Brains, Minds & Machines: An Analysis of Software-Hardware Dualism

This paper will be an investigation of the relations of brains, minds and computers. If we manage to specify those relations, the answer to the question "can computers think?" will be trivial. First, I will start with a healthy dose of conceptual analysis. Then, I will spend some time on what a computer program really is. I will then move on to Turing. He will give us a robust definition of a program, as well as a behavioral test that could serve as a mark of the mental. I will then present Searle's argument against strong Artificial Intelligence and use the definition of a program to attack one of his premises. I will conclude by proposing possible ways forward.

Brains, Minds & Digital Computers

1. We think that brains are purely physical entities, and that brain events and states can be picked up using purely physical vocabulary. One might also use biological vocabulary, omitting some information about the physical system for the sake of generalizability.

2. I don't know what minds or mental states are, it depends on the view one adopts. This paper will touch on views such as Dualism, Functionalism, Behaviorism and Identity Theory.

3. I want to spend some time on what digital computers are. The prevailing view among Philosophers such as Searle and even among some Computer Scientists, is that there are programs and there is hardware. Programs are defined entirely by their syntactic structure and hardware is the physical thing out in the world. Under these definitions the same program can be run on different hardware, and the same hardware can be used to run many different programs. It is just a coincidence that a program that adds two numbers is being run on my laptop. It could have very well been the case that it was running on Searle's laptop. Notice that this is similar to saying that I could have had the same mental life, even if I had a completely different body. This conclusion is a result of Dualism, a view that has crept into all fields including Computer Science & Engineering.

What is a Program?

4. Programs are not over and above electrical circuits, they are purely physical states. When I write a program, I push keys on my keyboard, triggering signals that cause the hardware to transition from one physical state to another. This further causes physical changes on my monitor, visual changes that *I interpret* as symbols appearing. The thing I see is just the representation of the program I have written. The program is not an abstract entity that uses the hardware as a tool, but it is inseparable from the hardware; it is a physical state the hardware is *in*.

5. Imagine that by pushing keys on my keyboard, I have written a program that adds two integers. I see a sequence of symbols appearing on my monitor. Now I memorize this sequence and push the corresponding keys on Searle's laptop and the same sequence appears on his monitor. Now, have I written the same program?

6. Most people including Searle would say Yes! Since both programs were written by putting the same symbols in the same order, they are syntactically identical. If programs are defined entirely by their syntactic structure, once it is executed, Ege and Searle will be running the same program on their laptops.

7. One could also take a Functionalist approach. For any given input, if both programs return the same output, they must be the same program. A program that outputs (x+1) + (y-1), and a program that outputs (x+y) are functionally one and the same program, although they will be syntactically different. We don't care about the specific instructions and their ordering, or the events occurring at hardware level.

8. One could also take a hardware-state Identity Theorist approach, which I am advocating for. A program is nothing but a hardware state, and the running of a program is nothing but the hardware transitioning from one physical state to another. An algorithm or an instruction written on a blackboard or a text file is not a program, but a representation of a program. Symbols are just tools used by programmers to help them write programs that manipulate the hardware. It is the programmer that uses symbols. A computer does not operate with symbols; therefore, it cannot manipulate symbols. A computer also does not use ones and zeros, they are just symbols we use to represent the physical states of transistors.

Alan Turing & The Imitation Game

9. We know move on from conceptual analysis to bigger questions: can a digital computer think or understand language? Alan Turing proposes the Imitation Game, claiming that if a human and a computer were chatting with an interrogator purely through an exchange of symbols; if the computer managed to trick the interrogator into believing that it was the human; it must be true that the computer can think. I assume that both the human competitor and interrogator are sufficiently smart individuals with extensive world knowledge, so that it is a challenging behavioral criterion. One could also change the structure of the game to avoid certain objections, but the point is that the criterion is purely behavioral; it does not depend on the computations that are being carried out, their order, their syntactical features or what kind of hardware the program is bound to. The only thing that matters is that the interrogator judges the computer to be the human.

10. Due to the nature of the Imitation Game, Turing is certainly a behaviorist. But his views also paint a dualistic picture: agents can be conscious of their thoughts and act upon them[1], thoughts mentally cause other thoughts[2], and both brain states and mental states play functional roles[3]. His views also entail that brain states are irrelevant in determining if something is a thinking thing.

11. Going back to our discussion on programs, Turing would fall into the functionalist camp. He spends considerable time arguing that it does not matter if we use a mechanical computer or an electronic computer, as long as it can be used to mimic any discrete-state machine. Any universal machine that is running a sufficient program may beat the game. He urges that if we wish to find similarities between brains and digital computers, we should look for mathematical analogies of function.

> "To program a machine to carry out the operation A" means to put the appropriate instruction table into the machine so that it will do A.

12. I believe this statement requires a tremendous amount of analysis. Let us take A to be the operation of adding two numbers. Even as we utter the words "the operation of adding two numbers", we are defining the operation in virtue of its function (addition) and by referencing its inputs and output. It is also necessary that the machine *does* A, justifying Turing's behavioral criterion. Looking at the definition, it states that when we have to program the machine, we must both (1) come up with a sequence of instructions that will fulfill that function and (2) make sure that such instruction table is *appropriate* to the machine at hand. Turing's obsession with step (1) is understandable since that is the part that requires mathematical ingenuity, and (2) is a just a step that could be easily handled once the algorithm is designed symbolically. Although this form of software-hardware dualism sounds reasonable on pragmatic grounds, if we are to drag digital computers into our discussion of minds and brains, we must not discard step (2). The program is inseparable from the machine.

John Searle & Syntactical Programs

13. Searle thinks digital computer cannot think or have minds. I present his argument:

P1. Brains cause minds.
P2. Syntax is not sufficient for semantics.
P3. Computer programs are entirely defined by their syntactical structure.
P4. Minds have mental contents, specifically semantic contents.
C1. A program on its own is not sufficient to give a system a mind. [2, 3, 4]
C2. Brain functions do not cause minds solely in virtue of running a program. [P1, C1]

14. Searle argues against the analogy that the mind is to the brain as the program is to the computer hardware. To understand his reasoning, we must first understand his views on minds and brains which is expressed by P1. He thinks that brains *cause* minds and that the mind is *realized in* the biological structure of the brain. This has the consequence that there is something essentially biological about the human mind and that mental states such as consciousness, intentionality and mental causation are also purely biological phenomena.

15. Searle then puts forward P2 as a logical truth so that it cannot be challenged. I do not wish to challenge it. He thinks that symbols have no inherent meaning, so it does not matter which symbols you decide to use, or how you decide to order them; you cannot get meaning unless you *attach* a meaning to the symbols. Almost all AI researchers are defeated by this premise, as they argue that behaviorally successful systems do attach meanings to symbols: vectors or other complex mathematical objects that are constructed as representations of the outside world, as the machine causally interacts with the world. They say it is these objects that are being manipulated, not the symbols they are attached to! But if we are operating with Searle's understanding of a program, these objects also turn out to be symbols themselves: syntactic entities that cannot give rise to semantics.

16. A huge portion of this paper was devoted to my rejection of P3. I wanted to direct our attention to the distinction between a program (a hardware state) and a symbolic representation of a program (the instructions I write on a text editor). The latter is not the program itself; it is a mathematical representation of the instruction table one uses to take care of step (1). Step (2) is jointly handled by other programs such as the compiler and the CPU itself. One might even be tempted to think of the CPU as the pineal gland of a computer; the thing that converts symbolic instructions to physical hardware events. This dualism, although being a *necessary* dualism for the programmer's mental health, is a misrepresentation of what is really happening in a computer. From start to finish, all events in a computer's history are hardware events, which are purely physical events. Computer programs are not defined entirely by their syntactical structure, their representations are.

17. Let us see what follows from the rejection of P3. C1 depends on P3, so we cannot say that a program on its own is incapable of giving a system a mind. When we use the word "program", we are no longer referring to purely syntactic structures anymore, but the states of a physical object. But we still have not shown that programs are capable of giving systems minds. We also have to give up C2 since it depends on C1; it could be the case that brains cause minds solely by running programs. His third and fourth conclusions will also be rejected due their dependency on P3.

18. Even if we reject Searle's argument, we still have to find the mark of the mental. Do we simply accept the Turing Test and apply it to all things? Searle invites us to imagine Martians landing on earth. He assumes that we somehow decide that they have minds. He does not specify how. We then open their heads and its filled with green slime.

"The green slime, if it functioned to produce consciousness and all the rest of their mental life, would have to have the causal powers equal to those of the human brain."

This is consistent with his view that minds are caused by biological processes. One could think of Searle as a functionalist whose version of multiple realizability, for some reason only applies to biological things such as brains and green slimes; a tiny subset of physical things. I think he is simply a behaviorist. Whatever his mark of the mental is, he applies it to the Martians before opening their heads. It does not have to be the specific game that Turing proposes, but it could be another very complex behavioral criterion (might involve communication or an analysis of their movements). Even if his mark was intentionality or the existence of semantic content, he seems to be inferring them by observing the Martians in terms of their behavior.

Conclusion

19. Where do we go from here? Even today, the field of Artificial Intelligence is operating with task-specific Turing Tests. We say that a problem is solved if and only if there is a program that achieves the same accuracy as humans on that specific task. There are programs that can classify dog breeds, the sentiment of a text or simply solve a Rubik's Cube using a mechanical arm.

20. The problem is, all successful programs are task-specific function approximators. The program is shown a thousand images for each dog breed, and the function is optimized until it assigns the correct category to each image. We then hope that this function will generalize to images it has not yet seen. This approach (machine learning) did not work until the rise of artificial neural networks. They are programs written on computers just like any other program, mimicking the *structure* and the *function* of biological neural networks of the brain; passing the Turing Test on the specific task they are trained on. In my opinion, this calls for a deeper philosophical investigation of the mathematical function & physical structure of biological neural networks (a reconciliation of Functionalism & Identity Theory).

21. So, can computers think? I don't know. I don't even know if that is a good question. I have presented multiple views on the relations of brains, minds and computers; the reader could pick one of them and the answer will follow. After all, if the Turing Test is correct, they think only if you think that they think.