Published in *British Journal for the Philosophy of Science* (2007) 58: 755–776.
doi:10.1093/bjps/axm035

mark.sprevak@ed.ac.uk

# Chinese rooms and program portability

Mark Sprevak
*University of Edinburgh*

4 October 2006

I argue in this paper that there is a mistake in Searle's Chinese room argument that has not received sufficient attention. The mistake stems from Searle's use of the Church–Turing thesis. Searle assumes that the Church–Turing thesis licences the assumption that the Chinese room can run any program. I argue that it does not, and that this assumption is false. A number of possible objections are considered and then rejected. My conclusion is that it is consistent with Searle's argument to hold onto the claim that understanding consists in the running of a program.

## 1 Searle's argument

Searle's Chinese room argument is an argument against the possibility of Strong artificial intelligence (Strong AI). The thesis of Strong AI is that running a particular program is sufficient for, or constitutive of, mentality: it is simply in virtue of running a particular program that a system has mentality. One aspect of mentality is understanding, and this is the aspect on which Searle focuses. Searle appreciates that understanding is a complex notion, so he has a particular form of understanding in mind, namely, the understanding of simple stories. He considers whether the Strong AI thesis holds in this case.

It seems intuitively obvious that when I read a simple story in English, I understand that story: somewhere in my head there is understanding going on. However, if I read a simple story written in Chinese (a language that I do not speak), there is no understanding going on. What makes the difference between these two cases? The

advocate of Strong AI says that the difference consists in my running a particular program in the case of English stories and my failure to run a particular program in the case of Chinese stories. If the program for understanding Chinese stories were given to me, then I would be able to understand Chinese stories. Similarly, if that program were given to any other sufficiently complex system (e.g. a Turing machine), then it too would be able to understand Chinese stories. Searle argues that this consequence of Strong AI—that running a program is sufficient for, or constitutive of, understanding simple Chinese stories—is false. Call this consequence AI.

Searle's argument against AI is as follows. Imagine a monolingual English speaker inside a room with a rule-book and sheets of paper. The rule-book contains instructions in English on what to do if presented with Chinese symbols. The instructions are of the form: 'If you see Chinese symbol *X* on one sheet of paper and Chinese symbol *Y* on another, then write down Chinese symbol *Z* on a third sheet of paper'. Pieces of paper with Chinese writing are passed into the room and the person inside follows the rules and passes pieces of paper out. Chinese speakers outside the room label the sheets that are passed in 'story' and 'questions' respectively, and the sheets that come out 'answers to questions'. Imagine that the rule-book is as sophisticated as you like, and certainly sophisticated enough that the responses that the person gives are indistinguishable from those of a native Chinese speaker. Does the person inside the room thereby understand Chinese? Searle argues that they do not. Some respondents to Searle claim that the person inside the room *can* understand Chinese. However, I wish to grant Searle this key intuition, and claim that his argument fails on other grounds.[1]

Searle notes that the Chinese room is a computer, and he identifies the rule-book with the program that it runs. He then reminds us that the thought experiment does not depend on the particular rule-book used: it does not matter how sophisticated the rule-book, the person inside the room would still only be mindlessly shuffling symbols and not understanding Chinese. Since rule-books are programs, the thought experiment demonstrates that the Chinese room cannot understand Chinese *no matter what program it runs*. The Chinese room is a universal computer, which according to Searle, means that it can run any program. Therefore, we can conclude that *no program* can be constitutive of understanding.

Searle's argument is sometimes presented in the form of a *reductio ad absurdum*. Take the best attempt at a program that is constitutive of understanding: for example, the best program that the AI research could ever hope to produce, or the program that putatively runs on the brains of Chinese speakers. Since the Chinese room is a universal computer, it will be able to run such a program. However, we

1. In this and what follows 'understanding simple Chinese stories' is sometimes abbreviated to 'understanding Chinese'.

cannot imagine the person inside the Chinese room *ever* understanding Chinese, no matter what program they are given. Hence, no such program that is sufficient for understanding can exist.

Both forms of the Chinese room argument appear in Searle's work.[2] Both depend on an assumption that I shall challenge.

The first argument has two premises. The first premise is Searle's key intuition: the person inside the room does not understand Chinese stories. The second premise is that the person inside the room can run any program. These two premises entail that no program can be sufficient for, or constitutive of, understanding Chinese.

**1.** The man inside the room cannot understand Chinese stories

**G** The man inside the Chinese room can run any program

**3.** Therefore, no program can be sufficient for, or constitutive of, understanding Chinese

Searle begins his original article with the second argument. This argument has the form: 'Let's assume that the mind actually works this way, let's assume that Strong AI is correct, what consequences follow?' One consequence is that the person inside the room, if given the right rule-book, would be able to understand Chinese. But, says Searle, that is *absurd*. We cannot imagine the person inside the room ever understanding Chinese no matter what rule-book they are given. Therefore, something must have gone wrong with our assumptions. Our main assumption was that Strong AI is true, so it must be that Strong AI is false.

**AI** Running a program is sufficient for, or constitutive of, understanding Chinese stories

**G** The man inside the Chinese room can run any program

**4.** Therefore, the man inside the room can understand Chinese stories

But this can't be

Either AI or G must be false. G is true, therefore AI must be false.

2. An example of the first form: 'I [the man inside the Chinese room] have inputs and outputs that are indistinguishable from those of a native Chinese speaker, and I can have any formal program you like, but still I understand nothing.' (Searle 1980b, p. 418). An example of the second: 'I offer an argument that is very simple: instantiating a program could not be constitutive of intentionality, because it would be possible to instantiate the program and still not have the right kind of intentionality. That is the point of the Chinese room example.' (Searle 1980a, p. 450).

Both versions of the argument depend on G: the assumption that the Chinese room can run any program. It is easy to see why G, or something like it, is needed. Searle's argument only works as an argument against AI if the Chinese room argument speaks to the general case of running *any* program. It would not be sufficient for Searle to show that only some programs do not produce understanding: he wishes to establish a general claim against Strong AI. In order to make the Chinese room argument work as a general argument against Strong AI—in order to make it speak to the case of all programs—Searle needs an assumption like G. My strategy in criticising Searle is to argue that G is false.

Note that one can weaken G and keep Searle's anti-Strong AI conclusion. Below are variants of the argument that weaken G as much as possible while preserving Searle's conclusion:

**1.** The man inside the room cannot understand Chinese stories

**G*** The man inside the Chinese room can run all programs putatively constitutive of understanding

**3*.** Therefore, no program putatively constitutive of understanding can really be sufficient for, or constitutive of, understanding Chinese

and,

**AI** Running a program is sufficient for, or constitutive of, understanding Chinese stories

**G**** The man inside the Chinese room can run at least one program constitutive of understanding

**4.** Therefore, the man inside the room can understand Chinese stories

But this can't be

Either AI or G** must be false. G** is true, therefore AI must be false.

The argument below is phrased primarily in terms of assumption G, but this is only for convenience. The argument works against G* and G** as well as G. Therefore, in what follows, my claim is not just that G is false, but that G* and G** are also false.

### 1.1 The Church–Turing thesis

Before considering why G is false, let us first consider why Searle might think that it is true. Consider the following statement from Searle (1992):

> We begin with two results in mathematical logic, the Church–Turing thesis and Turing's theorem. For our purposes, the Church–Turing thesis states that for any algorithm there is some Turing machine that can implement that algorithm. Turing's thesis says that there is a universal Turing machine that can simulate any Turing machine. Now if we put these two together, we have the result that a universal Turing machine [which is what Searle thinks a Chinese room is] can implement any algorithm whatever.[3] (Searle 1992, p. 202)

If this characterisation is correct, then it would justify Searle's assumption that a Chinese room can run ('implement') any program. However, there are two problems. First, Searle's statement of the Church–Turing thesis is incorrect. The Church–Turing thesis does not state that for any algorithm there is some Turing machine that can implement that algorithm. The Church–Turing thesis states that for any computable *function* (set of input–output pairings) there is some Turing machine that can reproduce that function. A function (a set of input–output pairings) and an algorithm for computing that function are not the same thing. Searle needs equivalence in algorithms, not functions, for his argument to work, and the Church–Turing thesis is silent about that. The second problem is that the notion of simulation in Turing's theorem is a technical one. In the context of Turing's theorem, simulation means reproduction of input–output pairings. Simulation does not mean, as Searle suggests, implementation of the same algorithm.

The Church–Turing thesis and Turing's theorem do not justify the assumption that the Chinese room can run any program, even under the assumption that the Chinese room is a universal Turing machine. Therefore, the Church–Turing thesis cannot support Searle's argument in the required way.[4]

3. For evidence that Searle thinks that the Chinese room is a universal Turing machine (strictly, an implementation of a universal Turing machine), see the exchange between Searle (1991) and Fodor (1991).

4. Copeland (1998) also criticises Searle for making a mistake with the Church–Turing thesis. Searle claims that the Church–Turing thesis entails that any physical process can be simulated by a computer (Searle 1992, p. 200). Copeland correctly points out that this is false: the Church–Turing thesis entails that a process can be simulated only if its input–output behaviour is governed by an effectively calculable function. Copeland argues that this requirement need not always be satisfied by a physical process. Both points concern the Church–Turing thesis, but they are different points. Copeland's point concerns the conditions under which the Church–Turing thesis can be applied to certain physical systems. My point concerns *what can be gained from* the Church–Turing thesis *once it has been applied to* those systems.

## 2 Criticism of Searle's argument

The Chinese room cannot run any program because even if the Chinese room is a universal computer, it is *not true* that a universal computer can run any program. The programs that a computer (universal or otherwise) can run depend on that machine's architecture. Programs are at the algorithmic level, and that level is tied to the implementation on particular machines. One cannot take a program that computes the addition function on, say, a register machine and run *that same program* on a Turing machine. The programs that can be run on the two machines are different in each case. This does not mean that a register machine and a Turing machine cannot compute the same functions. Famously, they can. The Church–Turing thesis states that any computer's input–output behaviour can be exactly reproduced by a universal computer. But this is equivalence at the level of input–output behaviour, not equivalence at the level of how that input–output is achieved.[5]

A Chinese room can reproduce the input–output behaviour of any Chinese speaker, but what Searle needs is for the Chinese room to reproduce that input–output pattern in the same way—to run the same programs—as a Chinese speaker. In particular, Searle needs the Chinese room to be able to run any program that Strong AI claims is constitutive of understanding. However, the programs that a system can run depend on that system's architecture. Certain architectures can run some programs and not others.

A Chinese room, like any computer, can run some programs and not others. *Prima facie*, there seems little reason to assume that a Chinese room can run all, or even any, of the programs that are putatively constitutive of understanding. Indeed, there are good reasons to think that it cannot. Systems that do understand, for example, human brains, have a radically different functional architecture to that of a Chinese room. An advocate of Strong AI could argue that understanding-producing programs are specific to brain-like architectures. Given the considerable architectural differences between brains and Chinese rooms, it is unlikely that programs that can be run on one must be able to be run on the other; certainly, this is an assumption that would require considerable defence. We have seen that Searle's defence in terms of the Church–Turing thesis does not work. Searle does not have anything to add to this defence. So as things stand, the advocate of Strong AI has the upper hand: she can justifiably reject the Chinese room argument. She can say that the Chinese room argument fails to show that understanding cannot consist in the running of a program, because the programs that she thinks are constitutive of understanding *aren't even considered by the argument*.

5. In short, Searle conflates the 'computational' and 'algorithmic' levels distinguished by Marr. Marr's computational level concerns *what* function is computed, his algorithmic level concerns *how* that function is computed. See Marr (1982), pp. 22–24.

What are the specific differences in functional architecture between the Chinese room and human brains? There is not enough known to give a full answer, but there are good reasons for thinking that there will be radical and irreconcilable differences.[6]

Consider the difference between parallel and serial architectures. The Chinese room has a serial architecture. Assume, as seems reasonable, that the brain has a parallel architecture. Programs that exploit specifically parallel architectures, for example, parallel search algorithms, literally cannot be run on serial machines. They rely on certain capabilities and operations that are not available on those machines—namely, the ability to perform more than one operation at a time. Serial and parallel computers can compute the same functions, but they cannot use the same methods for computing those functions. If the brain has a parallel architecture, and the programs that it runs are essentially parallel, as seems likely, then those programs literally cannot be run on a Chinese room.

Another source of difference is that the two systems may support different types of atomic operation. As specified by Searle, the Chinese room supports the atomic operations *compare*, *copy*, and *concatenate*. There seems no reason why the brain should support only these operations. If there is any difference in the atomic operations supported, then the individual steps in the two programs cannot be the same, creating problems for the notion that the two machines can run the same program.

Another possible source of difference is that the brain may have a different way of managing control from the Chinese room. There are many examples of architectures with different control flow from the von Neumann model that the Chinese room approximates. Consider the difference between the programs that can be run on a von Neumann-style PC and those that can be run on a computer based on Church's $\lambda$-calculus. Imagine trying to run a PC version of Microsoft Word on a machine with a $\lambda$-calculus architecture. Such a program could not be run as it currently stands. A $\lambda$-calculus version of Microsoft Word would have to work in a radically different way.[7] The brain may have a computational architecture that is as distant, or more distant, from the von Neumann model than the $\lambda$-calculus.[8]

In summary, Searle's Chinese room argument fails because G, or any suitably qualified version of it, is false. An advocate of Strong AI need not admit that *even one*

6. See Backus (1978) for a survey of different computational architectures and the effect that architecture has on the programs that a system can run.

7. For one thing, on a $\lambda$-calculus architecture there are no loops, instructions, or stored variables, so the standard von Neumann construction of a looped assignment statement could not be run.

8. A possible example: the von Neumann distinction between central processor and memory does not appear to hold true of the brain. Neurons, or group of neurons, appear to work as largely independent processors, and memory appears to be encoded within these processing units rather than in a separate store.

program that is putatively constitutive of understanding can be run on a Chinese room. The architecture of brains and Chinese rooms is too different. An advocate of Strong AI can claim, with justification, that understanding-producing programs are specific to brain-like architectures, and thereby resist G, G*, G**. In other words, she can hold onto her thesis for architectures other than that of the Chinese room.[9]

## 3 Objections and replies

### 3.1 The virtual brain machine objection

**Objection 1.** *Can't a Chinese room simulate any other computer? Can't one create, on the Chinese room computer, a virtual machine with the same architecture as the brain and run the brain's program on that virtual machine? Wouldn't the Chinese room then be able to run any program that could be run on the brain?*

Consider how such a proposal would work. Suppose that Strong AI claims that there is a program *P* that runs on the brains of Chinese speakers, and that program *P* is sufficient for, or constitutive of, understanding Chinese. Program *P* will be peculiar to the architecture of the human brain: it will use distinctive atomic operations, and manage flow of control and data in distinctive ways. Program *P* cannot be run on a Chinese room as it currently stands. The current suggestion is to get around this incompatibility by creating another program *Q* that runs on the Chinese room, a virtual brain machine (VBM). Program *Q* takes programs designed to be run on the human brain as data. Program *Q* processes these brain-like programs and produces output that is indistinguishable from that of the programs being run on a real brain. For many intents and purposes, a Chinese room running a VBM can be treated as a machine with the architecture of a brain. In particular, one can run program *P* on it. Therefore, *contra* my claim, the Chinese room can run program *P*. Furthermore, adding a VBM only adds another level of symbol shuffling, so it seems that even with a VBM, the person inside the room would still not be able to understand Chinese. Therefore, Searle's original argument goes through after all.

This objection relies on the notion of a virtual machine, and although that notion is compelling, it is easy to take it too literally. Virtual machines are not the machines that they appear to be. Virtual machines present interfaces that give the impression that a real machine is there, but they do not, strictly speaking, run the program that they are given. Virtual machines are automated procedures for turning one

9. This reply to Searle is (to my knowledge) novel; the replies that come closest are Lycan (1980); Rey (1986); Roberts (1990). However, these authors do not make the connection with the Church–Turing thesis, or the architecture-dependence of programs.

program into another. A VBM does the following: for each step or small group of steps $S$ in program $P$, it transforms those steps into a group of steps $S^*$ for the Chinese room to run, and then runs $S^*$. The VBM does not run program $P$, it runs a transformation of program $P$. In this respect, virtual machines are like compilers: they transform one program (the one that the user writes) into another program (the one that the machine runs). The difference is that compilers do the translation beforehand and all at once, virtual machines do the translation piece-wise and at runtime.

The original program and the one produced by the VBM have the same input–output characteristics, but they will almost certainly work in different ways. A transformed program and its original may only be distantly related if the architecture of the two machines is significantly different: this can be to the extent of the instructions being different, the order in which instructions are executed being different, steps inserted, steps removed, and large-scale features of the program changed.

Therefore, even if one were to construct a VBM, and give that VBM program $P$, the Chinese room would still not be running program $P$. The Chinese room would be running a systematic transformation of program $P$: a program with different steps, different atomic operations, and a different control structure. A VBM creates the illusion of the real brain being there—it presents the same I/O interface to the world—but it does not literally run the program that it is given. Therefore, adding a VBM does not get one closer to running program $P$ on the Chinese room.

### 3.2 The man with a brain objection

**Objection 2.** *Your criticism was that the Chinese room cannot run suitably brain-like programs, but the man inside the Chinese room has a brain, so surely he can run brain-like programs.*

This objection conflates two machines inside the Chinese room. One machine is the human brain, the other is the von Neumann-esque machine produced by the activity of the man inside the room. The claim of Strong AI is that a program that runs on the first machine constitutes understanding. Searle's argument is that if merely running a program is constitutive of understanding, then we can *take* the program that runs on the brain and run that program on the second machine: we can get the human clerk to work through it by hand. If merely running a program is sufficient for understanding, then understanding should be produced in this second case too. But, Searle says, it is absurd to think that the human clerk would understand Chinese merely by working through a program by hand. So, running a program cannot be sufficient for understanding Chinese. My point is that you

cannot transfer programs in the way that Searle suggests. The architecture of the two machines—the brain and the von Neumann-esque machine—is too different.

Searle requires this distinction between the two machines in order to make his argument work. His crucial move—taking the program that runs on a brain and giving that program to the human clerk—would not make sense otherwise. A consequence of the current objection is that Searle's own argument, as well as my criticism, would be impossible to understand.

A possible source of the confusion underlying this objection is that there are two senses of 'running a program' at issue in the Chinese room argument. In some cases, 'running a program' means running the program directly on the hardware of the brain. In others, 'running a program' means the human clerk working through the program by hand. Both senses of 'running a program' are legitimate, but programs are being run on different machines in each case.

### 3.3 The syntax/physics objection

**Objection 3.** *You say that the Chinese room cannot run the relevant program, but according to Searle's syntax/physics argument, any sufficiently complex physical system runs any program (Searle 1992, Ch. 9). So the Chinese room can run any program after all.*

According to Searle's syntax/physics argument, all that is required for a system to run a program is for an observer to be able to interpret that system's physical states in the right way. If the system is sufficiently complex, e.g. a Chinese room, then an observer can interpret it as running any program she likes. My claim is that there is something about the Chinese room, its architecture, that constrains the programs that it can run. If Searle's syntax/physics argument is correct, then there are no interesting constraints on the programs that a Chinese room can run.

There are two issues here. The first is whether Searle's syntax/physics argument is correct. If Searle's syntax/physics argument is incorrect, then the advocate of Strong AI is under no compulsion to accept it. I shall not pursue this issue here, but note that Chalmers (1996) and Copeland (1996) have argued that Searle's syntax/physics argument is wrong. The second issue is whether the syntax/physics argument, if correct, would be a good way for Searle to defend the Chinese room argument. Sadly for Searle, it is not: the syntax/physics argument cuts against the Chinese room argument just as much as it does against Strong AI. The syntax/physics argument cuts against the Chinese room argument in two ways: it undermines the ability to state the argument, and it undermines the motivation for giving the argument.

The syntax/physics argument undermines the ability to state the Chinese room argument because the Chinese room argument presupposes a determinate notion of computation. Searle acknowledges this: 'For the purposes of the original [Chinese room] argument, I was simply assuming that the syntactical characterization of the computer [assignment of computations to physical states] was unproblematic.' (Searle 1992, p. 210). The Chinese room argument assumes that a system runs *a* program. Without this assumption, it is difficult to make sense of Searle's claim of taking 'the' program that is putatively constitutive of understanding and 'giving' that program to the Chinese room, or his identification of the rule-book with the program that the Chinese room runs.

The syntax/physics argument undermines the motivation for giving the Chinese room argument because it renders Strong AI trivially false from the start. One of Searle's most firmly entrenched background assumptions is that understanding is an intrinsic property of a system: a system, such as a human being, either understands, or does not understand, independently of how observers view that system. If one were to conjoin to this assumption the conclusion of the syntax/physics argument—that the program that a system runs is an observer-*relative* property—then the conclusion immediately follows that running a program cannot be constitutive of understanding. An observer-independent property cannot be an observer-relative property. Strong AI is refuted before the Chinese room argument even gets started. This is a mixed blessing. One of the reasons why the Chinese room argument is interesting is that it shows that even if one treats the notion of computation as unproblematic, understanding cannot consist in the running of a program. Searle grants as many assumptions to an advocate of Strong AI as possible in an effort to provide a convincing argument that AI is the assumption that is false. In the revised argument above, one of these assumptions—non-trivial computational identity—is given up, and the overall argument is less interesting as a result.[10]

### 3.4 The abstraction objection

**Objection 4.** *You say that the Chinese room cannot run the relevant program, but the programs that a computer runs can be characterised more or less abstractly (Lycan 1987, pp. 46–47). If programs are construed abstractly enough, then the Chinese room can run them all.*

Lycan (1987) argues that a computer can be characterised as running a number of different programs depending on how abstractly one treats it. For example, a

10. Note that Searle himself claims that the Chinese room argument is independent of syntax/physics worries, see Searle (1992), p. 210; Searle (1994), p. 548; Searle (1997), pp. 14, 17.

chess-playing computer can be characterised as running a program that develops knights and looks for king-side weaknesses, or as running a program that evaluates trees of chess moves in a certain order, or as running a program whose steps are instructions in the programming language that the chess program was written, or as running a program whose steps are the machine code instructions of the processor on which the program is running.

At the most abstract level of program characterisation—developing knights and looking for king-side weaknesses—it seems possible that two systems with different architectures could run the same program. At the least abstract level—executing a specific set of machine code instructions—running the same program on radically different architectures looks less likely. Lycan argues that the question 'Do two machines run the same program?' does not have a single answer. Whether two machines run the same program depends on the level of program characterisation, and that level depends on the interests of the users.

First, note that even if Lycan is right, it is not clear how much this result would help Searle. Just because a system can be correctly described as running *more than one* program, that does not show that it can run *any* program. Let us call computational methods that can be accessed by a move to a higher or lower level of abstraction those that are accessible through a 'vertical' move. Let us call computational methods that can be accessed by a move to a different architecture those that are accessible through a 'horizontal' move. The problem that Searle faces is that methods that are accessible through vertical moves *are not the same* as those that are accessible through horizontal moves.

Consider the task of moving a parallel search program, running on a parallel machine, to a serial machine. As Lycan describes, the serial computer can be characterised as running a variety of programs by thinking of it more or less abstractly; in other words, it can access various computational methods through vertical moves. However, whatever flexibility there is here, it is not enough to allow the serial computer to run the parallel program. This is because at *no* level of abstraction can the serial computer perform parallel operations. In order to access *that* particular computational method, a horizontal move, not just a vertical move, is required. A serial computer, via purely vertical moves, cannot access every computational method. One might object that if the *parallel* computer is also described abstractly enough, then its programs *can* be run by the serial computer. But even if this is true, it is irrelevant. Pointing out that one of the computational methods accessible to a parallel computer via vertical moves can also be run on the serial computer does not establish what Searle needs—that *any* given computational method that can be run on the parallel computer can also be run on the serial computer. More generally, just because a program $Q$ that is accessible by a vertical move from program $P$ on a

parallel computer can be run on a serial computer, that does not show that *P* itself can be run on a serial computer.

Here is another example. Consider two versions of Microsoft Word, one for a $\lambda$-calculus computer and one for an ordinary PC. It is possible that at some level of abstraction, short of brute I/O equivalence, the two machines can be correctly described as running the same program. However, this does not establish what Searle needs: that *any* program that can be run on one can be run on the other, i.e. that they can share the same computational methods. If, at some level of abstraction, the two machines can be described as running the same program, then there is a single computational method that is accessible to both through vertical moves. But this does not show that every computational method is shared between the two machines. Indeed, there are good reasons to think that every computational method cannot be shared. For if every computational method *could* be shared, then there is no reason why the two versions of Microsoft Word would have to differ substantially at all, at any level of abstraction. It seems that the reason why they do differ, at least at some levels of abstraction, is that the computational methods available on one machine are not available on the other. Both machines may be able to access some common computational methods though vertical moves, but they cannot share all their methods—which Searle requires—without appropriate changes in architecture. In short, the problem with Lycan's defence is that it assumes that moving to higher levels of abstraction allows the Chinese room to access *all* computational methods, whereas it only allows access to some computational methods.

A second problem with this objection is that it is by no means clear that Lycan's position about program identity is correct. Lycan's argument presupposes that each of level of abstraction is equally valid. But it could be argued that this intuition is mistaken. For example, Pylyshyn (1984) argues that there is a privileged level of abstraction at which questions about program identity should be decided. This level is the level of the system's primitive representations and operations. Two systems with different architectures are not able to run the same programs unless they share the same primitive representations and operations. For Pylyshyn, talk of more abstract levels of program organisation may have pragmatic value, but it should not be given any metaphysical weight in deciding questions about program identity. Pylyshyn's position requires considerable defence, but it at least shows that there are alternatives to Lycan.[11]

11. Pylyshyn (1984), pp. 91–92.

### 3.5 The 'same conclusion' objection

**Objection 5.** *It does not matter if Searle made a mistake, your conclusion about the importance of the underlying architecture, the brain, is the same as his anyway.*

The view that I place at the disposal of an advocate of Strong AI does mesh with some of Searle's comments about the importance of underlying biology. According to the view that I suggest, the brain runs a particular program because it has a distinctive functional architecture, and it has a distinctive architecture because of its physical and biological makeup.[12] If one were to create an artificial system that runs the same programs as the brain, then one would create a system that, in some important respects, is similar to a brain. Therefore, both views give an important role to biology. Nevertheless, the two positions are different.

First, the motivation for appeal to biology, and the role played by biology, is different in each case. Searle's appeal to biology has been justly criticised for lack of motivation.[13] Searle concludes that since computation is not sufficient for understanding, there must be some other factor, and he appears to pick biology for lack of a better candidate. On Searle's account, it is mysterious which features of underlying biology are relevant, or why biology should be relevant at all—biology appears as a *deus ex machina*. On the view that I suggest, there is a clear rationale for appeal to biology. The physical makeup of brains determines their computational architecture, and therefore the programs that brains can run. Biology is important because it determines the hardware that supports the programs required for understanding. Only appropriately brain-like architectures can run the kinds of programs that are constitutive of understanding.

The second difference is that the claim that Searle denies—understanding can consist in running a program—is asserted on the competing view. Hence, the end-positions of the two views are different. Note that this is not due to a trivial redefinition of 'program' as 'whatever it is that produces understanding'. Rather, it is because of an acknowledgement that programs depend on functional architecture, and functional architecture depends on biology. Consequently, on the view that I suggest, it is still possible for AI to turn out to be false. Even if the Chinese room argument fails, AI may be shown to be false in other ways. (For example, someone might argue that human-like causal connections with the external world are necessary for understanding; computations, even brain-like ones, by themselves are insufficient.)

12. Cf. Marr (1982): 'The algorithm depends heavily on the computational theory, of course, but it also depends on the characteristics of the hardware in which it is to be implemented.' (p. 337). Remember that for the purpose of the Chinese room argument, both Searle and I are bracketing syntax/physics worries, see Objection 3.

13. For example, see Block (1980); Dennett (1980); Fodor (1980); Pylyshyn (1980).

### 3.6 The 'unnecessary baggage' objection

**Objection 6.** *The situation that you suggest—a brain-like computer running a characteristic program—involves no more than reproducing the causal powers of the brain, which was Searle's original suggestion. Why not drop the idea of running a program as unnecessary baggage, and say, as Searle does, that understanding systems should be characterised in physical terms, i.e. in terms of their causal powers?*

This objection is different from Objection 5, which claimed that the positions suggested by Searle and myself are identical. The current objection admits that the two positions are different, but claims that Searle's position is preferable to mine. The claim is that the explanatory virtues of the account that Searle suggests are greater than those of my account: the notion of computation in my account is just unnecessary baggage. Here the disagreement is not about the *members* of the class of systems that understand—both Searle and I agree that only appropriately brain-like systems can understand. The disagreement is about *how that class should be characterised*. Should the class be characterised purely in terms of physical features, as Searle suggests, or should it be characterised in terms of computational features? Let us consider the advantages and disadvantages of each approach.

The computational approach has had numerous explanatory successes. It explains a number of puzzling features of systems that understand such as the systematicity and productivity of thought, the possibility of rational inference, and why some aspects of the relationship between thought and language obtain. There are deficiencies, but the computational account does have something useful to say about what understanding systems have in common.

Now consider Searle's way of characterising the class of understanding systems. It is hard to see how his account could give a satisfying explanation of what such systems have in common. What purely physical features do understanding systems have in common? Acetylcholine and serotonin? Unlikely, since other substances could have been used in their place. There seems nothing special about acetylcholine and serotonin apart from their ability to occupy certain causal roles in understanding systems. Accounts of how systems understand, like accounts of other biological phenomena, should be functional: it is causal roles and their inter-relation that explain why a given biological system performs the function it does. This is true even of Searle's paradigmatic example of a biological phenomenon, photosynthesis. Look at any scientific account of photosynthesis and one finds functionalist explanation *par excellence*.[14] For this reason, it is unclear why, or even how, Searle could be hostile to functionalist accounts.

14. For example, see Lawlor (2001).

If Searle is not hostile to functionalist accounts, then it is unclear why he is hostile to giving the causal roles involved a computational gloss. As we saw in the response to the previous objection, there need be nothing unbiological about such an approach. If computational accounts offer the chance of explaining the cognitive phenomena listed above, then, provided that the notion of computation is not otherwise problematic, why not use it? Searle's alternative just does not seem plausible.

The same point can be phrased as a dilemma. If Searle's proposal is understood as a brute physical account with no functionalist component, then it would provide a massively disjunctive and unexplanatory characterisation of the class of understanding systems. If his account is understood as explanation in terms of causal role, hence functional, it is difficult to see why he is so hostile to giving those causal roles a computational gloss. There seems nothing to gain and much to lose by such a move.

### 3.7 The Chinese gym objection

**Objection 7.** *If the setup inside the Chinese room cannot run brain-like programs, then change it so that it can. Once the setup is changed, then Searle's argument can be run as before.*

Searle (1990) proposes a modification along these lines in response to connectionist criticism. Searle suggests replacing the single man inside the Chinese room with a collection of monolingual English-speaking men inside a gym. Each man would simulate the computational behaviour of a single neuron. The men would pass messages to each other, so that the gym as a whole would perform a connectionist computation. Searle claims that none of the men understand Chinese, and infers from this that there is no Chinese understanding going on inside the gym. He concludes that connectionist computation cannot be sufficient for, or constitutive of, understanding.[15]

Copeland (1993) correctly points out that this argument would not convince a connectionist. A connectionist does not claim that individual neurons understand, she claims that the system as a whole understands. Searle does nothing to show that the failure of the individual men to understand Chinese entails that the system as a whole cannot understand Chinese. Therefore, according to Copeland, this version of the Chinese room argument falls victim to a particularly vicious form of the Systems reply.

The problem with the Chinese gym argument—and the reason why it so easily falls victim to the Systems reply—is that unlike Searle's original thought experiment, the

15. For the argument, see Searle (1990), p. 22.

key intuition behind this revised argument is simply not convincing. A large part of the appeal of Searle's original Chinese room argument was that his key intuition—that there is no Chinese understanding going on inside the room—was plausible. The reason why it was plausible is that it was possible to imagine *oneself* performing the computation and failing to understand Chinese. One could *empathise* with the system carrying out the computation. This intuition is lost in the Chinese gym scenario. It is not possible to imagine what it is like to be a Chinese gym. As a result, Searle's key intuition is unconvincing in a way that, arguably, it is not in the original argument.

Therefore, Searle faces another dilemma: as it stands the Chinese room does not reflect the architecture of the human brain, and so cannot run the appropriate programs; however if Searle were to modify the architecture of the Chinese room to make it more brain-like, then he would lose the key intuition that originally made the Chinese room argument so convincing.

### 3.8 The syntax/semantics objection

**Objection 8.** *Searle now prefers a different argument, which he calls his 'axiomatic' argument. You have said nothing about this argument, so why should Searle worry?*

Searle's axiomatic argument is:

**A1** Programs are formal (syntactic)

**A2** Minds have contents (semantics)

**A3** Syntax is not sufficient for semantics

**A4** Therefore, programs are not minds

This argument is simple but deceptive. It is deceptive because although programs and computations have formal properties, they also have other properties. A number of advocates of the computational theory of mind (CTM) have plausibly argued that real-world computational tokens must have *representational* properties in order to qualify as computational tokens at all.[16] Typically, the argument is that computational identity, and the distinction between computational processes and other causal processes, can only be made if computations have representational content. Real-world computations must therefore have *both* formal and representational (semantic) properties if they are to qualify as computations. If one accepts this

16. Philosophers with views as divergent as Fodor (1975), Dennett (1971), Churchland (1986), and Cummins (1989) agree that real-world computation must involve representation.

conclusion—and it is not obvious that it is false—then Searle's axiomatic argument can be resisted.

What is wrong with Searle's axiomatic argument can be seen by evaluating premise A1. If A1 is interpreted as 'Programs are formal (and may have other properties)' then the axiomatic argument is invalid: the conclusion does not follow from the premises. If A1 is interpreted as 'Programs are *only* formal', then the argument may be valid, but it is no longer sound, since computations performed by real-world systems must have both formal and semantic properties. Computation, as performed by real-world systems, is not the strictly non-semantic notion that Searle assumes.

## 3.9 Turing's definition of algorithm

**Objection 9.** *You say that the Chinese room cannot perform certain brain-like computations, but Turing's original definition of computation defined computation as what could be achieved by a man working by himself. Therefore, the man inside the Chinese room can run any program.*

Turing (1936–1937) characterised an algorithm as a procedure involving a finite number of steps that could in principle be carried out by a human being unaided by any machinery save pencil and paper and demanding no insight or ingenuity on the part of the human. This sounds like the situation inside the Chinese room. For the purpose of this objection, make whatever changes are necessary to make Searle's situation identical to Turing's. It now appears that my claim that there are algorithms that a Chinese room cannot run is false. The Chinese room can, by definition, run any algorithm.

### *3.9.1 Consequences*

What if this objection is fatal to my argument?

First, note that this defence of Searle's argument does not challenge my claim about the architecture-dependence of algorithms. The reason why the Chinese room can run any algorithm is not because algorithms are hardware or architecture independent, but because the architecture of Chinese room happens to be *just that one* that can run any algorithm. If Searle had put any other machine inside the Chinese room (a Turing machine, a register machine, an electronic PC with unbounded memory), then my argument would have gone through.

Second, if this objection is correct, then it shows that the Chinese room argument is strong in an unusual direction. The objection not only entails that my criticism

fails, it also entails that connectionist criticisms fail. Connectionists argue that one can avoid Searle's conclusion by switching to a different architecture (connectionist instead of von Neumann). My original criticism could be understood as providing a rationale for such a change. However, if the current defence is correct, then all of this is wrong. Any criticism that relies on alternative architectures cannot succeed. This is because connectionists, to the extent they admit that they are contesting the same proposition as Searle—to the extent that they admit that the proposition at issue is whether understanding can consist in the running of a particular *program*—and to the extent that they accept Turing's definition of program, have to admit that the Chinese room can, by definition, run any program. Hence, the Chinese room can run any program that a connectionist system can run. An advocate of connectionism might object that a connectionist system is in many ways unlike a Chinese room. However, this is irrelevant unless accompanied by either a rejection of the thesis that running a program is sufficient for understanding (hence conceding that Searle is correct), or a rejection of Turing's definition of a program.

### *3.9.2 Criticism of the defence*

I shall now argue that the current defence fails in three ways. First, it is not clear that Turing gave the definition of the notion of algorithm used by Searle's defence; it is compatible with what Turing wrote to hold a different view, and one not conducive to Searle's argument. Second, on this alternative view, there are algorithms that the Chinese room cannot run. Third, even if one were to accept the current defence, it is not clear that it would ultimately help Searle.

Turing and Church can be read in a way that does not support Searle's argument. Turing and Church were primarily interested in constructing a formal predicate for picking out a certain class of functions: the computable functions. This class of functions can be characterised in a number of different ways. One way is in terms of the informal definition given above. Another way is in terms of the formal predicates introduced by Turing and Church. But there are plenty of other ways, both formal and informal, of picking out this class. There is no reason to take the setup described by Turing as particularly fundamental or primary; it is just one way among many of characterising that class. If one likes, the setup described by Turing can be taken as definitive of *what* functions can be computed, but it should not be taken as definitive of *how* they can be computed. What was special about the setup with the clerk working by hand was that it was particularly useful to Turing in arguing that his formal predicate (*viz.* Turing machines) ends up picking out the same class of functions as are captured by our informal notion of computation.

Turing's aim was to formalise what a human mathematician does in 'computing'

a function in the sense of evaluating it by application of rote procedure. In §9 of (Turing 1936–1937), Turing argues that any method performed by a human computer could be performed by a machine (*viz.* a Turing machine). However, Turing did *not* argue for the converse of this claim: that every method that can be performed by a computing machine can also be performed by a human computer. Turing wished to show that the operations of a human computer can be abstracted into the operations of a Turing machine, he did not show that any method used by any given computing machine can be performed by a lone human being.

Second, on at least some levels, it seems plausible that there are algorithms that a human working by herself cannot run. A human working by herself can compute any computable function, but she cannot use any possible method to do so. Consider parallel search algorithms, or data-flow algorithms, or quantum computing algorithms, or enzymatic algorithms.[17] It seems plausible that a human being working by herself cannot run these algorithms. Humans can run related algorithms that compute the same functions, but they are different algorithms in each case.

One might object: Why call the algorithms different? Why not say, for example, that the serial and parallel algorithms are different versions of the same algorithm? The problem with this response is that it only works as a defence of Searle if one already has a clear view of how algorithms should be individuated. What criteria should we use to decide when two algorithms are the same or different? One solution is to individuate algorithms extensionally, i.e. by the function they compute. This provides a simple, clear way of individuating algorithms. But individuating algorithms extensionally is no good for Strong AI or functionalism. It would collapse those theories into a form of behaviourism. What is distinctive about Strong AI and functionalism is that they care about differences between algorithms beyond differences in their I/O behaviour.

The question then arises of what features, over and above I/O, should count as making two algorithms the same or different. First, note that the individuation conditions of algorithms appear to be sensitive to context. It does not make sense to ask whether two algorithms are the same or different *simpliciter*: in some situations, two algorithms count as the same, in other situations they do not. If we are interested in serial–parallel differences, then we tend to classify algorithms that differ in serial–parallel structure as different. If we are interested in other features of algorithms (e.g. their control structure), then we may *ignore* serial–parallel differences and treat two previously distinguished algorithms as the same. There does not appear to be

17. See Grama and Kumar (1995) for a discussion of parallel search algorithms. See Treleaven, Brownbridge and Hopkins (1982) for a discussion of data-flow algorithms. See Nielsen and Chuang (2000) for a discussion of quantum computing algorithms. See Barrett (2005) for a discussion of enzymatic algorithms.

a single, context-independent, answer to whether two algorithms are the same or different.

Our original question can therefore be rephrased: *in the context of Strong AI and the CTM*, what are the individuation conditions of algorithms? Are algorithms that differ, for example, in serial–parallel structure, different? Unfortunately, the answer is unclear. The reason why it is unclear is that Strong AI and the CTM give us *no clue* which features of algorithms are relevant in their context. According to Strong AI, it left entirely open, and rather mysterious, *what it is* about running a particular algorithm that is supposed to produce mentality. The thesis of Strong AI says nothing about *which* features of running an algorithm, over and above its I/O behaviour, are responsible for understanding; it only says that *some* such features are responsible. Thus, we are left with no idea as to which features of algorithms are significant to their individuation in the context of Strong AI. Maybe serial–parallel differences do matter; Strong AI is silent on the issue.

The problem for Searle can be summarised as follows. Turing's supposed definition of the notion of algorithm has difficulties with certain kinds of process that we would intuitively like to call algorithms. For example, it has difficulties with parallel processes. One of two approaches can be taken: (i) admit that there are algorithms that a human working by herself cannot run; or (ii) say that these processes are the same as algorithms that she can run. Approach (i) concedes that Searle's defence fails as a defence of the Chinese room argument against my original criticism. Approach (ii) opens up a new and difficult area. It relies on a notion of algorithm individuation in the context of Strong AI and the CTM. Individuation conditions for algorithms are hard to come by. A simple approach, such as individuating algorithms extensionally, is not going to work. However, it is unclear how to improve on it. Strong AI and the CTM leave the issue entirely open. If Searle wishes to use Turing's definition in his defence then he had better have an appropriate—and justifiable—notion of algorithm individuation up his sleeve. It is not clear how he could have this.

Finally, even if the current defence is accepted, it is not clear that it would ultimately help Searle. Changes need to be made to Searle's setup in order to match exactly the situation described by Turing. One of these changes is that the rule-book, instead of containing three columns of Chinese symbols and a rewrite rule, can now contain *any* instructions in English that can be followed without undue insight or ingenuity. This gives a lot more room to manoeuvre for a critic who argues that the man inside the room can understand Chinese. In the original thought experiment, it was intuitively plausible that the man could not understand Chinese no matter which Chinese symbols were put in the three columns, or how complicated the rewrite rule. But now that almost any instruction in English can be given, the situation is not so clear. Why not give the man a Chinese–English dictionary and compositional

theory of meaning? Davidson has indicated that such a theory of meaning could be suitably extensional and followed without undue insight, provided that one has a pre-existing grasp of the meta-language (English in this case), and Searle explicitly allows us to assume that this is true.[18] If Searle wishes to exclude these cases then he has to move away from the details of the situation specified by Turing, and if he moves away from the details of that situation, then he opens himself up to my original criticism.

## 4 Conclusion

The Chinese room argument fails to rule out the possibility that understanding can consist in the running of a program. An advocate of Strong AI can hold onto her thesis for architectures different from that of the Chinese room. However, we have seen that part of the reason why Searle's criticism fails is that Strong AI leaves entirely open *what it is* about running a particular program that is supposed produce understanding. In this respect, Strong AI is an inchoate position. Searle might take comfort in the thought that if the thesis of Strong AI were more fully specified—and consequently we had a better grasp of the individuation conditions of algorithms in that context—then an analogous argument to the Chinese room argument may still be possible.

## Acknowledgements

I would like to thank a number of people who have made helpful comments on earlier drafts of this work, including Mike Beaton, David Chalmers, Ron Chrisley, Tim Crane, Julien Deonna, John Hyman, Chris Jones, Martin Kusch, Neil Manson, Ben Morrison, Alex Oliver, Tim Lewens, Peter Lipton, Erica Roedder, and John Searle. Sections from this paper have been presented at conferences and seminars at the University of Oxford, University of Cambridge, Columbia University, University of Sussex, and University of Geneva.

## References

Backus, J. 1978. 'Can programming be liberated from the von Neumann style? A functional style and its algebra of programs'. *Communications of the ACM* 21:613–641.

18. Searle (1980b), p. 418.

Barrett, H. C. 2005. 'Enzymatic Computation and Cognitive Modularity'. *Mind and Language* 20:259–287.

Block, N. 1980. 'What intuitions about homunculi don't show'. *Behavioral and Brain Sciences* 3:425–426.

Chalmers, D. J. 1996. *The Conscious Mind.* Oxford: Oxford University Press.

Churchland, P. S. 1986. *Neurophilosophy.* Cambridge, MA: MIT Press.

Copeland, B. J. 1993. 'The curious case of the Chinese gym'. *Synthese* 95:173–186.

———. 1996. 'What is computation?' *Synthese* 108:335–359.

———. 1998. 'Turing's O-machines, Searle, Penrose and the brain'. *Analysis* 58:128–138.

Cummins, R. 1989. *Meaning and Mental Representation.* Cambridge, MA: MIT Press.

Dennett, D. C. 1971. 'Intentional Systems'. *The Journal of Philosophy* 68:87–106.

———. 1980. 'The milk of human intentionality'. *Behavioral and Brain Sciences* 3:428–430.

Fodor, J. A. 1975. *The Language of Thought.* Sussex: The Harvester Press.

———. 1980. 'Searle on what only brains can do'. *Behavioral and Brain Sciences* 3:431–432.

———. 1991. 'Yin and Yang in the Chinese Room'. In *The Nature of Mind,* edited by D. M. Rosenthal, 524–525. Oxford: Oxford University Press.

Grama, A. Y., and V. Kumar. 1995. 'A survey of parallel search algorithms for discrete optimization problems'. *ORSA Journal of Computing* 7:365–385.

Lawlor, D. W. 2001. *Photosynthesis: Molecular, Physiological and Environmental Processes.* Oxford: BIOS Scientific Publishers.

Lycan, W. G. 1980. 'The functionalist reply (Ohio State)'. *Behavioral and Brain Sciences* 3:434–435.

———. 1987. *Consciousness.* Cambridge, MA: MIT Press.

Marr, D. 1982. *Vision.* San Francisco, CA: W. H. Freeman.

Nielsen, M. A., and I. L. Chuang. 2000. *Quantum Computation and Quantum Information.* Cambridge: Cambridge University Press.

Pylyshyn, Z. W. 1980. 'The "causal power" of machines'. *Behavioral and Brain Sciences* 3:442–444.

Pylyshyn, Z. W. 1984. *Computation and Cognition.* Cambridge, MA: MIT Press.

Rey, G. 1986. 'What's really going on in Searle's "Chinese room"'. *Philosophical Studies* 50:169–185.

Roberts, L. 1990. 'Searle's extension of the Chinese Room to connectionist machines'. *Journal of Experimental and Theoretical Artificial Intelligence* 2:185–187.

Searle, J. R. 1980a. 'Author's Response'. *Behavioral and Brain Sciences* 3:450–456.

———. 1980b. 'Minds, brains, and programs'. *Behavioral and Brain Sciences* 3:417–424.

———. 1990. 'Is the brain's mind a computer program?' *Scientific American* 262:20–25.

———. 1991. 'Yin and Yang Strike Out'. In *The Nature of Mind,* edited by D. M. Rosenthal, 525–526. Oxford: Oxford University Press.

———. 1992. *The Rediscovery of the Mind.* Cambridge, MA: MIT Press.

———. 1994. 'Searle, John R. entry'. In *A Companion to the Philosophy of Mind,* edited by S. Guttenplan, 544–550. Oxford: Blackwell.

———. 1997. *The Mystery of Consciousness.* London: Granta Books.

Treleaven, P. C., D. R. Brownbridge and R. P. Hopkins. 1982. 'Data-Driven and Demand-Driven Computer Architecture'. *ACM Computing Surveys* 14:93–143.

Turing, A. M. 1936–1937. 'On computable numbers, with an application to the *Entscheidungsproblem*'. *Proceeding of the London Mathematical Society, series 2* 42:230–265.