COMPUTER PROCESSES AND VIRTUAL PERSONS: Comments on Cole's "Artificial Intelligence and Personal Identity"

William J. Rapaport

Department of Computer Science
and Center for Cognitive Science
State University of New York at Buffalo
Buffalo, NY 14260
rapaport@cs.buffalo.edu

Abstract

This is a draft of the written version of comments on a paper by David Cole, presented orally at the American Philosophical Association Central Division meeting in New Orleans, 27 April 1990. Following the written comments are 2 appendices: One contains a letter to Cole updating these comments. The other is the handout from the oral presentation.

In general, I am sympathetic to Cole's arguments; my comments seek to clarify and extend the issues. Specifically, I argue that, in Searle's celebrated Chinese Room Argument, Searle-in-the-room does understand Chinese, in spite of his claims to the contrary. He does this in the sense that he is executing a computer "process" that can be said to understand Chinese. (The argument that the process in fact does understand Chinese is made elsewhere; here, I merely assume that *if* anything understands Chinese, it is a "process" executed by Searle-in-the-room.) I also show, by drawing an analogy between the way that I add numbers in my head and the way that a calculator adds numbers, that Searle-in-the-room's claim that he does not understand Chinese does not contradict the fact that, by executing the Chinese-natural-language-understanding algorithm, he does understand Chinese.

1 COLE'S ARGUMENT.

David Cole (1990) claims that, in the Chinese Room Argument (Searle 1980, 1982, 1984), "Searle has ... succeeded in proving that no computer will ever understand ... any ... natural language" (1).[1] Cole's emphasis, however, is not on it being the case that the computer cannot *understand*, but on it being the case that it is the *computer* that cannot understand, for he goes on to say that "this is consistent with the computer's causing a new entity to exist (a) that is not identical with the computer, but (b) that exists solely in virtue of the machine's computational activity, and (c) that does understand English" [*sic*; I assume he meant to say: (c) that does understand *Chinese*] (1). The new entity is a "virtual person", to be thought of along the lines of what computer scientists call a "virtual machine" (1-2). Cole says that "showing that the machine itself does not understand does not show that nothing does" (1). Later, he says that "from the fact that *someone* [viz., Searle in the room][2] does not understand Chinese it does not follow that *no one* understands Chinese" (3).

Throughout, Cole assumes that there *is* understanding of Chinese (or, of natural language) going on in the Chinese Room. I have argued elsewhere that indeed something does understand Chinese (Rapaport 1985, 1986ab, 1988ab). My arguments for this center around two claims: (1) The program being executed *implements* the abstract data type of a Chinese-understander. (2) Understanding *can* be purely syntactic. But I shall not repeat those arguments here.

[1] Page references are to Cole's manuscript unless otherwise indicated.

[2] I shall use 'Searle-in-the-room' to refer to the person in the Chinese Room who follows the natural-language-processing algorithm. I shall use 'Searle' to refer to John Searle, the author of the Chinese Room Argument.

My comments will focus on Cole's claims (1) that it is not Searle-in-the-room or the computer that understands natural language and (2) that what understands natural language is a "virtual person", where a virtual person, according to Cole, is what computer scientists call a "virtual machine". In particular, I shall argue (a) that what understands natural language is, not a virtual machine, but what computer scientists call a "process", and (b) that, nevertheless, there is a sense in which Searle-in-the-room does understand Chinese, in spite of his (Searle-in-the-room's) claim to the contrary. That is, *if the virtual person understands natural language, so does the computer and so does Searle-in-the-room!*

2 THE KORNESE ROOM.

Cole's first argument for the claim that it is not Searle-in-the-room who understands Chinese concerns the "Kornese" Room (3-5): Imagine a very large AI/natural-language-processing program, as described in Searle's original Chinese Room Argument, for "understanding" Chinese (more precisely, for modeling a cognitive agent, with a distinctive personality, who understands Chinese) *as well as* for "understanding" Korean. (In order to be a bit more definite, we might imagine two generalized augmented-transition-network (ATN) parser-generators that interface with a knowledge-representation and reasoning system (cf. Shapiro 1982, Rapaport 1988b) such that if the input is determined to be in Chinese, a Chinese sub-ATN is jumped to from the starting node of the ATN, and if the input is determined to be in Korean, a Korean sub-ATN is jumped to.) Suppose that Searle-in-the-room executes this program. Suppose that there is no information flow between the two fragments of the program (so neither "knows" about the other; for instance, except for a common starting node, the two branches of the ATN would be entirely separate, and we would need the two knowledge bases constructed during the course of the conversation also to be kept separate). Finally, "suppose that the behavioral evidence is as clear as can be ... that there are two *distinct* individuals in the room" (5). Let P_c be the Chinese-"understanding" individual; let P_k be the Korean-"understanding" individual.

Now, I have no quarrel with the plausibility of this extension of Searle's Chinese Room. Indeed, the SNePS/CASSIE research projects of the SNePS Research Group and the Center for Cognitive Science at SUNY Buffalo consist in the design of just such cognitive agents as are in the Chinese Room. Designing such *dual* cognitive agents (as are in the Kornese Room) seems to me an interesting research project.

Cole needs to show that P_c is not Searle-in-the-room; i.e., he needs to show that it is possible that Searle-in-the-room can fail to understand Chinese while something other than Searle-in-the-room does understand Chinese. His argument (5-6) is as follows (letting 'S' stand for Searle-in-the-room):

1. $P_c \neq P_k$	: assumed in the description of the Kornese Room (4-5)
2. $P_c = S$ iff $P_k = S$	: assumed in the description of the Kornese Room (5)
3. $(P_c = S \;\&\; P_k = S) \vee (P_c \neq S \;\&\; P_k \neq S)$	: from 2, by propositional logic
4. $\neg(P_c = S \;\&\; P_k = S)$	: else $P_c = P_k$ (by transitivity of '='), contra 1
5. $P_c \neq S \;\&\; P_k \neq S$	: from 2, 3, by propositional logic
6. $P_c \neq S$	: from 5, by propositional logic

This argument is valid. But, I claim, it is unsound. Or perhaps I should say that it is not expressed properly. For '=' is the wrong relation between P_c and S (or P_k and S). The relation *is*, as Cole argues, that between a virtual machine (roughly, P_c) and a machine (namely, S) that implements it. But implementation is not identity. (In fact, I believe that implementation is a *semantic* relation, but that, too, is a story for another day.) And, I shall claim, S *can* implement *both* P_c *and* P_k without it being the case that $P_c = P_k$.

Cole's second argument for the claim that it is not Searle-in-the-room who understands Chinese is that "the *same* mind [i.e., the same P_c] could have been realized by the activity of someone other than Searle—Searle could even lose his job in the Room and be replaced by another—while the Chinese conversation continues" (9). I certainly agree with Cole's premise. But what I think follows from it is not that $S \neq P_c$ or that Searle-in-the-room fails to understand Chinese, but—again—that '=' is not the relationship between S and P_c. The relationship is one of implementation: S *implements* P_c, and, of course, implementation is precisely such that some $S' \neq S$ can *also* implement P_c.

Cole also argues that it is not the *computer* that understands either Chinese or Korean: "it would be a mistake to attribute ... [understanding Chinese] to the computer itself" (6): If one asks the Kornese Room in Korean whether it understands Chinese, it will reply that it doesn't; and, presumably (though Cole does not say so), if one asks it in Chinese whether it understands Chinese, it will reply that it does. But does it follow that the *computer* does not understand Chinese? I think not: I see nothing incompatible with the computer understanding Chinese but not *understanding that it understands Chinese*.

To see how this might be the case, consider a student in an introductory computer-science course who is given a complicated Turing-machine algorithm that, unknown to her, computes the greatest common divisor (GCD) of two integers. Suppose that the student, as an exercise in using Turing machines, is given pairs of integers and asked to follow the algorithm. Suppose further (though, no doubt, this is bad pedagogy!) that she is not told that what she is doing is computing the GCD of the inputs and that she (alas!) does not even know what a GCD is. Surely, she *is* computing GCDs, though she does not understand that she is doing so. (Or consider the *Korean* Room (introduced in Rapaport 1988b, not to be confused with the *Kornese* Room), in which a benighted Korean scholar waxes eloquently about plays that are in fact Korean translations of Shakespeare, not knowing that they are translations—he understands Shakespeare, but does not understand that he understands Shakespeare. Or consider the calculator example, discussed in Section 6, below.)

Cole says that in the Kornese Room, "no *single* entity understands both Chinese and Korean; there are two subjects, virtual persons: one who understands Chinese and one who understands Korean. These two virtual subjects are realized by a single substratum, the computer" (6). However, by my arguments, the computer understands both languages, but it does not "know" that it understands both. (Actually, it's not quite right to say that it is the computer that understands both; see below.) Now, *if* the computer could share information between its two modes of operation, it should eventually come to realize that it understands both. For instance, if the two knowledge bases shared information such as visual information, then it's highly likely that the system would infer what was happening. Or, for instance, suppose that our student who is using a Turing machine to compute GCDs learns in her math class what GCDs are and is given a *different* algorithm in that class for computing them. Eventually, she should be able to realize that the Turing-machine algorithm is also a GCD algorithm. Of course, in the Kornese Room, if there is no possibility of shared information, then we have a "schizophrenic" mind; for all practical purposes, there would be two minds. (Or, rather, there would be *at most* two minds, for Cole's argument does not deal with whether *anything* is understanding Chinese or Korean. For that, of course, we need other arguments, such as the two I adverted to above.)

So, assume that there is understanding of Chinese going on in the Chinese Room. If it is not Searle-in-the-room who understands it, then there must be something else that does. Call it a "virtual person". Similarly, in the Kornese Room, "there are two subjects, virtual persons: one who understands Chinese and one who understands Korean" (6). What is a virtual person? It is "a mind *realized* by" Searle-in-the-room (9). Cole appears to claim that this virtual person is the *virtual machine* defined by the natural-language-understanding program that Searle-in-the-room executes. What, then, is a virtual machine?

3 PROCESSES AND VIRTUAL MACHINES.

According to Andrew S. Tanenbaum's standard text (1976), "Each machine [i.e., each computer] has some machine language ... . In effect, a machine defines a [computer programming] language. Similarly, a language defines a machine—namely, the machine that can execute all programs written in that language" (Tanenbaum 1976: 3-4). There can be a sequence of virtual machines, each implemented in the next. So, a virtual machine is an abstract machine, defined by a language and implemented in either another virtual machine or (ultimately) an actual (physical) machine. A program, P, written in language L can be executed by an L-machine, M_L. If M_L is a virtual machine, then P is "executed" by M_L only in an abstract sense; ultimately, P is executed by some physical machine.

At this point, we need another technical notion: "A *process* ... is ... a program in execution. It is an active entity, capable of causing events to happen. ... A process is in contrast to a program, which is a passive entity" (Tanenbaum 1976: 12). A process is to a program as a live animal is to "a dead animal, or model of an animal" (Tanenbaum 1976: 12). As I have argued elsewhere (and I believe that Cole would agree (1-2)),[3] if anything *understands* Chinese, it is not a Chinese-understanding *program*, but a *process*—the Chinese-understanding program being executed (Rapaport 1988b).

Before pursuing this further, let's return briefly to the Kornese Room. There we have one program but, apparently, two virtual persons. Let us agree (at least for the sake of the argument) that the program/computer doesn't understand *both* languages. The notion of a process can clarify this. First, arguably there are *two* programs, not one, even if there is only one piece of code. Since there is no information flow, the two fragments of "the" program could be teased apart. Second, even if there really is only one program, there are two separate *processes*: A process consists of a (static) program (a piece of text) and a (dynamic) "state vector" (containing information about which instruction of the program is to be executed next, the current values of the variables, etc.; cf. Tanenbaum 1976: 12). The state vector can be teased apart into two separate ones *because of the lack of information flow*. So, it's *not* "the program" or "the computer" that understands natural language (if anything does) or that computes GCDs. Rather, it's a *process* that understands natural language (if anything does) or that computes GCDs. Similarly, it's not the human brain that understands natural language (or that computes GCDs); rather, it's a process (implemented) in the brain that does so.

So we have the following situation: A given computer, M_1, can really only execute instructions in its machine language, ML. We can write a program, P_L, in a "higher-level" (or, at least, easier) language, L, which is translated into ML; the translation is then executed on M_1. (This can be done either by "compiling" the L-program, P_L, into an ML-program, P_{ML}, which is then executed on M_1, or by "interpreting" each instruction of P_L, step by step, into (usually several) input-output–equivalent instructions in ML and executing them before interpreting the next P_L-instruction.) Alternatively, we could have a physical computer M_2 whose machine language *is* L, and we could execute P_L directly on M_2.

A third alternative is to "imagine the existence of a hypothetical computer or *virtual machine* whose machine language is" L (Tanenbaum 1976: 2). This is a level of abstraction: Rather than thinking of writing L-programs that are translated into ML and run on M_1, we can think more abstractly of writing L-programs to be run directly on A hypothetical M_2. This sequence of abstractions can be continued: To take an example from our own research, the computational cognitive agent (or virtual person, if you will) that we call 'CASSIE' (cf. Shapiro & Rapaport 1987 and forthcoming; Rapaport 1988b and forthcoming) is essentially a program written in the SNePS knowledge-representation and reasoning system; SNePS is written in Lisp; the Lisp program is either executed directly on a physical TI Explorer Lisp machine or else is translated into a C program; the C program is translated into machine language for an Encore Multimax (another physical computer). So, CASSIE is a virtual machine with respect to SNePS, which is a virtual machine with respect to Lisp, which is either a virtual machine with respect to the physical Explorer or is a virtual machine with respect to C, which is a virtual machine with respect to Multimax machine language, which is a virtual machine with respect to the physical Multimax.

The *process* corresponding to an L-program P_L (such as the SNePS-program CASSIE) that is being executed "exists" abstractly in the L-machine (the virtual SNePS machine), but it exists actually in the bottom-level physical machine (either an Explorer or a Multimax).

So, if anything understands natural language, it's the CASSIE *process*. Is it also the *SNePS* process? (And, ultimately, the *Multimax* process?) Can we say that *CASSIE* understands natural language (if she does) but that *SNePS* does not? (Or that CASSIE does, but that the Multimax does not?)

4 WHO UNDERSTANDS?

As Yorick Wilks (1984: 109-110) has pointed out, the software/hardware distinction is a matter of *convention*, because any software can be hardwired. That is, any virtual machine can be built as a physical machine (subject only, perhaps, to engineering ingenuity). That is, the virtual-machine/physical-machine distinction

[3] For even clearer indications of agreement, cf. the revised version (dated 6-6-89) of Cole's paper, pp. 11, 17.

is a matter of *convention*. But this means that the notion of a virtual person as a solution to the puzzle of the Chinese Room gets us nowhere as an explanation of *who* (if anything) understands Chinese. (It will, however, help explain how Searle-in-the-room can both understand and not understand Chinese! But more on this below.) Why does the notion of a virtual person not help? For one thing, even though it is a process, not a machine, that does the understanding—that is, even though Cole's notion of a virtual person is more properly to be identified with a *process*, not with a virtual machine—nevertheless, the virtual machine that executes the program, resulting in the process, is the *embodiment or locus* of the understanding.

(Elsewhere, I have characterized a process as like being an actor (read "machine") playing (read "executing") a role in a play (read "program") (Rapaport 1987, 1988a). Of course, this raises the objection that an actor who, playing the role of Hamlet, "kills Polonius" does not actually *kill Polonius* or *kill the actor who is playing Polonius*. But the actor playing Hamlet *does* "talk to Polonius" and *is* actually talking to the actor who is playing Polonius. Similarly, if the actor playing Hamlet "kisses Ophelia", then he actually *kisses* Ophelia; i.e., he actually kisses the actor who is playing Ophelia. As with many other cases, sometimes a simulation of X *is* an X (cf. Rapaport 1988ab; Shapiro & Rapaport, forthcoming.))

Another answer to the question of why the notion of a virtual person doesn't help is that, as I hope to make clear shortly, *all* levels are executing the "same" process: That is, given a program P in language L, which is implemented in language L', which is implemented in ... which is implemented in machine language ML, which is implemented in machine M, *all* of them (working simultaneously) are doing what program P does. But there *is* a level that is the "main" one and which is such that *it* is the primary locus of explanation and description, and which can be re-implemented elsewhere. That level is the program P (it is CASSIE in our example, or the Chinese-understanding program in Searle's case; cf. Dennett's (1978) theory of the intentional stance). The bottom (or physical) level does what the top-level program does *because* the top level does it; it is the top level that "drives" the bottom level (just as it is the bottom level that implements the top level).

Now, *why* do I say that all levels are executing the "same" process? Consider a GCD algorithm, written in Lisp. Does the Lisp process compute GCDs? Let's assume that it does (this is fairly, but not entirely, uncontroversial).[4] But suppose that the Lisp program is compiled into C. Does the C process compute GCDs? Absolutely: I could walk away with the C program and execute *it* to compute GCDs without ever knowing that the C program was produced from a Lisp program (compiling is a form of automatic programming). And similarly down to the machine-language level. Of course, *I* might not understand how (or what) the C program (let alone the machine-language program) does if that's all I have: Nothing about compilers requires that the compiled program be well-structured, suitably modular, and adequately self-commenting. It is (only?) the higher-level abstractions in the Lisp program that can best show me how and why the program works. Similarly for CASSIE: it's the program at *her* level of organization and abstraction that enables *me* to say what she's doing.

Strictly speaking, however, it is not the *same* process that all levels are executing (hence the "scare quotes" on 'same', above). This is because, according to Tanenbaum's definition, a process consists of a program and a state vector. If there are two different programs (two different pieces of text), then, *a fortiori*, there are two different processes. (Moreover, the variables referred to in the two state vectors will, in general, be different.) Nevertheless, just as there is a virtual person, namely, P_c, implemented in a virtual machine, so is there a Chinese-understanding virtual person implemented by Searle-in-the-room. But whether or not this Chinese-understanding virtual person is different from P_c, it *is* a Chinese-understanding process.

So, although I am sympathetic to Cole's claim that it is a "virtual person" that understands Chinese (or Korean, or whatever) *if* anything does, I would not only go further and assert that it *does* understand Chinese (or Korean, or whatever) (as I have argued in Rapaport 1988b) but that the machines—*all* of them, all the way down to the physical one—that implement the virtual person *also* understand Chinese (or Korean, or whatever). That is, Searle-in-the-room is the embodiment or locus of the understanding. But this needs a bit more clarification.

[4] What I have in mind here is the alleged need of a human to interpret the program's input and output.

5 UNIVERSAL TURING MACHINES.

First, Cole's example of a virtual machine is too simplistic. He talks of "emulation software [that] may make a computer built using an Intel 8088 processor behave as though it were a Z80" (7). But the issues are more general than virtual machines. Let T_G be a Turing-machine program that takes as input two integers, a and b, and that outputs their GCD, g: $T_G(a, b) = g$. Now let U be a universal Turing machine that takes three items of input: a description (D_{T_G}) of T_G, a, and b, and that outputs g; so, $U(D_{T_G}, a, b) = g$. What does U do? One thing is quite clear: U computes GCDs; U does what T_G does. Or is it more accurate to say that U (merely) runs a (virtual) Turing machine (namely, T_G), which, in turn, computes GCDs? Now, one of the major points about universal Turing machines is precisely *that* they can compute any computable function. *How* they do it is irrelevant to *what* they do: U might, indeed, implement the virtual T_G; i.e., U might run T_G on input (a, b). But U might also look at D_{T_G} to determine that T_G computes GCDs and then use *its own* GCD-algorithm to compute them. (Or—though I have some doubts about this, having to do with the Halting Problem[5]—U might use its own inductive-inference algorithm to learn what T_G does as well as how it does it, and then write its own algorithm to mimic T_G's behavior.) Or, as John Case (personal communication) has suggested, U "might just run mysterious code (MC) which happens to compute gcd, but the MC could be so obscure that one cannot prove in *ZF* that it does the job. Furthermore, one may not be able to prove in *ZF* that the machine is universal even though it is ...". Etc.

So, *what* the Multimax (or Explorer, or Intel 8088, or whatever) (i.e., the Multimax that is executing a program that creates a virtual machine that executes a program that ... that creates CASSIE, who understands natural language)—what that Multimax is doing is ... understanding natural language.

6 TWO WAYS OF ADDING.

Second, let me turn to some further considerations on this theme that, I think, clinch the argument that it is the bottom-level machine (insofar as it is a machine rather than a process running on the machine) that is understanding natural-language. These considerations will also indicate how it is possible for the machine (or for Searle-in-the-room) both to understand Chinese and not to understand Chinese.

Most of us lack clear intuitions about how computers work or about universal Turing machines. So let me give some more mundane examples of analogous situations. First, consider a hand-puppet. A hand-puppet without my hand in it is like a program; with my hand animating it, it's like a process. Suppose that, putting my hand in the puppet, I cause it to pick up a piece of chalk and write a numeral on a blackboard. Did *I* write the numeral? It seems to me that I did, though indirectly (but not much *more* indirectly than had I been wearing a glove).

Next, consider a marionette. Suppose that, by manipulating its strings, I cause the marionette to pick up the chalk and write the numeral. Did *I* write the numeral? Again, it seems to me that I did, although in a roundabout way (and not quite as clearly as in the hand-puppet case)..

Suppose, next, that I press some buttons that cause a robot to pick up the chalk and write the numeral. Here, clearly, I did *two* things: (1) I pushed buttons. (2) I also—in a roundabout way, *by* pushing the buttons—wrote the numeral. (Or, to be more cautious about it, I caused the numeral to be written. There are, it would seem, interesting issues to be pursued here in terms of causation and moral responsibility; alas, they will have to be left for another time.)

My final example is, for me, the clearest: When I use my pocket calculator to add two numbers, what am I doing? Am I pushing buttons? Yes, directly. Am I adding two numbers? Yes, indirectly. But—and, here, recall the variety of possible universal Turing machines—note that insofar as I *am* adding, I am *not* doing it in the same way that I do it by hand (using the elementary-school paper-and-pencil algorithm we all know and love) or the way that I do it in my head (there, I use a slightly different "mental arithmetic" algorithm). So, I know (or, I have) at least two *different* and *unrelated* ways to add: (1) using my paper-and-pencil algorithm (or my mental algorithm); (2) pushing buttons in an appropriate manner on a pocket

[5] Thanks to Johan Lammens for pointing this out to me.

calculator. It's irrelevant whether the calculator also uses either of my own algorithms. The important point is that when I add by pushing buttons on a calculator, *I* am *not* using either of those algorithms. The analogy I wish to draw is this: my using (say) my mental algorithm stands in the same relationship to my pushing buttons in order to add on my calculator as Searle-in-the-room's understanding *English* stands to his manipulating the cards with "squiggles" on them to understand *Chinese*. Searle-in-the-room has two different ways to understand language: (1) the way he understands English and (2) the way he understands Chinese. These two ways are unrelated.

Moreover, that Searle-in-the-room can claim that he doesn't understand Chinese is like a claim that I might make that I don't understand what I am doing with my calculator. This seems to me to be a perfectly reasonable claim. That is, someone might ask me to push certain buttons on a calculator and to read off the display. In so doing, I am (let us suppose) adding. But I might not understand that that is what I am doing, and I might even *deny* that that is what I am doing. (This is even clearer if the "calculator" is a high-tech one and my button-pushings cause it to compute some obscure-to-me statistical function.) Yet adding is going on when I push the buttons on my calculator, and *I* am adding, albeit *differently* from how I normally do it, not to mention *indirectly* (by pushing buttons). Note, though, that it is quite possible that the calculator's adding algorithm is precisely the same as my mental algorithm. In that case, the calculator is not adding *differently* from how I normally do it. Rather, there are, then, *two* separate and independent processes going on. In fact, when adding with a calculator, I often check the arithmetic by *simultaneously* doing the addition in my head and comparing my results with the calculator's. This, then, is how Searle-in-the-room can implement both P_c and P_k even though $P_c \neq P_k$: it is as if Searle-in-the-room were using two calculators.

Now this would seem to suggest that it is really the *calculator*, not me, that is adding; I'm just pushing buttons (I'm just providing the causal energy for the calculator). So, to return to the Chinese Room, who understands Chinese? Perhaps, after all, it is *not* Searle-in-the-room, except indirectly and ignorantly. What is it in the Chinese Room that corresponds to the calculator? Suppose that, instead of cards with Chinese "squiggles" on them, Searle-in-the-room has a Chinese-natural-language-understanding calculator: The English program tells Searle-in-the-room which buttons to press. More precisely, let us suppose that the native Chinese speakers outside the Room push buttons that produce a display on Searle-in-the-room's calculator. Searle-in-the-room's English program tells him which buttons to press in reply, given that display. He presses them, producing a new display that the native speakers see. And so on. Now, who (or what) understands Chinese? The calculator *plus its program as causally energized by Searle-in-the-room*, that is, the *process*.

7 CONCLUSION.

So: The native speakers of Chinese are communicating with something that understands Chinese. They are communicating with a *process* (call it a virtual person if you will; I have no quarrels with that)—a process being executed by Searle-in-the-room. Searle-in-the-room can say that he himself does not understand Chinese. That does *not* contradict the fact that, by playing the elaborate Chinese-natural-language-understanding card game, he is also understanding Chinese.

Similarly, when I use a calculator to add two numbers while simultaneously adding the two numbers in my head, I am executing two processes. If I never learned how to add, but were given instructions on how to use the calculator, then I would be adding. Yet I could say that I myself did not know how to add.[6] That would not contradict the fact that, by pushing calculator buttons, I am adding.

Or would it be better to say merely that there is an adding process going on? The question raised here—echoing Bertand Russell's objection to Descartes's *cogito ergo sum*[7]—is: "Where does *I* come from?".[8]

[6] This, no doubt, has ramifications for contemporary mathematics education!

[7] Cf. Russell 1950, as cited in Kenny 1968: 58-59. The VIth set of Objections to Descartes's *Meditations* may also be relevant to the Chinese Room Argument.

[8] With apologies to Mary Galbraith for usurping her phrase.

8 REFERENCES.

1. Cole, David J. (1990), "Artificial Intelligence and Personal Identity," paper presented at the Colloquium on AI, American Philosophical Association Central Division, New Orleans, 27 April 1990.
2. Dennett, Daniel C. (1978), "Intentional Systems," in D. C. Dennett, *Brainstorms* (Montgomery, VT: Bradford Books): 3-22.
3. Kenny, Anthony (1968), *Descartes: A Study of His Philosophy* (New York: Random House).
4. Rapaport, William J. (1985), "Machine Understanding and Data Abstraction in Searle's Chinese Room," *Proceedings of the 7th Annual Conference of the Cognitive Science Society (University of California at Irvine)* (Hillsdale, NJ: Lawrence Erlbaum Associates): 341-345.
5. Rapaport, William J. (1986a), "Searle's Experiments with Thought," *Philosophy of Science* 53: 271-279; preprinted as *Technical Report 216* (Buffalo: SUNY Buffalo Department of Computer Science, 1984).
6. Rapaport, William J. (1986b), "Philosophy, Artificial Intelligence, and the Chinese-Room Argument," *Abacus* 3 (Summer 1986) 6-17; correspondence, *Abacus* 4 (Winter 1987) 6-7, *Abacus* 4 (Spring 1987) 5-7.
7. Rapaport, William J. (1988a), "To Think or Not to Think," *Noûs* 22: 585-609.
8. Rapaport, William J. (1988b), "Syntactic Semantics: Foundations of Computational Natural-Language Understanding," in J. H. Fetzer (ed.) *Aspects of Artificial Intelligence* (Dordrecht, Holland: Kluwer Academic Publishers): 81-131.
9. Rapaport, William J. (forthcoming), "Predication, Fiction, and Artificial Intelligence," *Topoi*.
10. Russell, Bertrand (1950), *History of Western Philosophy* (London: Allen and Unwin).
11. Searle, John R. (1980), "Minds, Brains, and Programs," *Behavioral and Brain Sciences* 3: 417-457.
12. Searle, John R. (1982), "The Myth of the Computer," *New York Review of Books* (29 April 1982): 3-6; cf. correspondence, same journal (24 June 1982): 56-57.
13. Searle, John R. (1984), *Minds, Brains and Science* (Cambridge, MA: Harvard University Press).
14. Shapiro, Stuart C. (1982), "Generalized Augmented Transition Network Grammars for Generation from Semantic Networks," *American Journal of Computational Linguistics* 8: 12-25.
15. Shapiro, Stuart C., & Rapaport, William J. (1987), "SNePS Considered as a Fully Intensional Propositional Semantic Network," in N. Cercone & G. McCalla (eds.), *The Knowledge Frontier: Essays in the Representation of Knowledge* (New York: Springer-Verlag): 262-315; earlier version preprinted as *Technical Report No. 85-15* (Buffalo: SUNY Buffalo Department of Computer Science, 1985); shorter version appeared in *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86, Philadelphia)* (Los Altos, CA: Morgan Kaufmann, 1986): 278-283.
16. Shapiro, Stuart C., & Rapaport, William J. (forthcoming), "Models and Minds: Knowledge Representation for Natural-Language Competence," in R. Cummins & J. Pollock (eds.), *Philosophical AI: Computational Approaches to Reasoning*.
17. Tanenbaum, Andrew S. (1976), *Structured Computer Organization* (Englewood Cliffs, NJ: Prentice-Hall).
18. Wilks, Yorick (1984), "Machines and Consciousness," in C. Hookway (ed.), *Minds, Machines and Evolution* (Cambridge, Eng.: Cambridge University Press): 105-128.