Knowledge, Understanding, and Computational Complexity

Ian Parberry*
Center for Research in Parallel and Distributed Computing
Department of Computer Sciences
University of North Texas

Abstract

Searle's arguments that intelligence cannot arise from formal programs are refuted by arguing that his analogies and thought-experiments are fundamentally flawed: he imagines a world in which computation is free. It is argued instead that although cognition may in principle be realized by symbol processing machines, such a computation is likely to have resource requirements that would prevent a symbol processing program for cognition from being designed, implemented, or executed. In the course of the argument the following observations are made: (1) A system can have knowledge, but no understanding. (2) Understanding is a method by which cognitive computations are carried out with limited resources. (3) Introspection is inadequate for analyzing the mind. (4) Simulation of the brain by a computer is unlikely not because of the massive computational power of the brain, but because of the overhead required when one model of computation is simulated by another. (5) Intentionality is a property that arises from systems of sufficient computational power that have the appropriate design. (6) Models of cognition can be developed in direct analogy with technical results from the field of computational complexity theory.

Penrose [30] has stated

...I am inclined to think (though, no doubt, on quite inadequate grounds) that unlike the basic question of computability itself, the issues of complexity theory are not quite the central ones in relation to mental phenomena.

On the contrary, I intend to demonstrate that the principles of computational complexity theory can give insights into cognition.

In 1980, Searle [36] published a critique of Artificial Intelligence that almost immediately caused a flurry of debate and commentary in academic circles. The paper distinguishes

*Author’s address: Department of Computer Sciences, University of North Texas, P.O. Box 13886, Denton, TX 76203-3886, U.S.A. Electronic mail: ian@ponder.csci.unt.edu.
between weak AI, which uses the computer as a tool to understand cognition, and strong AI, which has as its main goal the recreation of cognition in a computer by means of a formal symbol-processing program. Searle professes to prove by thought-experiment, analogy, and introspection that no formal program can think, and thus deduces that strong AI is misguided.

Despite the flood of criticism and counter-criticism that has been published, Searle seems to have changed his opinions little over the past decade (Searle [37, 38]). As a theoretical computer scientist I do not find his arguments convincing. I propose here to expose some fundamental misunderstandings in his arguments. I do not directly refute his claim that strong AI is misguided, but I propose to show that his demonstration of this proposition is deeply flawed. I believe that strong AI cannot be dismissed on purely philosophical grounds. However, in the course of my argument I will raise some of my own doubts about strong AI.

The three main weapons that Searle uses against strong AI are introspection, reasoning by analogy, and gedankenexperiment. Introspection can be highly unstable pedagogical ground, since in using the mind to observe and reason about itself, one risks running afoul of the Heisenberg Uncertainty Principle: the process of self-analysis may change the mind to the extent that any conclusions are cast into serious doubt. Nonetheless, I am prepared to allow introspection within certain bounds: I will allow Searle to look within himself and state that he understands English and does not understand Chinese.

I am suspicious of reasoning by analogy primarily because one needs little experience to realize that an analogy can be inappropriate if not properly subjected to the scrutiny of logic. Similarly, the gedankenexperiment, despite its illustrious history, can be seriously misguided. Since a gedankenexperiment is carried out purely in the mind, the conductor of the experiment is free to construct a fictional world in which reality does not apply, and hence runs the risk of coming to conclusions that have no basis in the real world. This is the fundamental flaw in Searle’s reasoning: he carries out his gedankenexperiment in an imaginary world where computation costs nothing.

Many academics from outside the field of Computer Science who like to publish papers in the field appear to suffer from the misguided belief that Computer Science is a shallow discipline (if nothing else, because it has the word “Science” in its name). Searle, like many critics of Computer Science, does not appear to be aware of current tends in research. Searle’s arguments are limited to the theoretical computer science before the 1970’s, which is based on the concept of computability, and the Church-Turing thesis that all models of symbolic computation are essentially the same.

Such a computational model assumes that computation is free. Unfortunately, just because a function is computable in the Church-Turing sense does not automatically mean that it is computable in the real world. Computation consumes resources, including time, memory, hardware, and power. A theory of computation, called computational complexity theory has grown from this simple observation, starting with the seminal paper of Hartmanis and Stearns [16]. The prime tenet of this technical field is that some computational problems intrinsically require more resources than others. The resource usage of a computation is measured as a function of the size of the problem being solved, with the assumption that we

---

1Computational complexity theory should not be the confused with the more recent science of complexity popularized by physicists.
can solve small problems with the computers available to us now, and we will wish to scale up to larger problems as larger and faster computers become available.

The crux of Searle’s argument is the following: just because a computer can compute something does not imply that it understands it. This is a reasonable hypothesis in the light of 1950’s Computer Science: a function being computable is not sufficient reason to believe that something that computes it truly understands it. According to Searle, proponents of strong AI, in contrast, believe the opposite. The Turing test (Turing [42]) pits a human being against a computer. If an independent observer cannot tell in conversation with the two via some anonymous medium such as a teletype which is the computer and which is the human being, then the computer is said by proponents of strong AI to be “intelligent”.

Searle’s *gedankenexperiment* consists of the following. Program a computer to converse in a natural language by providing it with a table of all possible inputs and their corresponding outputs. When given an input, the computer looks up the correct response in the table, and outputs that response. He reasons that this passes the Turing test, but cannot be said to really *understand* what it is doing. He justifies the latter observation with an analogy. A human being can be given such a look-up table for a language that he or she does not understand, for example, Chinese. This person can pass the Turing test in Chinese, despite the fact that they do not understand Chinese. Unlike many of Searle’s critics, I am quite comfortable with this line of argument, and quite willing to concede that a computer programmed in this manner does not understand what it is doing in any reasonable sense of the word. However, Searle has missed an important point early in his argument. He has assumed that such a computer program is possible. I believe that such a program is not possible for the simple reason that it requires too much in the way of resources.

Since the number of legal utterances in a natural language is uncountable (Langendoen and Postal [17]), it is impossible to compile a complete look-up table of a language such as English or Chinese. However, this is not a serious barrier to the experiment. It would be sufficient for the purposes of passing the Turing test to compile a table of commonly used statements and legitimate responses. Whilst the number of commonly used questions and statements is a matter of some debate, a conservative lower bound is easy to obtain by considering questions of a particular form.

Consider queries of the form

“Which is the largest, a <noun>1, a <noun>2, a <noun>3, a <noun>4, a <noun>5, a <noun>6, or a <noun>7?”

where <noun> denotes any commonly used noun. Seven nouns were chosen rather than any other number since that appears to be the number of concepts that a typical human being can grasp simultaneously (Miller [20]). How many queries are there of this form? There is little difficulty in constructing a list of 100 commonly known animals (see, for example, Figure 1). Therefore there are 100 choices for the first noun, 99 for the second, etc., giving a total of $100 \cdot 99 \cdot 98 \cdot 97 \cdot 96 \cdot 95 \cdot 94 = 8 \times 10^{13}$ queries based on Figure 1 alone.

---

2 Alan Turing made fundamental contributions to both theoretical computer science and AI, which is not surprising since the two fields were at the time inexplicably intertwined by the fact that the only computational device upon which to model a computer was an intelligent one: the brain.

<table>
<thead>
<tr>
<th>aardvark</th>
<th>crocodile</th>
<th>guinea pig</th>
<th>orangutan</th>
<th>shark</th>
</tr>
</thead>
<tbody>
<tr>
<td>ant</td>
<td>deer</td>
<td>hamster</td>
<td>ostrich</td>
<td>sheep</td>
</tr>
<tr>
<td>antelope</td>
<td>dog</td>
<td>horse</td>
<td>otter</td>
<td>shrimp</td>
</tr>
<tr>
<td>bear</td>
<td>dolphin</td>
<td>hummingbird</td>
<td>owl</td>
<td>skunk</td>
</tr>
<tr>
<td>beaver</td>
<td>donkey</td>
<td>hyena</td>
<td>panda</td>
<td>slug</td>
</tr>
<tr>
<td>bee</td>
<td>duck</td>
<td>jaguar</td>
<td>panther</td>
<td>snail</td>
</tr>
<tr>
<td>beetle</td>
<td>eagle</td>
<td>jellyfish</td>
<td>penguin</td>
<td>snake</td>
</tr>
<tr>
<td>buffalo</td>
<td>eel</td>
<td>kangaroo</td>
<td>pig</td>
<td>spider</td>
</tr>
<tr>
<td>butterfly</td>
<td>ferret</td>
<td>koala</td>
<td>possum</td>
<td>squirrel</td>
</tr>
<tr>
<td>cat</td>
<td>finch</td>
<td>lion</td>
<td>puma</td>
<td>starfish</td>
</tr>
<tr>
<td>caterpillar</td>
<td>fly</td>
<td>lizard</td>
<td>rabbit</td>
<td>swan</td>
</tr>
<tr>
<td>centipede</td>
<td>fox</td>
<td>llama</td>
<td>racoon</td>
<td>tiger</td>
</tr>
<tr>
<td>chicken</td>
<td>frog</td>
<td>lobster</td>
<td>rat</td>
<td>toad</td>
</tr>
<tr>
<td>chimpanzee</td>
<td>gerbil</td>
<td>marmoset</td>
<td>rhinoceros</td>
<td>tortoise</td>
</tr>
<tr>
<td>chipmunk</td>
<td>gibbon</td>
<td>monkey</td>
<td>salamander</td>
<td>turtle</td>
</tr>
<tr>
<td>cicada</td>
<td>giraffe</td>
<td>mosquito</td>
<td>sardine</td>
<td>wasp</td>
</tr>
<tr>
<td>cockroach</td>
<td>gnat</td>
<td>moth</td>
<td>scorpion</td>
<td>weasel</td>
</tr>
<tr>
<td>cow</td>
<td>goat</td>
<td>mouse</td>
<td>sea lion</td>
<td>whale</td>
</tr>
<tr>
<td>coyote</td>
<td>goose</td>
<td>newt</td>
<td>seahorse</td>
<td>wolf</td>
</tr>
<tr>
<td>cricket</td>
<td>gorilla</td>
<td>octopus</td>
<td>seal</td>
<td>zebra</td>
</tr>
</tbody>
</table>

Figure 1: 100 animals.

Figure 2: The Great Pyramid of Cheops and the look-up table.
This is a very large number that requires grounding in everyday experience. The Science Citation Index\(^3\) is a publication that approaches the human limit for usable information crammed into the smallest amount of space. Each page contains approximately 275 lines of 215 characters, and each inch thickness of paper contains 1000 pages (over \(5.9 \times 10^7\) characters). Assuming we could fit two queries of the above form and their responses on each line, each inch of paper would contain \(5.5 \times 10^5\) queries. Therefore, if a look-up table for queries of the above form were constructed, and all the pages were stacked up, they would be \(1.45 \times 10^8\) inches, that is, 2,300 miles high. This would require a volume of paper almost 200 feet long, 200 feet wide, and 200 feet high. In contrast, the Great Pyramid of Cheops was (at the time of construction) over approximately 760 feet square and 480 feet high (see Figure 2).

A reasonable defense against this objection is that computers can store data more efficiently than the printed word. It is possible in principle to construct a hard-disk array capable of storing our example look-up table. If we extrapolate slightly from current state-of-the-art, a disk capable of storing \(2.5 \times 10^9\) characters takes on the order of 100 cubic inches of volume and costs on the order of \$1000\). Therefore, \(8 \times 10^{13}\) queries at 100 characters per query requires 3.2 million disks, which would take up a volume of \(1.85 \times 10^5\) cubic feet (or a cube 57 feet on a side), and cost \$3.2 billion.

It is clear that our toy example only scratches the surface of the true size of a look-up table for a natural language. It is not too difficult to compile a list of 1400 fairly common concrete nouns (see the Appendix). It is not unreasonable to expect computers to be able to match the highest human ability, which would be 9 nouns per query (Miller [20]). The total amount of storage required for \(1400^9 = 2 \times 10^{28}\) queries, with 100 characters per query, 5 bits per character, is \(10^{31}\) bits.

If we were to store this on paper, it would require a stack almost \(10^{10}\) light years high. In contrast, the nearest spiral galaxy (Andromeda) is \(2.1 \times 10^6\) light years away, and the distance to the furthest known galaxy in 1988 was \(1.5 \times 10^{10}\) light years (Emiliani [11]). If we were to use hard-disks, it would take \(5 \times 10^{30}\) drives, which would occupy a cube 580 miles on a side. Even if we were to extrapolate wildly beyond the limits of foreseeable technology and conjecture that each bit could be stored on a single hydrogen atom, it would require almost seventeen tons of hydrogen. Yet our query set is still relatively small compared to the true number of reasonable natural language queries. It is not too difficult to compile thirty different adjectives or adjectival clauses to replace “largest”), which multiplies the resource requirement by thirty. Increasing the list of nouns to two thousand increases it by a further factor of twenty. Increasing the list of nouns to ten thousand increases it by a further factor of almost two million.

Therefore, it is fairly safe to conclude that it is not possible to pass the Turing test by simply using a look-up table. Where does this leave Searle’s Chinese Room gedankenexperiment? A look-up table certainly contains knowledge, but no understanding. Searle’s gedankenexperiment illustrates that understanding enables us to perform computations with a reasonable amount of resource usage; certainly less memory than is required to store a look-up table, and less time than is required to access one. This is a purely operational definition of understanding, and thus may not be satisfactory to a philosopher such as Searle.

\(^3\)Published by the Institute for Scientific Information.
who is more interested in a denotational definition, but I believe that any theory of cognition
that does not take this into account rests on unstable foundations.

Naturally, understanding is not a Boolean trait; one can have a little understanding, rather than being limited to no understanding or complete understanding. With a little understanding of the concept of size, one can reduce the look-up table for the example queries simply by sorting the list of objects in increasing order of size. We appear to understand such things not by memorizing lists of facts, but by grounding the abstract concepts of the objects involved in everyday experience, from which information we compute facts such as their relative size. I believe that understanding evolved as the most efficient way of storing, cross-referencing, and reasoning about large quantities of environmental data (that is, the most efficient way that can be realized within the design parameters of evolution).

One point on which Searle and I agree is that a digital computer can, in principle, simulate a human brain. The electrical behaviour of a single neuron is far from being well understood, but I would be surprised if it could only be described using continuous mathematics. My first objection is on general principle: most phenomena in the Universe appear to be discrete, although in many cases the quanta are so small that continuous mathematics is a good approximation to reality. My second objection comes from the experimental observation that the brain often continues to function when large numbers of neurons are damaged, and under conditions in which a large number of them misfire. I find it difficult to believe that this robustness would be possible if it were essential that every neuron compute a real value to infinite precision. Fixed precision is almost certainly enough, and probably not too large a precision. Any fixed precision computation can be realized by a discrete computation.

Searle feels uncomfortable with the consequences of the Church-Turing thesis. Computers can be realized with any medium that can represent Boolean values and compute binary conjunction and complement, including water pipes. In principle, a plumber could devise a sewer system that can simulate a human brain. Searle finds this absurd, but not for the same reasons that I do. There is far too much computational power in the brain to implement it as a sewer system.

Can we make a rough estimate as to how much computational power is contained in the human brain? Barrow and Tipler [2] give a range of $10^{10}$ to $10^{12}$ floating point operations per second, but they assume that the computation is taking place purely within the soma of the neuron. Conventional wisdom currently conjectures that a significant amount of the computation actually takes place within the synapses. Turing [42] made an estimate in the 1950's that with the benefit of modern knowledge seems optimistically low.

It is difficult to obtain reliable estimates of the number of neurons in the human brain. Shepherd [39] estimates that the human cortex has a surface area of about 2,400 square centimeters, and Rockell, Hendors, and Powell [31] report a uniform density of about $8 \times 10^4$ neuron per square millimeter, from which we can conclude that the number of neurons in the cortex alone is of the order of $10^{10}$. I assume that the bulk of the information passed from one neuron to another passes through the synapses; the number of such connections per neuron varies with the type of neuron in question, and is somewhat difficult to estimate, but a figure of $10^3$ connections per neuron is probably conservative. It is probably optimistic to assume that a pair of inputs to a neuron can be combined using a single floating point operation; even so, this implies that each neuron computes the equivalent of $10^3$ floating
point operations to combine the information input to it across its synapses. Combining these naive estimates with a firing time of $10^{-2}$ seconds per neuron, we see that the brain appears to have a processing power equivalent to at least $10^{15}$ floating point operations per second.

Searle’s water-pipe brain simulator is clearly something that can be imagined, but not constructed. Even under high pressure, water would flow so slowly in the pipes that in order to achieve $10^{15}$ floating point operations per second it would require on the order of $10^{15}$ floating point operations to be computed simultaneously at different parts of the sewer. Even if these results could be combined in a meaningful way in a fast enough manner, the sheer size of the system make it so unreliable that it would stand little hope of passing the Turing test. For that matter, could a computer do a better job? Current supercomputers can execute $10^{10}$ floating point operations per second, and it is estimated that we might reach $10^{12}$ by 1994 (Bell [3]). The brain appears to have more available computational power than a thousand of these hypothetical supercomputers.

This type of argument rests on shaky pedagogical ground because it is impossible to make an accurate assessment of the brain’s computational power given our current almost complete lack of understanding of the principles of brain style computation. Our estimate may well be too high or too low by several factors of ten. A second weakness is that technology is advancing so rapidly that, if Bell is correct and $10^{12}$ floating point operations per second are achievable by 1994, and advances in technology double computing speed annually, then computers may reach the $10^{15}$ floating point operations per second needed to rival the brain by as early as 2004.

One thing that we can be fairly certain of, however, is that the brain’s architecture is in a sense optimized for the type of computation that it is to perform. I say “in a sense” because there is little reason to believe that it is the absolutely optimum architecture (for evidence that biological computing systems are suboptimal, see, for example, Dumont and Robertson [8], Stork et al. [41], and Stork [40]). Rather, it is reasonable to believe that evolution has led to a locally optimal solution to a complicated optimization problem whose constraints include such factors as computational efficiency, heat loss, weight, volume, and nutritional requirements. Current computers, on the other hand, have architectures that are optimized within the constraints of current technology for the types of symbol processing problems for which they are used. It is hardly surprising that the architectures of the brain and the computer are radically different.

The simulation of the brain on a computer, then, is the task of simulating one model of computation on a second, architecturally different, model. The concept of one computer model simulating another is a key one in the theory of computational complexity theory. The Church-Turing thesis states that any reasonable model of computation can simulate any other one. Computational complexity theory has similar theses that state that these simulations can be carried out with a fairly small overhead in resource use; there is the sequential computation thesis (Goldschläger and Lister [15]), the parallel computation thesis (Goldschläger [13, 14]), the extended parallel computation thesis (Dymond [9], and Dymond and Cook [10]), and the generalized parallel computation thesis (Parberry and Schnitger [28]).

Nonetheless, each simulation requires an overhead in either hardware or time, often by as much as a quadratic in amount of that resource used by the machine being simulated.
Table 1: Number of synapses, and synaptic weight updates per second for some common computers to simulate a neural network (from [7]). The measurements for the MX–1/16 are projected performance only.

<table>
<thead>
<tr>
<th>Computer</th>
<th>Synapses</th>
<th>Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC/AT</td>
<td>$1.0 \times 10^6$</td>
<td>$2.5 \times 10^4$</td>
</tr>
<tr>
<td>Symbolics</td>
<td>$1.0 \times 10^7$</td>
<td>$3.5 \times 10^4$</td>
</tr>
<tr>
<td>VAX</td>
<td>$3.2 \times 10^7$</td>
<td>$1.0 \times 10^5$</td>
</tr>
<tr>
<td>SUN 3</td>
<td>$2.5 \times 10^5$</td>
<td>$2.5 \times 10^5$</td>
</tr>
<tr>
<td>MARK III, V</td>
<td>$1.0 \times 10^6$</td>
<td>$5.0 \times 10^5$</td>
</tr>
<tr>
<td>CM–2 (64K)</td>
<td>$6.4 \times 10^7$</td>
<td>$1.3 \times 10^6$</td>
</tr>
<tr>
<td>Butterfly (64)</td>
<td>$6.0 \times 10^7$</td>
<td>$8.0 \times 10^6$</td>
</tr>
<tr>
<td>WARP (10)</td>
<td>$3.2 \times 10^5$</td>
<td>$1.0 \times 10^7$</td>
</tr>
<tr>
<td>Odyssey</td>
<td>$2.6 \times 10^5$</td>
<td>$2.0 \times 10^7$</td>
</tr>
<tr>
<td>CRAY XMP 1–2</td>
<td>$2.0 \times 10^6$</td>
<td>$5.0 \times 10^7$</td>
</tr>
<tr>
<td>MX–1/16</td>
<td>$5.0 \times 10^7$</td>
<td>$1.3 \times 10^8$</td>
</tr>
</tbody>
</table>

Therefore, any computer doing a neuron-by-neuron simulation of the brain need not only be as computationally powerful as the brain, but dramatically more so. For example, contrast our figures on raw computing power above with experimental figures in [7] on simulating synaptic weight updates in current neuron models (summarized in Table 1). The reason why the computer figures are so poor (capable of simulating neural capacity somewhere between a worm and a fly, see Table 2) is that the raw computing power figures that we gave earlier completely ignored the extra overhead involved in the simulation. This is the real reason that we should abandon any hope of simulating cognition at a neuron-by-neuron level, rather than any philosophical or pedagogical objection.

Searle’s reasoning by analogy that there is little reason to believe that a simulation of cognition is not the same as cognition is unconvincing. Certainly a simulation of a fire is not

Table 2: Number of synapses, and synaptic weight updates per second for some common creatures (from [7]).

<table>
<thead>
<tr>
<th>Creature</th>
<th>Synapses</th>
<th>Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leech</td>
<td>$7 \times 10^2$</td>
<td>$2 \times 10^4$</td>
</tr>
<tr>
<td>Worm</td>
<td>$5 \times 10^4$</td>
<td>$2 \times 10^5$</td>
</tr>
<tr>
<td>Fly</td>
<td>$8 \times 10^7$</td>
<td>$1 \times 10^9$</td>
</tr>
<tr>
<td>Aplysia</td>
<td>$2 \times 10^8$</td>
<td>$2 \times 10^{10}$</td>
</tr>
<tr>
<td>Cockroach</td>
<td>$9 \times 10^8$</td>
<td>$3 \times 10^{10}$</td>
</tr>
<tr>
<td>Bee</td>
<td>$3 \times 10^9$</td>
<td>$5 \times 10^{11}$</td>
</tr>
<tr>
<td>Man</td>
<td>$1 \times 10^{14}$</td>
<td>$1 \times 10^{16}$</td>
</tr>
</tbody>
</table>
a fire, but for some purposes it does just as well. Pragmatically, if a simulation of cognition is not possible by reason of the fact that such a simulation carries too much overhead, then it is merely a matter of definition whether one calls it true cognition. Nonetheless, Searle has raised an important point that has deep ramifications. Strong AI proceeds by construction of a model of how the mind performs a task (such as understanding short stories, Schank and Abelson [33]), and then implementing (Searle would say “simulating”) that model on a computer. But what is introspection, if it is not simulating cognition? When one introspects, one constructs a conscious model of mind process, in essence a simulation of the mind. What right have we to believe that the products of introspection, which is no more than the construction of an internal simulation of mind, bear any real resemblance to the mind?

A crucial part of Searle’s argument is the concept of intentionality, which describes directed mental states such as beliefs, desires, wishes, fears, and intentions. Intentional states are related to the real world, but are not in one-to-one correspondence with it. One can have beliefs that are false, desires that are unfulfillable, wishes that are impossible, fears that are groundless, and intentions that cannot be realized (Searle [34, 35]). There are conscious intentional states, and unconscious intentional states, yet Searle devises a logic of intentional states that is mirrored in linguistics (Searle [35]). Searle comes to this conclusion from introspection, that is, by constructing a conscious and therefore by its very nature symbolic simulation of intentionality. If a simulation of intentionality is fundamentally different from intentionality itself, and if a conscious model of intentionality is merely a simulation of intentionality (rather than the real thing), then we are drawn to the inevitable conclusion that Searle’s logic of intentionality tells us little about intentionality itself. The inadequacy then is not in strong AI, which can take any consciously generated symbol-based model of cognition and turn it into a computer program, but rather with the analytical tools of cognitive psychology.

Searle argues that a formal program cannot have intentionality, and that intentionality is a crucial part of cognition. I am in agreement with the latter hypothesis, but in the former hypothesis Searle exhibits a strong anti-machine bias that he does not defend to my satisfaction. He is willing to accept that an animal has intentionality because it is the simplest explanation of its behaviour, but only because it is made of the same “stuff” as we are; apparent intentional behaviour from a robot is insufficient for him because (Searle [36, p. 421]):

“... as soon as we knew that the behaviour was the result of a formal program, and that the actual causal properties of the physical substance were irrelevant we would abandon the assumption of intentionality.”

Searle makes the assumption that intentionality is a property of the “stuff” of biological organisms, and cannot arise from the “stuff” of computers by execution of a formal program. We do not know enough of how intentional states are realized in human beings (Searle [35] considers the question briefly and dismisses it as irrelevant) to be able to say with confidence that formal programs can never exhibit them. It is reasonable to hypothesize that intentional states can arise in computational systems that are both sufficiently powerful, and properly organized. There is no reason to believe that intentional states arise in simple machines
such as thermostats, and it is a reasonable hypothesis that they do occur in higher primates and human beings. A reasonable hypothesis is that simple machines lack the computational power to have intentional states. That proper organization is necessary for intentional states is a properly conservative view; it is too optimistic to believe that they occur naturally in any computational system that is powerful enough to exhibit them. The real reason why computers do not have intentional states is that they are too simple to have them.

Searle is offended by the thought that a mere computer program could have intentional states, or think. But there is nothing “mere” about a computer program. The task of producing correct, fast, and robust software for even simple tasks (such as an airline reservation system) is incredibly difficult, as anyone who has attempted to write more than “toy” programs will agree. I have no philosophical problem with the hypothesis that there exists a program that when executed gives rise to cognition. However, there is a great chasm between that belief and strong AI. It may be that the program is far too complicated for us to understand. It may be that, when written as a formal program, it has resource usage that is beyond the power of the human race to provide. Simply because cognition can be realized in the brain (in some fashion that we do not yet fully understand) with a reasonable resource requirement is no reason to believe that its resource requirements as a formal program will be reasonable too. We have already seen that there is a large overhead in simulating one machine by another; it is often the case in computational complexity that a program for machine $A$ requires large overhead to implement on machine $B$, regardless of whether machine $B$ simulates the program for $A$ directly, or executes a completely unrelated program that produces the same results. The overhead of achieving cognition on a computer may be so large as to render the task impossible.

For example, it is clear that one cannot simulate intentionality by a Chinese Room algorithm, since such a look-up table must have entries for questions of the form:

“Would you believe that a $<$noun$>_1$ could be larger than a $<$noun$>_2$, a $<$noun$>_3$, a $<$noun$>_4$ a $<$noun$>_5$, a $<$noun$>_6$, or a $<$noun$>_7$?”

or of the form

“Which would you like to see most of all, a $<$noun$>_1$ a $<$noun$>_2$, a $<$noun$>_3$, a $<$noun$>_4$, a $<$noun$>_5$, a $<$noun$>_6$, or a $<$noun$>_7$?”.

The previous arguments about the size of the look-up table apply equally well here.

In summary, I believe that intentionality and cognition can in principle be obtained by executing the appropriate formal symbol manipulation program, but that there are other barriers that prevent intentionality and cognition from being realized that way in practice. To draw an analogy, the *Principia Mathematica* reduces mathematics to symbol manipulation, yet this is not how mathematicians do mathematics. Whilst they freely acknowledge that it is a necessary condition for any “proof” to be in principle expressable in formal logic, it is not necessary that it be so expressed. Mathematicians reason informally principally for the purposes of communication: a human being simply cannot understand a proof of any great depth and difficulty if it is expressed in symbolic logic. I believe that in the same sense, the mind can in principle be reduced to a symbol manipulation program, but the program would be far too long and complicated for human beings to understand (see also
Campbell [4, p. 109]), and that the reason why we don’t see thinking beings that are “mere symbol processors” is that the mind reduced to a symbol processing program may be too greedy of resources to be realized in the physical world.

We must also face the fact that it may not be possible to build a computer that matches the brain in speed, size, reliability, portability, power consumption, and ease of fabrication. It may be, as some biologists believe, that biology is the only way to achieve these goals simultaneously. But perhaps not. Perhaps the brain is the only way that such a computational device could evolve. It is an open question whether we can devise one ourselves, independent of the constraints of evolution. It is still an open question as to whether we could make such a device sentient. I believe that it is not possible given our current state of technology and current state of knowledge about cognition, but Searle’s arguments have failed to convince me that such a thing is in principle impossible.

Many believe that neural networks adequately refute Searle’s Chinese Room gedanken-experiment (see, for example, Churchland and Churchland [5]). Searle dismisses neural networks and parallel computation as not bringing anything new to the concept of computation as it applies to cognition. In a sense he is right; they bring nothing new to the 1950’s style of computability theory that he uses to bolster his arguments. However, parallel computers are more efficient at solving some problems than sequential computers are (see Parberry [24]), and the same can be said of neural networks (see Parberry [25, 27]).

The prime contribution of neural networks is that they use a computational paradigm that differs from the traditional Church-Turing one is self-evident in some cases, but this is not the death-knell for Computer Science as many of the proponents of neural networks would have us believe. Theoretical computer science has dealt with unconventional modes of computation for decades, as we will see later in this article.

The prime contribution of neural networks is the capacity for efficient computation of certain problems. The first computers were created in rough analogy with the brain, or more correctly, in rough analogy with a carefully selected subset of what was known about the brain at the time. Although technology has advanced greatly in recent decades, modern computers are little different from their older counterparts. It is felt by some scientists that in order to produce better computers we must return to the brain for further inspiration.

I believe that it is important to determine which features of the brain are crucial to efficient computation, and which features are by-products or side-effects of these (see Parberry [25, 26]). I do not believe that a computer that is comparable in computing power to the brain can be obtained by merely simulating its observed behaviour, simply because the overhead is too great. The general principles of brain computation must be understood before we try to implement an artificial system that exhibits them.

Computational complexity theory is a powerful technique that can be used to divine some of the general principles behind brain computation. However, the theory is in its infancy. Surprisingly, many apparently simple questions about efficient computation turn out to be difficult and deep. Whilst computational complexity theorists equate exponential resource usage with intractability and polynomial resource usage with tractability, in real life any resource usage that grows faster than log-linear in problem size is probably too large to be of any real use. It remains to develop the tools that can make that fine-grained a distinction
in resource requirements; for example, we cannot distinguish between problems with time requirements that intuitively grow exponentially with problem size from those that do not (see, for example, Garey and Johnson [12]).

Nonetheless, computational complexity theory often gives insights that may have profound philosophical ramifications. For example, many neural network researchers use a continuous model (i.e. one in which the neurons compute a continuous value). It can be shown in certain technical senses that if one assumes that neuron outputs are robust to small errors in precision, then their model is essentially the same as a discrete one within a “reasonable” overhead in resources (Obradovic and Parberry [22, 23]). More importantly, the same is true even without the assumption of robustness (Maass, Schnitger, and Sontag [19]).

The general framework used by neural network researchers is a finite network of simple computational devices wired together similarly to the network shown in Figure 3 so that they interact and cooperate to perform a computation (see, for example, Rumelhart, Hinton, and McClelland [32]). Yet there is little attention paid to how these finite networks scale to larger problems. When one designs a circuit to solve a given task, such as performing pattern recognition on an array of pixels, one typically starts with a small number of inputs, and eventually hopes to scale up the solution to real life situations. How the resources of the circuit scale as the number of inputs increases is of prime importance. A good abstraction of this process is to imagine a potentially infinite series of circuits, one for each possible input size, and to measure the increase in resources from one circuit in the series to the next (see Figure 4.

Figure 3: A finite neural network with 9 nodes and 2 layers.
Figure 4: A neural network family.
There is an apparent flaw in this abstraction, however. Since for every natural number \( n \),
every Boolean function with \( n \) inputs can be computed by a finite Boolean circuit (essentially
by using a look-up table, a formalization of Searle’s Chinese Room), our infinite-family-of-
finite-circuits model can compute any Boolean function, and so violates the Church-Turing
thesis. This can be remedied in one of three ways (among others). Firstly, we could insist
that each finite circuit in the infinite series be similar to its predecessor in the series in
the sense that a Church-Turing computer can compute the differences between the two.
This type of circuit is called a uniform circuit (whereas the former is called a nonuniform
circuit). Secondly, we could insist that the number of processing units in the finite circuits
grows only polynomially with input size. This would avoid the embarrassment of being able
to compute every Boolean function, since it is easy to show by a counting argument that
there are Boolean functions that require exponential size. Thirdly, we could insist that the
structure of each circuit be different from its predecessor, but that there exists a computable
set of circuits in which substantially more than half of the alternatives compute the correct
function. The first solution satisfies the Church-Turing thesis, and the other two do not.
This need not necessarily be a problem: the Church-Turing thesis is a model of reality, and is
not inviolate. The second solution is not particularly desirable, since it can make the design
of the circuits difficult in practice. The third solution is more appealing since although we
may not be able to compute the layout of each circuit, a subset of circuits chosen randomly
from the computable set of alternatives has high probability of turning out a circuit that
works.

Allowing computers access to a random source appears to make them more efficient than
a plain deterministic computer in some circumstances (see, for example, Cormen, Leiserson,
and Rivest [6, Section 33.8]). (Note that this is different from randomly choosing a deter-
iministic algorithm.) In this case, it is sufficient for the algorithm to compute the correct
result with high probability, say 0.999. Surprisingly, such a randomized algorithm can be
replaced with a nonuniform one with only a small increase in resources (Adleman [1]). This
principle can even be applied to probabilistic neural networks such as Boltzmann machines
(Parberry and Schnitger [29]).

Randomness and nonuniformity are two methods for reducing the resource requirements
of algorithms, both of which reach outside the confines of the Church-Turing thesis. The use
of randomness occurred to Turing [42], as did the possibility that the program of the mind
(if it exists) may be far too complicated for us to analyze. He also raised the possibility that
a computer could learn, as does a child. Computational complexity theory has started to ask
questions in this domain with the recent development of computational learning theory (see,
for example, Natarajan [21]). Of prime importance is the probably-approximately-correct,
or PAC model of learning, in which it is sufficient for the system to learn a response that
is with high probability close to the correct answer. Valiant [43] proposed the original
distribution-free PAC learning, and more recent versions include the Universal distribution
(Li and Vitanyi [18]).

The theoretical results described above demonstrate that randomness and continuous
computation do not offer a large increase in efficiency because they can be simulated with a
small increase in resources by discrete, deterministic computation. This does not, of course,
mean that we should use discrete computation to realize neural networks. As discussed
above, what is small overhead for a theoretician often becomes overwhelming in practice. Nonetheless, the theoretical results indicate that there is nothing new and alien in probabilistic and continuous computation, and that any philosophy based in them does not necessarily differ radically from a philosophy based on discrete, deterministic computation, contrary to all appearances.

Acknowledgements

I would like to thank George Mobus, Tim Motler, Kathleen Swigger, and John Young for discussions on the subject of this paper and comments on a draft of the manuscript.

References


