, make the notion of implementation more like that of Aristotelian individuation.) Finally, one abstract data type can implement another. For example, the abstract data type *sequence* can be implemented by the abstract data type *linked list*, which, in turn, can be implemented by *symbolic expressions* in Lisp. Symbolic expressions can be thought of either as a "real" implementation, or as yet another abstract data type ultimately to be implemented by electronic signals in a computer. And (though there's no special reason to do it) stacks and, say, trees could probably be used to implement each other!

Some Applications. Since these notions are important for understanding what's wrong with Searle's theory, a few applications of them to other areas may prove helpful. In doing this, I shall be likening certain things to abstract data types, but I don't think that all of them *are* abstract data types in the computer-science sense (although some are). For convenience, I shall call them "Abstractions."

• When you listen to music on a record, are you *really* listening to music or merely to a recording—a simulation—of music? Clearly both, because the reproduction of music *is* music. A musical score is an Abstraction to be implemented by, say, an orchestra. Perhaps it is more like a computer program than an abstract data type. But programs, too, are Abstractions that can be implemented in different media. The relations between an abstract data type and an implementation of it are precisely those between a score and a performance of it. Even the "implementation" of an abstract data type by the mathematical notation used to describe it is paralleled by the relation between the abstract idea of the score and its printed copy.

• Different collections of water molecules and collections of alcohol molecules are implementations of the Abstraction *Liquid*. (I shall return to this example later.)

• Two recent newsworthy controversies can be resolved using this analytical tool:

1. Can trees communicate with each other? Apparently, some trees that are being attacked by insects will produce defensive chemicals that are "sensed" by other trees, inducing them to produce the defensive chemical, too. There was much speculation in the press as to whether this was a form of communication in the way that human language is. What most of the participants in the debate seem to have missed is that there were not merely two concepts to compare—human and arboreal communication—but three: there might be common features that could be called the Abstraction *Communication*, of which the human and arboreal forms can be seen as implementations.

2. Is sport a religion? Some writers believe that it is; others see the two as essentially different—even more different than arrays and stacks. Yet perhaps there is an Abstraction—call it *Religion*—of which both sport and the more familiar religions are both implementations.

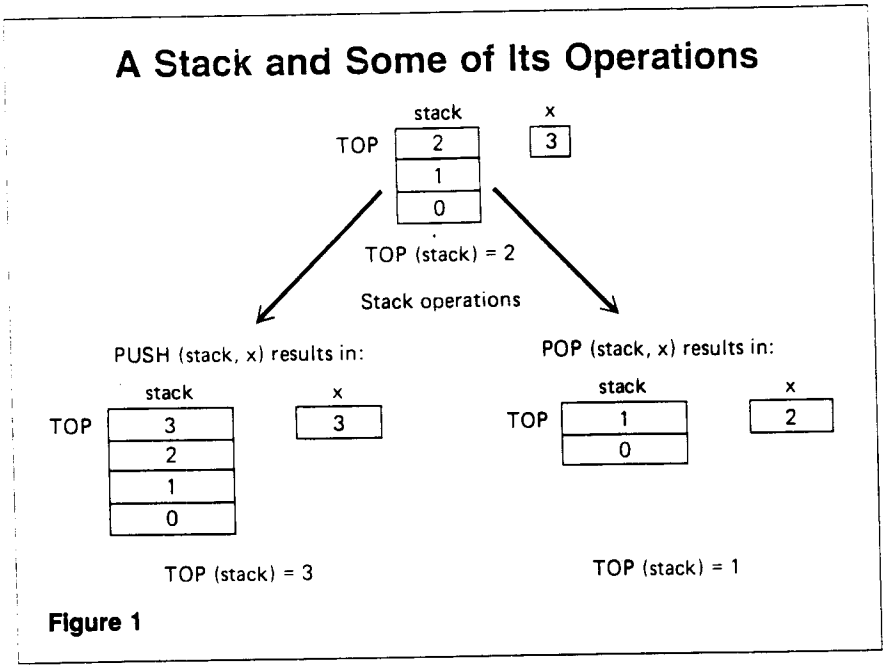


Figure 1

• Douglas Hofstadter, in an article on the Turing Test in his book, *The Mind's I*, discusses whether a weather-prediction program that operates by simulating a hurricane produces an actual hurricane. One might think, "Of course not—a simulated hurricane doesn't destroy houses or get you wet." But if the simulation is complete, it might include simulated houses and simulated people. And while they might not actually get destroyed or actually get wet, surely they will suffer *simulations* of destruction and wetness. As James Gleick recently expressed it in an article on weather prediction in *The New York Times Magazine*,

> The computer models are really little universes, their air, water, mountains, coastlines and—most important—their natural laws distilled into an essence that can be manipulated electronically. The substance of the model is the substance of the real world, more or less intact, but translated into mathematical equations. The model contains equations that behave very much like wind.

In other words, the Abstraction *Hurricane* can be implemented in a computer as well as in the real world. The computer program can be considered both an *implementation* of the Abstraction (Hurricane) and a *simulation* of the earthly hurricane. The two kinds of Hurricane can be distinguished by differences between the implementing media, and it is the computer hurricane that enables us to learn about the earthly one. But they are both Hurricanes.

Perhaps mental phenomena, like abstract data types, are Abstractions that can be implemented in different media.

• Finally, of most relevance to the issues Searle raises, perhaps

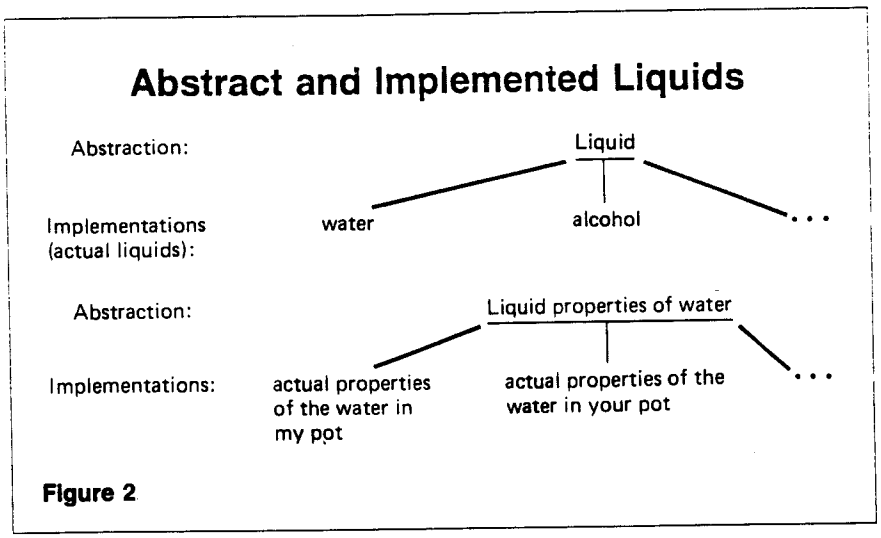


Figure 2

mental phenomena, like abstract data types, are Abstractions that can be implemented in different media—say, human brains as well as electronic computers.

Abstract versus Implemented Mental States

When Searle says that "mental states are . . . *realized* in the structure of the brain," he should really be saying that *abstract* mental states are *implemented* in brain structure. Remember, Searle argues that a computer running the Chinese program does not understand Chinese because only human brains can understand in the appropriate way. According to him, understanding, even if describable in a computer program, is biological; hence, an electronic computer cannot understand. On the other hand, I am arguing that there can be an abstract notion of understanding—in fact, a computational one—that can be implemented in computers as well as in human brains; hence, both can understand. I gain my leverage over Searle by making a distinction between an Abstraction and its implementations—a distinction that is implicit in his theory but that he fails to make.

The distinction also makes it evident that Searle and most of his adversaries have been talking at cross-purposes. Just as an array does not implement a stack in the same way that a linked list does, so a computer that implements the understanding of Chinese need not do so in precisely the way that a human does. In other words, the Abstraction *Understanding* does not have to capture the "essence" of human understanding; it only has to be something that both humans and machines can implement and in virtue of which they are able to understand, each in their own way. Nevertheless, they both understand.

The Abstraction* Understanding *only has to be something that both humans and machines can implement and in virtue of which they are able to understand.

The distinction that Searle does not make explicit can be discerned in the following passage from *Intentionality*:

> Mental states are as real as any *other* biological phenomena, as real as lactation, photosynthesis, mitosis, or digestion. Like these other phenomena, mental states are caused by biological phenomena and in turn cause other biological phenomena. [Italics mine.]

This passage suggests that by "mental states" Searle means *implementations of abstract mental states*. I shall refer to these as *implemented mental states*.

Searle's arguments make more sense when understood in this way. Implemented mental states *would* be caused by the brain, just as an implemented stack (that is, an actual stack in an actual program)—but not an abstract stack—can be said to be caused by an array. Indeed, the only way a mental state could cause and be caused by other biological phenomena is if it were a biologically implemented mental state.

But this distinction between the Abstraction and the implementation also holds for the other biological phenomena that Searle listed. Lactation, thirst, and so forth, can be given abstract definitions and then implemented in nonbiological or exobiological stuff—even suitable beer-can-and-windmill contraptions.

To make the distinction a bit clearer, consider again the example of liquids. A *Liquid*, considered as an Abstraction, can be realized (implemented) in a collection of molecules. But what that collection causes is *actual* (or implemented) liquidity. Similarly, the Abstraction *Liquid Properties of Water* can be realized (implemented) in different collections of water molecules, but the actual liquid properties of particular samples of water are caused by different actual behaviors. For instance, the boiling of the water in *my* pot is caused by the behavior of the water molecules in my pot, whereas the boiling of the water in *your* pot is caused by the behavior of the molecules in yours. Moreover, the Abstraction *Liquid* can certainly be realized

(implemented) in different collections of molecules (water molecules, alcohol molecules, etc.), and the actual liquidity of these molecules is caused by different actual behaviors. Yet all are liquids, not because of any *causal* relationship, but because of the *realizability* relationship. (See Figure 2.)

So, too, the Abstraction *Understanding* (or *Understanding Chinese*) can be realized (implemented) in both humans and computers. *Actual* understanding can be caused by both humans and computers: because of the realizability relationship, both are Understanding.

When an Abstraction is implemented in some physical device, the implemented entity is sometimes said to *be* the physical device "under a description," or it is sometimes said that the device is playing a certain role. These are useful metaphors. In Woody Allen's film, *The Purple Rose of Cairo*, Tom Baxter is a character in a film-within-the-film; in that context, the role of Baxter is played by a fictional character—a second-rate actor named Gil Shepherd. Shepherd "implements" Baxter. (And Shepherd, in turn, was "implemented" by the real actor, Jeff Daniels!) But Baxter-as-played-by-Shepherd is a third entity; there is Baxter-the-character, who could have been played by anyone; Shepherd-playing-the-role-of-Baxter; and Shepherd himself. In the case of mental states in general, and the Chinese Room in particular, there are three corresponding things: *Understanding Chinese* (the Abstraction), the computer that understands Chinese, and the computer itself. We shall explore the relations between these below.

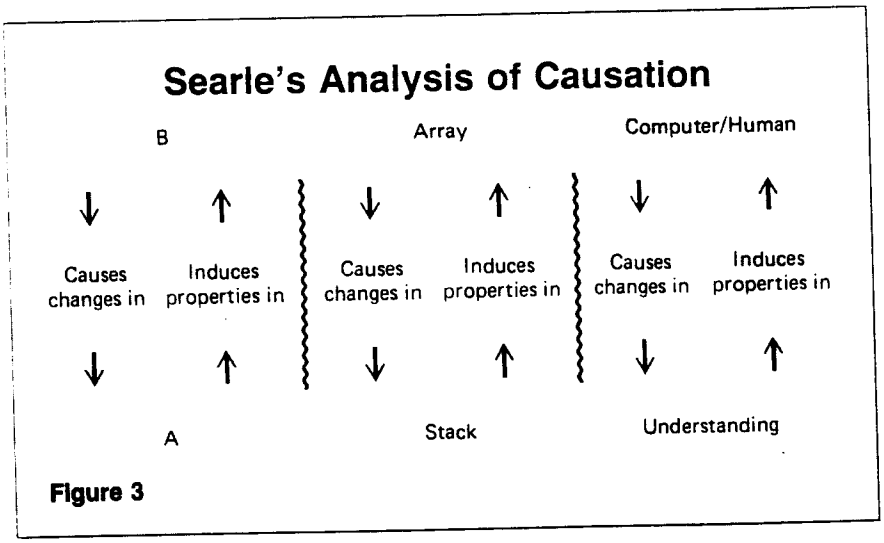


Figure 3

Searlean Causation

First, recall that Searle also said that "mental states are . . . *caused* by the operations of the brain." It is time to find out exactly what he means by "causation," remembering that he does not distinguish between an Abstraction and its implementation.

According to Searle, if *A* is caused by *B*, then changes in *B* must cause changes in *A* and properties of *A* must induce properties in *B*. (See Figure 3.) For example, the liquid properties of water are "caused" by molecular behavior; therefore, heating water yields steam, and the liquidity of water causes it to be pourable.

Consider another example. Suppose that stacks are caused by arrays. Do changes in arrays cause changes in actual, implemented stacks? What would count as a relevant change? Perhaps this: changing the contents of the implemented stack by using array operations rather than stack operations would surely change the stack. It is precisely such changes that the full theory of abstract data types tries to rule out in practice. Do properties of abstract stacks induce properties in arrays? Yes: since new elements can only be added to the top of a stack, it follows that new elements can (or should) only be added to the "top" of its implementing array.

What about understanding? Surely, changes in computers or humans can cause changes in their understanding: physical changes (such as the addition of parallel processing in the case of computers, or education in the case of humans) would improve or impair their ability to understand. And properties of understanding Chinese induce in an "implementing" computer or human the ability to converse with a native speaker.

Machine understanding and human understanding are both instances of the more abstract, computational characterization of understanding.

Searle offers an analogy, and it is here that the full, three-dimensional complexity of the theory reveals itself (though he sees only two dimensions). In *Intentionality*, Searle characterizes realization in this way:

> The liquid properties of the water . . . are *realized* in the collection of molecules. . . . When we describe . . . [water molecules] as liquid we are

just describing those very molecules at a higher level of description than that of the individual molecules.

Similarly, given a program that implements stacks as arrays, we can describe the program at a low level as having arrays and at a higher level as having stacks. Looked at "bottom up," there are only arrays: looked at "top down," there are (only) stacks.

For the Chinese-Room thought experiment, these ideas may be summarized as follows. Machine understanding is, indeed, understanding, just as human understanding is; they are both instances of the more abstract, computational characterization of understanding. Machine understanding is "caused" by a computer (or computer program) in which abstract understanding is realized. But this sense of "cause" is not the ordinary, physical causation that is involved, say, when one billiard ball hits another, causing it to move, or when one neuron causes another to fire. Rather, it is the sort of "noncausal causation" that (according to philosopher Jaegwon Kim) is involved when a sibling's marriage "causes" you to become an in-law. Or perhaps it is what another philosopher, Hector-Neri Castañeda, has described as *consubstantiation:* a relation, somewhat like the relation of being coextensive†, holding between intensional entities—entities that we can conceive of as being distinct even though they are physically identical; the standard example is the morning star and the evening star. Here, the intensional entities that are "consubstantiated" are the actual, implemented understanding and the behavior of the computer executing the program—that is, the computer itself and the computer considered as "the thing that understands Chinese." A simpler way of saying this is: machine understanding is an instance of abstract understanding, which is realized in a computer.

Glossary

The definitions in this glossary are only rough ones, and philosophically-minded readers may disagree with many of them. Indeed, each of these terms has been the subject of a great deal of philosophical analysis. One of the principal methods of philosophical analysis is to examine carefully what authors mean by the terms that they use. Frequently, the result of such an analysis is the discovery that a term is being used inconsistently.

causal powers The physical ability to produce changes in the properties of an object (usually in a lawlike way).

coextensive entities Two entities that have all the same properties:

Extensional entities are those whose "identity conditions" (the conditions for deciding when two of them are really the "same") do not depend on the way in which they are represented or described. Alternatively, they may be characterized as those entities satisfying the following rough principle: Two extensional entities are equivalent (for some purpose) if and only if they are identical (that is, if and only if "they" are really one entity, not two). For example, the following are extensional: physical objects; sentences; truth values; and mathematical objects such as sets, functions defined in terms of their input/output behavior (that is, as sets of ordered pairs), and *n*-place relations defined in terms of sets of ordered *n*-tuples.

Intensional entities are those whose identity conditions do depend on the way in which they are represented or described. Alternatively, they are those entities that satisfy the following rough principle: Two intensional entities might be equivalent (for some purpose) without being identical (that is, they might really be two, not one). For example, the following are intensional: concepts; propositions; properties; algorithms; and objects of thought, including fictional entities (such as Sherlock Holmes), nonexistents (such as the present King of France), and impossible objects (such as the largest cardinal number or a round square). The objects of mental states and processes (which are "intentional"—see definition below) are also intensional, whereas physical objects are always extensional.

intentionality The property of mental states and processes (as opposed to purely physical ones) that they are always "directed" to an object: when you think, you always

The Biological Character of Intentionality

There is one quesion left over: Why does Searle think that intentionality must be biological? Because, he says, only biological systems have the *requisite causal properties* to produce intentionality. And what are these causal properties? According to Searle, they are those properties that are "causally capable of producing perception, action, understanding, learning, *and other intentional phenomena*" (my italics). Now, first of all, this begs the question, rather than answering it. All it says is that intentionality must be biological because only biological systems can produce it. But why should it be that *only* biological systems can produce it? And secondly, we have seen that what he really means when he says that only biological systems "produce" intentionality is that only they can *realize* intentionality—that only biological systems can implement that collection of Abstractions. Most importantly, the theory presented here shows that "the requi-

think *about* something; when you believe, you always believe *that* something is the case. These objects of mental states and processes need not exist or be true; you can think about Santa Claus, and you can believe that he is skinny.

methodological solipsism A solipsistic theory that does not deny the existence of the external world, but claims that it is not necessary to study the interaction between a mind and the world in order to understand the mind.

phenomenology A technique of philosophical inquiry into the objects of human mental experience, whose chief method is an examination of mental states and processes "bracketed off" from their connections to other (especially scientific) activities. Nevertheless, many phenomenologists hold that AI will fail because the human mind can only be fully understood as being "situated" in the world.

Phenomenology's chief rival techniques—those of *analytic philosophy*—have tended to focus on formal, piecemeal studies of logic and language (hence the term "analytic"). Analytic philosophers have tended to be more sympathetic to the goals of AI, perhaps because computer programs can be seen as a formal, analytical tool.

semantics The study of linguistic meaning, and of the relationships between language and the world.

solipsism At one extreme, a theory that I am the only thing that exists (and that you are a figment of my imagination). A more moderate version holds that I have no direct access to things in the external world (the world "outside" myself)—if it exists.

syntax The study of grammar and of the formal rules for manipulating symbols.

site causal powers" are not really causal at all; they are simply the ability to realize a species of an Abstraction. "Causality" is a red herring. But Searle has offered us no reason to believe that intentionality, abstractly conceived, could *not* be implemented in a beer-can-and-windmill device.

Conclusion

Computer scientists and AI researchers owe Searle a debt of gratitude, because his objections make us think harder about our theories. It is also important for computer scientists to realize that theories like data abstraction are the intellectual descendants of well-understood theories with good philosophical pedigrees. And it is important for AI researchers to be forced to articulate their claims, so that they can be reassured that their research is on the right track. Most importantly, perhaps, philosophers and scientists (and computer scientists in particular) need to realize that they have a lot to say to each other.

For Further Reading

Churchland, Paul M. *Matter and Consciousness: A Contemporary Introduction to the Philosophy of Mind.* Cambridge, MA: MIT Press, 1984.

An excellent survey of philosophical theories of the mind.

Dreyfus, Hubert L. *What Computers Can't Do: The Limits of Artificial Intelligence.* Revised edition. New York: Harper & Row, 1979.

A classic argument against AI from a phenomenological point of view.

Hofstadter, Douglas R., and Dennett, D. C., eds. *The Mind's I: Fantasies and Reflections on Self and Soul.* New York: Basic Books, 1981.

A fascinating collection of essays, fiction, and commentary on the nature of the mind, edited by a computer scientist and a philosopher.

John R. Searle's writings on the Chinese-Room Argument:

- "Minds, Brains, and Programs." *Behavioral and Brain Sciences* 3 (1980): 417–57.
- "Analytic Philosophy and Mental Phenomena." *Midwest Studies in Philosophy* 6 (1981): 405–23.
- "The Myth of the Computer." *New York Review of Books* (29 April 1982): 3–6. Cf. correspondence, same journal (24 June 1982): 56–7.
- *Intentionality: An Essay in the Philosophy of Mind.* Cambridge: Cambridge University Press, 1983.
- *Minds, Brains and Science.* Cambridge, MA: Harvard University Press, 1984.

Turing, A. M. "Computing Machinery and Intelligence." *Mind* 59 (1950). Reprinted in A. R. Anderson, ed. *Minds and Machines.* Englewood Cliffs, NJ: Prentice-Hall, 1964: 4–30.

Turing's classic article on his "Imitation Game," now known as the "Turing Test."

Weizenbaum, Joseph. *Computer Power and Human Reason: From Judgment to Calculation.* San Francisco: W. H. Freeman, 1976.

A wide-ranging essay on the nature of computers and their role in society, by the author of the well-known ELIZA program.

The Id and Ego of Programming Teams

I am appalled by the article "Programming Teams" by Henry Ledgard in the Spring, 1986 issue of ABACUS.

The thrust of the article is that the key to successful programming teams is the principle of subordination of the individual to the collective. Professor Ledgard stresses this theme throughout his article. The idea is introduced gently, at first, with such statements as:

> Each crew member learns and accepts that a collective goal transcends individual effort. (p. 8)

> Team players do what is necessary for the project to succeed, even if it does not appear compatible with their own individual objectives. (p. 9)

Later, the idea is stated more boldly:

> Individual egos must be sacrificed for something higher: a collective ego. (p. 13)

> Precocious programmers and copilots have to adopt a stance that is counter to normal human behavior—trading the personal for the collective ego. (p. 13)

Professor Ledgard's notion that an ideal programming team is characterized by subordination of the individual to the collective goal grows out of his attempt to draw an analogy between programming teams and aerial combat crews. But, this analogy is not valid. He does not seem to realize the significance of the fact that there are two fundamentally different classes of team goals.

Survival teams, such as trained combat or rescue teams, focus on the relief of emergency situations. Their goal is to make their reactions to situations automatic and mechanical, while minimizing freedom or creativity on the part of the team members. Survival depends on the predictability of the reactions of one's teammates.

The focus of production teams, such as programming teams, is quite different. Their aim is the creation of a product, ideally, the best product which their minds can create. In such a team, the creativity rather than predictability of the individual should be valued most highly. Thus, programming teams should not behave like combat teams, and vice versa.

The implication of Professor Ledgard's assertion that the collective ego is somehow "higher" than the egos of its members is that the individual is morally bound to sacrifice his or her own ego for the sake of the collective. Sacrifice of this nature is both immoral and impractical. A team can only be successful when its individual members each take personal pride in his or her own work. "Pride cannot be achieved by sacrificing personal goals, principles, standards or values—the personal ego—to the collective." So long as the rewards sought by individuals are only those that are earned, the common goals of the team and the goals of its individuals need not conflict.

Jack H. Schwartz
Senior Staff Engineer
Bendix Computer-Aided Engineering Center
Teterboro, New Jersey

Author's Response:

Perhaps I did not make the point clear in the article, or perhaps there is a large communication problem. Let me say this:

- *Egoless programming is not (ultimately) egoless. Exposing one's work to praise and criticism has a single goal—to make the work better. If the work produced is of better quality, certainly the personal ego is reinforced.*
- *Being egoless is not counter to human nature. When the project succeeds, everyone succeeds.*

Ultimately, the collective ego and the personal ego go hand in hand.

Henry Ledgard

Can Machines Understand?

Rapaport's article on whether machines can "understand" [Summer 1986] is informative and entertaining, but his refutation of Searle's position is more complex than necessary. Searle's argument that machines cannot understand can be simply refuted, without the use of abstract data types, if we start from the following three propositions stated by Searle himself to be the core of his argument:

1. Programs (and machines) are purely syntactic.
2. Syntax is neither equivalent to nor sufficient by itself for "semantics".
3. Minds have mental "semantic" contents.

These three propositions imply

that programs and the machines on which they run cannot possess semantic understanding. But they are not only fuzzy because "semantics" is ill-defined but are in fact inconsistent.

It is hard to conceive of a notion of semantics that is both contained in minds as required by proposition 3 and not syntactic as required by proposition 2. Searle's argument is refuted by the fact that no "reasonable" notion of semantics satisfies his three propositions.

Although the notion of data abstraction is thus not needed to refute Searle's arguments, it is useful in explaining Turing's affirmative position that machines can in fact be said to understand. Had the concept of data abstraction been current when Turing wrote his Turing-test paper, he might well have used the concept as a basis for defining his model. Turing's thesis is based on a behavioral view that defines thinking and understanding in terms of behavioral abstraction and views implementations by brains and computers as equally satisfactory.

The abstract data-type argument is unlikely to convince Searle because Searle clearly rejects the behavioral model of what it means to think and understand. He believes that understanding is an intensional property that cannot be modelled solely by behavior. In order to address Searle's concern we must dig deeper than simple behaviorism and examine the kind of intension that Searle would admit as understanding. It is here that Searle is on weak ground because he does not wish to make his criterion so narrow as to be purely biological or so broad as to include purely syntactic mechanisms. The question is whether an acceptable notion of intensionality exists that is broader than purely biological but excludes purely syntactic sets of rules. This is essentially the same dilemma as that of finding an acceptable definition of semantics satisfying Searle's propositions 2 and 3. But the position that intensionality must be purely biological violates both technical usage of the term in disciplines like mathematics and informal usage in linguistics and cognitive science. In contrast, syntactic and, therefore, computer-implementable definitions of semantics are quite common and fall within the spectrum of acceptable meanings associated with this elusive term.

One of the clearest arguments for the usefulness of syntactic definitions of semantics is object-oriented programming, which supports an "intensional" style of problem specification in which objects in the real world are modelled by syntactic objects of the programming language. Objects such as tables are represented by data abstractions in a way that even Searle might admit as intensional. The essence of object-oriented programming is the modelling of real-world objects by finite sets of (syntactic) attributes.

But there is in fact considerable evidence that modelling by finite sets of attributes cannot provide a "complete" characterization of real-world objects. If by understanding we mean "complete understanding" then this limitation of object-oriented modelling techniques provides a stronger foundation than arguments based on the mystical or biological nature of intensionality for the position that machines cannot understand.

The idea of modelling tablehood by a finite set of necessary and sufficient attributes is attractive but not practicable, since tables need not have four legs or even a horizontal surface. In fact, it is difficult to identify even a single necessary attribute of tables, much less a set of necessary and sufficient attributes. Philosophers call concepts which cannot be characterized by a finite set of attributes "natural kinds". The existence of natural kinds is evidence against the adequacy of abstract data types as models of mental concepts and consequently of the brain.

However, even if we cannot model tablehood completely by its attributes we can approximate the concept by increasingly good approximations. This applies also to the modelling of abstract concepts like intelligence and understanding. In principle we should be able to approximate both individual concepts and the brain by finite sets of attributes comprising an abstract data type. But in practice this task is often combinatorially intractable. Thus, although Searle's argument is wrong and Rapaport is right in refuting it, Searle's position may well be right and Rapaport may therefore have won the battle but lost the war.

Of course, it can be argued that complete understanding is not possible for humans either, and that both computers and humans use approximate models in performing complex intellectual tasks like understanding Chinese. This reduces the question whether machines can understand to a matter of opinion, with criteria for deciding that are behavioral (Turing-like) rather than intensional (Searle-like). If we accept that both human and computer understanding may be imperfect then it appears that Turing rather than Searle has won the day.

In summary, Searle's position that machines cannot understand because intensionality and semantics are biological is based on too narrow a view of what it means to understand. Those of us who disagree with his argument but agree with his conclusion can justify our position by appealing to the inherent impossibility of modelling real-world objects by finite sets of attributes. But this position may again be overturned by asserting that object-oriented models are imperfect in a way that is similar to the imperfection of mental models.

The bottom line is that we can answer the question either way, depending on our interpretation of the term "understanding". But the affirmative position seems much more exciting as a starting point for constructive research than the negative position. Thus we opt for the affirmative position for pragmatic reasons rather than because it can be logically proved. Turing's test should be viewed as a pragmatic challenge rather than as a metaphysical statement concerning the nature of thinking or understanding. In answering a metaphysical question like "Can Machines Think?" it is more important to answer it in a manner that is useful than to juggle the meaning of fuzzy concepts to prove its truth or falsity.

Dr. Peter Wegner
Department of Computer Science
Brown University
Providence, Rhode Island

Author's Response:

Wegner says that my argument is "more complex than necessary." Perhaps; but many people tend to think that any *philosophical argument is more complex than necessary. Searle's argument, however, is subtle and requires subtle responses.*

Having said that, let me also say that I am sympathetic to Wegner's refutation of Searle, but I think that it's only half the story; the other half is the anti-biological half, which was the central thrust of my article. But even on the anti-semantic half, I differ from Wegner in several details.

First, Wegner thinks that 1–3 are inconsistent. I prefer to argue merely that 2, on a certain reading of "semantics," is false: semantic understanding of natural language only requires a "knowledge base" and a mechanism for manipulating it, but all of that is merely syntactic.

Second, although Turing's model of understanding was behavioral, the abstract-data-type approach is not, or at least not in the same sense. The abstract-data-type would go into more detail about the intermediate processes that produce the input-output behavior.

Third, I don't quite understand Wegner's use of the term "intensional," especially in connection with object-oriented programming. For one thing, there are other AI techniques—for instance, knowledge representation—that model real-world objects by syntactic objects; so I see nothing special about the object-oriented approach. For another, one way that a knowledge-representation system can be "intensional" is by modeling objects that don't exist (but that can be thought or spoken about) or by modeling objects in a fine-grained way—for instance, distinguishing between two different "viewpoints" or mental concepts of a single real-world object (such as the Queen of England as distinguished from the Queen of Canada). But this has nothing to do with the syntax/semantics issue.

Finally, Wegner says that "modelling by finite sets of attributes cannot provide a 'complete' characterization of real-world objects." But it's not clear that humans *have such complete characterizations. The point is: what are we modeling? The real world? A cognitive agent's mental model of the real world? It is the latter that is needed for (computational) natural-language understanding, so completeness is a red herring. Moreover, such finite modeling does* not *have to be modeling by necessary and sufficient conditions, especially if the model can change as beliefs are revised (as Wegner seems to admit when he talks of "increasingly good approximations"). As for the combinatorial intractability of this, if humans only use finite models (and I think that they must), then if it's not computationally intractable for* us, *it couldn't be for a computer.*

William J. Rapaport

Intelligence: Pigs, Persons, and Computers

Regarding "Philosophy, Artificial Intelligence, and the Chinese-Room Argument" (ABACUS, Summer 1986): Suppose that I am locked in a room and grunting like a pig. From outside the room I am absolutely indistinguishable from a pig. Using William J. Rapaport's ideas, the pig and I are both implementations of the same abstract notion of intelligence. Nearly everything can be made an implementation of this abstraction using the argument presented in the article.

If William J. Rapaport's argument is used for defining intelligence, it is no longer necessary to prove that computers are intelligent. The task is rather shifted; you now have to disprove that you are a computer even if you can simulate one.

Thomas Pinegger
Passau, West Germany

Mythrepresenting Ada

EDITOR'S NOTE: *In our Fall 1986 issue, Velma and Harry Huskey reviewed a disparaging biography of Ada, the Countess of Lovelace, who has generally been credited with furthering the work of Charles Babbage. What follows is a rebuttal from the book's author. It has been substantially condensed in the interest of space.*

One of the pleasant consequences of publishing a book is the num-

Continued on page 66