Conjuring Conscious Minds Out of Digital Soup

Derek Shiller

November 10, 2022

Introduction

Many cognitive capacities now restricted to humans may soon be shared by artificial systems. Large language models are already capable of verbal behavior that is difficult to distinguish from humans (Schwitzgebel 2022). The question of whether such systems could also have conscious experiences or whether they might merely mimic us will matter greatly, not just to law and policy, but also to how we see the future roles and responsibilities of our species (Shulman and Bostrom 2021).

Recent advances in artificial intelligence have utilized neural networks. Artificial neural networks perform similar computational work to brains. If these systems succeed in replicating human behaviors, the primary grounds on which we deny them consciousness may be a naked bias against the inorganic.

While I remain optimistic about the possibility of artificial consciousness, I am skeptical that contemporary approaches to building neural networks, based as they are traditional computer architectures and programming paradigms, will result in conscious systems. This paper will compare neural networks in biological and traditional digital environments and identify several differences that bear on our assessment of digital consciousness.

The core idea of this paper is that current approaches to computer design and programming may preclude consciousness. Although humans and computers share structures at some level of abstraction, those structures are implemented in a less substantial way in computers. They lack integrity. Instead, the parts of a computer that play the relevant functional roles are lost in a sea of digital soup.

I begin this paper by exploring functionalism and the multiple realizability of consciousness. I then argue for the need for further constraints on how systems implement functional organizations in order to satisfy the spirit of the view. I next describe some differences between brains and computers. Against this background, I present three plausible integrity-related constraints on what it takes to properly implement a functional organization and argue that these constraints provide reasons to be wary of consciousness in computers. I conclude with some thoughts on the prospects for satisfying these constraints in future systems.

Multiple Realizability and Functionalism

Philosophers of mind generally accept that consciousness is not a special product of the particular physiologies of human brains. Other animals are conscious, even animals with rather different sorts of brains. Non-biological systems could also be conscious. The traditional way to make sense of multiple realizability of mental states is with functionalism. Functionalism about consciousness is the idea that the functional organization of a system, the relationships between its possible states, determine whether it is conscious.

The neuronal basis of human cognition is a product of our evolutionary heritage. We are multicellular organisms. Our bodies are built from the sequencing of the DNA activations that guide cellular division and differentiation. Neurons are a category of cells capable of the complex interactions that allow coordination and computation. They are also living, growing, metabolizing things. Neurons began their evolutionary life suited to some other purpose and found a role in coordinating complex behaviors. Mutation and selection have crafted neurons for hundreds of millions of years, combining them into specialized circuits and modules that perform computational work as neural networks.

Neural networks are systems of interconnected nodes whose activities influence the activities of their neighbors. Neural networks are well suited to producing complex and flexible behavior. Biological neural networks utilize neurons, but artificial neural networks implemented as software in computers have demonstrated remarkable flexibility. Given the advances over so short a time, it seems that what is special about neurons is that they enable organic organisms to produce neural networks; it is neural networks that are special, not neurons.

It would be a surprising coincidence if, among all of the different ways in which it was physically possible to implement a neural network, the way that our ancestors stumbled upon was the sole way to allow for genuine consciousness. It would be surprising if neural networks weren't capable of consciousness unless they were made out of living, growing, metabolizing cells.

Functionalism provides a way around accepting such coincidences.

The functional organization of a system is comprised of the relations between the external inputs it receives, the states of its parts, and the behavioral outputs it produces. Inputs have an effect on the system's internal state. The overall state of a system can be subdivided up into the states of its parts, such that the effects of the inputs on behavioral outputs depends on how those inputs influence the state transitions of the parts, how those state transitions beget further state transitions, and how behaviors are determined by the states of the parts of the system.

Global workspace theory (Baars 1993) provides an example of how functionalism about consciousness could look in practice. Global workspace theory holds that consciousness is a matter of having the right architecture of information accessibility. The brain includes a network of neurons whose job it is to coordinate which information is shared among different faculties. This network constitutes a single repository of contents. Modules can request to admit information into this repository. Information in the repository gets broadcast widely to a number of faculties, including memory and introspection. Straightforwardly interpreted as a functionalist theory of consciousness, the global workspace theory predicts that any system that implemented a global workspace architecture would be conscious.

Abstract formal models help isolate what is important about functional organizations. David Chalmers (1996) develops an approach using vectors. On that approach, systems with n separate parts are characterized with a vector of n values (e.g. numbers), with each value representing a state of a particular part. Inputs and outputs are also represented with values. We model functional organizations as sets of transition functions: functions from input, state vector pairs to output, state vector pairs.

A system will be said to possess the functional organization characterized by a model if there exists a way of carving up the inputs it receives, the states it can occupy, and the outputs it produces so that there exists an isomorphic mapping between those and the inputs, states, and outputs of the model that obeys the transition functions.

Neurons are fantastic building blocks. They can be composed into structures that interact in all sorts of different ways. The structures that they compose satisfy abstract requirements for consciousness. It is these structures that make the brain conscious. Neurons make these structures possible, but any other building block capable of implementing a neural network is up to the task.

Functionalism holds that consciousness turns on functional organization, not on the medium from which that organization is constructed. Just as neurons are excellent building blocks, so too are electrical circuits. Any suitable building block can be used to create structures that constitute consciousness.

The Proper Implementation of Functional Roles

There are many ways of carving up every system. For any given formal model, no mapping is possible under most carvings. It is hard to define a single canonical carving that will deliver the verdicts we want. This pressures functionalists toward liberalism: the traditional position is that all it takes to implement a formal organization is for there to be an isomorphic mapping under at least one proper carving of the system.

If all carvings are proper, then any isomorphism is sufficient. This makes it far too easy to implement a functional organization. Further constraints on propriety are needed to ward off unintuitive verdicts.

The oldest version of the worry is attributed to Paul Hinckfuss (described in Lycan 1981), who evocatively suggested that the fluid dynamics in a pail of water might come to briefly occupy the relevant functional structure of a human brain. This is problematic if it is obvious that a pail of water would not conscious no matter how it sloshes about. While it is relatively unintuitive that an ordinary pail of water could ever be conscious, it is also dubious to put too much trust in our intuitions about rare configurations of water molecules. Subsequent critics have been more willing to bite the bullet on the possibility of an occasional conscious pail of water (Sprevak 2018), so long as it has peculiar internal dynamics.

A second and more popular version of the problem suggests that it isn't just some remote possible configuration of water molecules we need to worry about, because nearly any material object can be gerrymandered to fit whatever functional organizations are thought to be necessary for consciousness.

The classical presentation of this problem comes from Hilary Putnam (1988). Putnam argued that for sufficiently complex systems, there nearly always exists some way of carving its states that can make it trivially easy to satisfy any functional organization. The idea is that if, given any input, a system transitions through a series of unique states from a unique starting point, then we can find an agreeable carving by grouping states together so as to be isomorphic to any formal structure. Large numbers of unique states provide a blank canvas on which any formal structure can be drawn. Multiple realizability provided a significant motivation for functionalism, but if it says everything is conscious all the time, it has gone too far.

Chalmers (1996) describes two properties that together provide the requisite complexity for Putnam's argument. The first property he terms a 'clock', an aspect that constantly updates to a never-before-seen form. The second property is a 'dial', an aspect that can take and retain any of a limitless number of forms. If each part of a system contains a clock and a dial, then each part will go through enough different states (clock and dial combinations) to allow for isomorphic mapping of all counterfactual contingencies. The trick is to map sets of possible inputs to sets of dial settings so that each individual dial setting is unique to each set of inputs, and then map clock and dial combinations to the states that would result from those inputs.

However, critics have noted that the complex systems identified by Putnam, while they may get all the counterfactuals right, don't support the kinds of causal relations we expect to exist between parts of a functional organization (Chalmers

1996; Chrisley 1994). Our assignment of states is likely to be such that, given the original configuration, the values of some clock and dial combinations in some parts of the system are correlated with the clock and dial combinations of some other parts of the system, but neither causes the other.

A third version of the problem suggests that there are some systems that liberal functionalism will say are conscious that it shouldn't. The problem isn't that these things are obviously not conscious, so that it gets them wrong, but that it violates the spirit of functionalism to treat them as such.

Consider a grid of mirrors that is placed facing another mirror, such that each mirror in the grid displays a reflection of the whole grid. If we write a distinct number on each mirror then it will contain both its number and a deeply nested reflection. This reflection will take time to form: before light has made a trip, each grid mirror reflects just its number. A moment later it reflects that number and the whole grid of numbers. A moment later it reflects that number and one level deep of nested reflections, and so on.

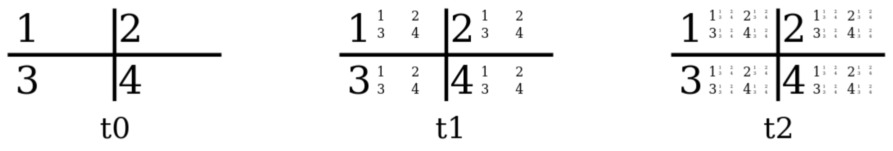

Figure 1: Nested Reflections

At each moment, the state of each distinctly numbered mirror in the grid is unique over the history of the system and sensitive to the total prior state of the grid. With such a setup, we could carve out not only any states we want from disjunctions of surface reflectance patterns, but also support all of the causal relations that exist in the model with causal relations in the grid. To find an isomorphism, we simply map the input states of the formal model to some set of initial numbered states of the grid and map each subsequent state of the formal model to the disjunction of states of the grid that would result from those initial configurations.[1]

It is as obvious that the mirror grid would not be conscious as anything. This suggests a need for further constraints. The problem isn't that we shouldn't think that the mirror grid is conscious, but rather that a good functionalist theory should not say that it is. If the letter of our theory says that the grid is conscious, then we need to reformulate the theory to better fit the view.

The problematic isomorphisms depend upon gerrymandering states into whatever shape is necessary. There are many ways to limit gerrymandering. These ways involve identifying further features of an implementation scheme that prevent gerrymandered systems from properly implementing functional organization. Going forward, I'll assume that a 'proper implementation' is interpreted as an implementation *suitable for consciousness*, given the right functional organization. (The question of what it takes for a system to implement a computation in general is related to but separate from this issue. There is little reason to think that the features that allow a machine to intuitively count as a computer for a certain algorithm should be the same as the features that allow a system to count as implementing the functional role of consciousness.)

One way to refine propriety involves enforcing specific interpretations of inputs and outputs (Chalmers 1996). Perhaps the inputs of a proper implementation involve real perceptual events and the outputs involve real behaviors. The pail

[1]Chalmers (1996, 328) anticipates a system like this and suggests that it would need to be too complex to be possible. If the smallest reflections were 1/10,000 of a millimeter and the grid were a kilometer, we could fit approximately nineteen nestings of a nine-paneled grid, meaning we'd be limited to functional organizations with no more than nine parts and nineteen state transitions. Perhaps this is not enough to implement the functional role of a moment of consciousness. Larger grids and smaller images are surely possible, but I'm wary of making the claim dependent on physical possibility. The issue isn't that liberal implementations allow for minds where there shouldn't be, but that liberal implementations don't live up to the spirit of functionalism. The conceptual possibility of mirror grids shows that. (Chalmers agrees, and suggests that we might need further constraints on the kinds of causation that are appropriate.)

of water and the mirror grid have no sensory faculties and so perceive nothing. Nor do they move any limbs. Their complex dynamics might implement the functional structure of a human brain, especially with gerrymandered states, but the inputs and outputs of the implementation correspond to non-sensory non-behavioral states. Requiring suitable inputs and outputs will limit proper functional states to systems with external behaviors a lot like ours.

There are a few problems with this solution. For one, enforcing specific inputs and outputs won't distinguish functionalism from behaviorism (Putnam 1988). If Putnam was right, then any system that presented the right inputs and outputs could be carved up in some way so that it fits any functional role. If we're relying on the inputs and outputs to exclude improper implementations, then internal states do no significant work.

A second problem is that it is a heavy cost to deny consciousness to human beings without sensory and behavioral capacities. A human brain can continue to function, at least for a while, when it is cut off from sensory inputs and motor controls. Such a system would lack the inputs and outputs needed to properly play the same functional roles as other human brains under this approach, but this seems like no grounds to deny it consciousness.

Godfrey-Smith (2009) presents a third problem. For any system, it is conceivable that we could set up a massive and complicated interface that converted perceptual inputs into perturbations of the system and converted states of the system to behavioral outputs. By attaching the appropriate external interface to any pail of water (or grid of mirrors), we could ensure that it has the right sensory inputs and behavioral outputs. The implementation of such an interface can't be what makes the difference to how these systems feel.

A second way to refine propriety rules out arbitrary collections of unique states by requiring that the system may be divided into spatially separated (e.g. non-overlapping and/or contiguous[2]) and logically independent parts (Chalmers 1996; Schweizer 2019). This constraint places limits on what kinds of gerrymandering of internal states of a system as a whole are permitted. Insofar as gerrymandering is acceptable, it must be constrained inside the each of the system's parts.

In proposing this, Chalmers notes with regret that this will block some plausible systems from properly implementing functional states. It is too conservative. As we will see below, structures in a computer program may not be so easy to pull apart and spatially segregate, so this proposal alone should invite skepticism about digital consciousness.

A third way to refine propriety invokes a similarity requirement on proper implementation of individual states (Godfrey-Smith 2009). Insofar as a state of a model is properly mapped to a massively disjunctive collection of system configurations, the disjuncts must be mutually similar. This prohibits eclectic configurations from counting as the same state.

Such a similarity requirement begs to be supplemented with a requirement that the configurations mapped to distinct states be comparatively dissimilar. These requirements together would mean that configurations mapped to the same state are more alike and configurations mapped to different states are less alike.

The point of running through this debate is to highlight that the notion of proper implementation is important for a plausible functionalist theory, and anything-goes approaches skew the view. The proposals surveyed here aimed to avoid unintuitive implications. They do so mainly by adding constraints on the integrity of functional organization's realizers. Integrity has two components: 1) the parts of a system must be distinctive entities, e.g. they must have natural

[2]Chalmers (1996, 325) says "distinct physical region", but is vague about the specifics.

boundaries (which need not be precise, or spatial) that explain their separation from the other parts of the system. 2) The states of a part of the system must be naturally distinguished from each other, so that which configurations fall under which states is principled, not arbitrary. Chalmers' separateness requirement touches on the first. Godfrey-Smith's similarity requirement touches on the second.

Integrity constraints make sense for functionalism. Functionalism says that consciousness rests on having parts interacting in particular ways. It is natural to expect that the parts require some independent existence with a substantial identity. Once we accept constraints on proper implementation, those constraints can take on a life of their own. We should assess candidates in part by their intrinsic appeal, and not just the work they can do for us in avoiding gerrymandered minds. If there are reasons to move in the direction of integrity constraints in order to avoid commitment to excessively liberal functionalism, we should be open to exploring the significance of integrity more broadly.

I see no reason to expect a priori that the true constraints on proper implementation are the minimum constraints necessary to avoid gerrymandered minds. Later on, I'll present three integrity-related constraints on proper implementation that have a similar intrinsic plausibility.

Biological and artificial networks

In assessing the significance of potential constraints for digital consciousness, it will be helpful to sketch some similarities and differences between brains and computers. I'll summarize briefly how brains and computers work and how neural networks in contemporary hardware differ from neural networks in neurons.

Brains

The human brain is composed principally of neurons organized into separate structures. Neurons share a basic role and morphology, but differ to suit their particular functions. They include a cell body and long thin branches to other neurons. Neuronal branches act as conduits for electrical signals. Axons receive signals from other neurons. Dendrites send signals to other neurons. When a signal is generated, typically in response to receiving signals, electrical fields travel down the length of the branch. The signals stimulate the release of neurotransmitters that can cross over to stimulate the axons of other branches. Neighboring neurons will receive these neurotransmitters and potentially propagate their own signals in response.

Whether a neuron transmits an electrical signal down its dendrites ('fires') is a function of the signals it has received. At its most basic level, if a neuron receives sufficient signals in a short period of time, it will also fire. This basic pattern can be complicated in a variety of ways. Some neurons act to suppress the firing of their neighbors. Neurons also affect the physical properties of their environment; through these activities they can modulate the behaviors of whole regions.

In humans and other mammals, the cortex is the structure that houses most of the neurons responsible for visual processing, long-term planning, language generation and motor control, along with many other cognitively sophisticated functions. Neurons in the cortex are organized into layers and are connected layer-to-layer to comprise columns. These columns are integrated into larger circuits that allow cycles of signalling. Neurons are fairly noisy, often firing unprompted, it is patterns of firing within and across circuits that allows for cognition.

Neuronal signals travel fairly slowly; millions of neurons contribute to computational processing in parallel. A sensory signal will lead to a widening net of firings that parse the features of the signal, calculate a representational content, and decide on a behavioral response over the course of hundreds of milliseconds. Those activations can feed back to earlier stages of processing to allow coherences between different parts to ramify and incoherences to dissipate.

Computers

Computers operate primarily through the interaction of a processor (CPU) and memory (RAM). Computation-intensive programs may also delegate much of their work to a separate specialized processor (GPU or ASIC). The CPU directs the movement of electricity through electrical circuits. Some specialized circuits, binary cells, switch between states in a manner that allows them to store data values. Since binary cells switch between two states, it is often useful to think of data as binary numbers. A single binary cell stores one 'bit' of data with two possible values. Eight cells store one 'byte' of data with 2^8 possible values.

The CPU contains a number of specialized sequences of binary cells, 'registers' that store the data on which it operates. One register in a CPU, the instruction set, houses the bit representation of the operation the CPU is to perform. Another register, the program counter, contains the bit representation of the location in memory of the next instruction to be performed – after it interprets an operation, it can fetch the next operation from memory and store it in the instruction set. RAM contains vastly more binary cells, and those cells are optimized for longer-term storage. Many operations involve moving data from RAM into registers, moving data between registers, and returning data from the registers into RAM. Others perform logical or arithmetic operations on the data in the registers.

Programs are loaded into RAM when they are launched. Part of the process of building a program is converting the characters of a text file into sequences of atomic instructions, represented by series of bits, that operate within in the design of the instruction set circuitry to direct the CPU. These instructions are ferried from RAM to the instruction set and performed.

One difference between brains and computers relates to use of memory in programs. The earliest computers required manual reconfiguration to implement different programs. The standard von Neuman architecture, in contrast, involves configuring the CPU to behave differently depending on the bits stored in its instruction set. This allows us to store programs in the same way we store data. The difference between programs and other data lies in how the bits are used.

The CPU/RAM architecture of computers contrasts strikingly with biological brains. Whereas RAM and registers are general-purpose data stores that can be used to enact any of a variety of operations as encoded in the instruction set, neurons in a brain are more specialized. There are no long sequences of data stored in a brain, fetched, and manipulated in accordance with general rules. Neurons are far too slow for this to be feasible. (Whereas CPUs can perform billions of sequential instructions every second, it takes a neuron milliseconds to fire.)

A second difference is the robustness of represented states. In computers, states are stored as the exact electrical properties of circuits. In brains, it is the existence of physical neural connections and the patterns of firing of neurons in a neuronal circuit.

Computers are discrete, generally performing one precise instruction at a time in accordance with a rigid logic and in a tight sequence, as dictated by the ticking of an internal clock. Bits don't randomly flip, and the difference between one bit can fundamentally alter how the computer functions. Data can be stored reliably in precise formats at precise locations, so programs can march through long sequences of memory chains. Brains, on the other hand, are continuous and far more messy. Firing takes time, neural activations overlap, and things like the coincidence or non-coincidence of different activities make a significant difference to the resulting behavior.

A third difference concerns parallelization. The instructions for a program are carried out in sequence. The illusion that many programs are all running continuously and simultaneously can be created by the rapid switching between threads containing the instructions for different programs, but one operation is carried out at a time.

A number of forms of parallelization have been implemented to improve computer performance. Resources may be duplicated and instructions may be cleverly pipelined to allow sequential operations to be carried out simultaneously. Modern processors often make use of multiple cores (and many computers have multiple processors), so that programs can divvy up work across several copies of the basic CPU architecture, with their own program counters and instruction sets. CPUs can also create sets of instructions that GPUs or ASICs can apply to thousands of different data points simultaneously. Computer behavior results from long chains of sequential operations, whereas neural processing is distributed over a large number of short chains of neurons.

These forms of parallelization generally fall well short of the parallelization in brains. Between pipelining and instruction parallelization, no more than a few dozen operations can be performed at once in any ordinary single core. CPU cores generally are relatively few in number (though supercomputers can use hundreds of thousands of total cores) and programs must be carefully designed to divide up work across them. GPUs are constrained to allow for specific forms of parallelization and can only perform a limited set of identical operations on a dataset.

Artificial Neural Networks

Modern advancements in artificial intelligence have come to rely on implementations of neural networks. Nodes in a neural network function something like neurons in a brain: they switch between states of activation and inactivation as a result of the activities of their neighbors.

Artificial neural networks might theoretically be produced in a number of ways. In practice they are created with standard digital computers. Nodes in a network are represented in memory and in processor registers as the states of binary cells. Those bits are manipulated in the processor in accordance with program instructions.

Nodes in abstract neural networks are generally separated into layers and activations across nodes is calculated layer by layer. A node's activation at one time is dependent on a function applied to the activation of the nodes it is connected to in the previous layer.

A particular node's activity level is generally represented by binary cells formatted as a *float*, a binary representation of a number with bits used for a sign, an exponent, and a digital fraction. A layer's activations can be represented by lists of floats in blocks of memory. Weights between layers can also be represented as lists of floats. The relative locations of the activation floats and the weight floats within their respective sequences determine which must be combined to derive the activity levels of the next layer.

The various parts of a network are separately represented. A program needs to keep track of connections between nodes indefinitely but only needs to calculate activation strengths momentarily. When a network is run, the connection strengths between layers are fetched and combined with input activations via series of matrix multiplications and the application of activation functions.

Three Integrity Constraints

Functionalism receives some of its plausibility from the fact that it allows for the multiple realizability of mental states, including conscious experiences. However, if the requirements for properly playing a functional role are relaxed, functionalism will over-generate, allowing some complex systems to count as implementing just about any functional organization. In order to place reasonable limits on multiple realizability in accordance with the spirit of functionalism, we must supplement functionalism with constraints on proper implementation.

We've seen several possible constraints so far. On the one that Chalmers favored, the realizers of functional roles must correspond to spatially distinct parts of the system. On the alternative that Godfrey-Smith favored, different instances of the state of a functional part must resemble each other. There are many more possible constraints. Some of these might help with the trivialization threat. Many independently plausible constraints won't. Once we open the door to the existence of constraints on proper implementation to solve one problem, we should be open to the existence of more.

I'm somewhat skeptical about the capacity for either philosophical intuitions or scientific experimentation to clearly and unambiguously resolve which constraints are correct. Assuming consciousness is a mind-independent phenomenon and our judgments don't constitute the truth about it, I see no reason why we should have a reliable innate grasp on where it arises. Instead, I think we are best served by generating a list of plausible constraints and applying them to produce varying degrees of certainty regarding the consciousness of artificial systems.

If, on every plausible constraint we can come up with, a system implements the same functional role as a human brain, functionalists should regard it as conscious. The greater the number of plausible constraints that a system fails, the more skeptical we should be. Ideally, we should avoid producing artificial minds of uncertain status insofar as will be less able to know what kinds of regard we owe them (Schwitzgebel and Garza 2020).

In this section, I explore three further integrity-related constraints – contrast, causal integration, and continuity – on the proper implementation of a functional role whose plausibility should lead us to question whether current approaches to implementing artificial neural networks would allow them to be conscious.

Contrast

The degree of *contrast* of a structured system is a measure of the extent to which the parts of the system are distinguished by their intrinsic properties both from the other parts the system and from other nearby objects. High contrast systems have parts whose identities are recognizable without appeal to the functional roles they play. The parts of the system form a relatively natural carving.

The component parts of a car – the windshield, the steering wheel, the carburetor, the battery etc. – have a high contrast with the other parts and with the surrounding atmosphere, the driver and passengers, and any luggage. Simple principles of material continuity, composition, and connectedness can be used to distinguish one part from another and from other things. An engineer who knew nothing about the function of a car could start identifying the boundaries between its pieces before they figured out how it functioned.

A solid block of concrete has no macroscopic parts with much contrast. Every block contains within it a number of statues depicting a perfect likeness of Julius Caesar. Those statues have no contrast with the surrounding concrete. We can project boundaries into the concrete to define these shapes, but those boundaries don't correspond with any intrinsic properties of the block. There might be historical or functional bases on which to distinguish parts (we might rest a ball on top of the block and then distinguish the column supporting the ball from the rest of the block). However, these bases wouldn't depend on intrinsic properties of the block, so they don't contribute to contrast.

The neural structures relevant to consciousness that exist within human brains may not have as high a degree of contrast with the rest of the brain as the parts of a car, but have much more contrast than the statues in a block. There are grounds on which neural structures can be distinguished based on their anatomy and connections. Neuroscientists working in the 19^{th} and 20^{th} centuries divided up maps of the brain based on cell architecture. More recent techniques have allowed neuroscientists to follow connections and build maps of circuitry. It is complex work, but analysis of a full map of neural connections would likely reveal the circuits that play the brain's functional roles.

A global workspace, for instance, is likely to be wired in peculiar ways that would be recognizable as anatomically distinct. It seems likely that on the basis of their connections, neuroscientists could identify the circuits comprising the global workspace before they were able to step through and figure out exactly what it did (Deco, Vidaurre, and Kringelbach 2021). This is not to say that neural structures would have an obvious role to anatomists, but it is likely that a fine-grained evaluation of the brain could identify many of the parts that form the components of functional organizations.

Skepticism around Hinckfuss's pail attests to the plausibility of contrast as a constraint on proper implementation. The particles in a pail of water whose properties might be mapped onto functional states corresponding to a human brain will, like the shapes in a block, have a low degree of contrast. This is especially true if the states of the parts are massively disjunctive and are selected by way of their gerrymandered satisfaction of functional roles. It is plausible that no pail of water could be conscious. The lack of contrast is a plausible part of the story why.

The data representations of nodes and connections in a digital representation of an artificial neural network are hard to distinguish from other data stored in RAM. The parts of a neural network are stored as sequences of binary cells in memory but as a sequence of binary cells they need not appear any different from the other data. Text, numbers, program instructions, image data, all are just sequences of bits. An informed investigator might make an educated guess as to the contents of some sequences based on patterns in the data (e.g. how common the bit encoding of the letter 'e' is) but such guesses are imperfect.

The data created by a program need not be stored in a continuous sequence, let alone in any particular location. Programming languages make frequent use of pointers, which allow one bit of memory space to indicate a particular piece of data is stored at another location in memory. Data objects with multiple properties will frequently not store the values of those properties in one section of memory, but will point to different places for each property. Data that are related to each other within the program, such as connection strengths between different layers of a network, may be stored in distant stretches of memory and be separated by other kinds of data, including data for other programs. (Each layer's weights will be stored in a relatively contiguous block of memory, but different layers may be stored separately.) The memory storing a neural network may thus be widely distributed and be linked together by series of pointers.

There are no clear divisions in memory between a block assigned to one program's data and another. Binary cells are associated with addresses through which they are accessed. Programs store the address at which data structures start along with the length of those data structures (if it is not fixed), but there is nothing in the physical structure of memory marking the beginning or end. Somewhere in memory, there is a sequence of bits that happens to represent the address at the start of a data structure, and, for variable-length data, there is another sequence of bits that happens to represent the length of that data structure. Somewhere in memory, there is another address that signifies that those addresses hold the relevant data regarding the location and boundaries of the data structure. It is only by tracing these addresses back through many references that we are able to see how a program uses the bits that provide their meaning as addresses and not as integers, characters, or other data.

The distribution and genericness of data entails that the divisions between data structures are heavily dependent on functional properties of the program. In order to see what sequences of memory define a data structure, we have to follow along how the program works.

The divisions between the parts of a data structure in a computer are in this way intimately tied to their place in the functioning of the program. The data structures have no contrast on their own, separate from the behavior of the program, and the behavior of programs is complicated and highly defuse, depending on the code for the program, the operating system, and the physical structure of the hardware.

It is easy to overlook the lack of contrast in programs because programs aren't written in the same language in which they are run. Reading a text file for a program conveys some of the program's structure graphically. For instance, a variable may be defined with a name and a type, and the same orthographic string located between the same curly braces will signify a variable with the same content. But CPUs can't understand programming languages as they are written – they must be translated into a native format, and that translation process removes much of the contrast suggested by the program code.

Causal Integration

Given a functional role, the *causal integration* of a structured system is a measure of the degree to which the system's functional parts play their roles by virtue of an innate causal profile. Systems that aren't causally integrated may only act as they do because of the operation of some external power. Books and bombs both can have immense effects on the world, but books do so only through the way that their readers respond to them. They are not causally integrated. Bombs, in contrast, are causally integrated because their effects are produced by their internal chemistry.

Structures within brains that are built out of neurons have the power to influence one another. Neurons help to generate the functional dynamics of brain regions through their capacity to produce signals and their receptiveness to the signals of other neurons. The functional roles played by mental states composed of neurons are ultimately grounded in the causal powers of the neurons that comprise them. These neurons directly influence the states of the neurons they connect with, and through these interactions parts of the brain influence each other.

The same abstract structures that have causal power within human brains might be implemented in systems where they lack causal powers and are instead given their functional role by an external manipulator. We might reconstruct a human brain structure through intricate diagrams of neurons, their states, and their connections on a massive blackboard. Teams of grad students might go through and update the diagram so it evolves over time in correspondence with a real brain. The marks on the blackboard have no power themselves. They sit unchanging until attended to. They serve as notes that lead the researchers to make appropriate updates. The markings on the blackboard might have the same structure and dynamics as a human brain, but it is plausible they would not be a proper implementation on account of their internal powerlessness.

The structures in a computer that represent nodes in a neural network also do not have any power themselves. The representation of the state of a node and of its connections are contained within sequences of binary cells. The same binary cells with the same contents could be just as easily put to any other purpose. They can be fetched and manipulated by any program granted access to that region of memory space. The node representations can do nothing all by themselves. Instead, those representations are used by a program to represent a neural network. A structure of node representations only plays a functional role insofar as it is used for that end by a program.

It might be argued that the right computer structures within an implementation of a neural network aren't just the sequences of binary cells representing the node states and connections in memory, but also all of the various pointers that point to those structures, all of the instructions and other data in memory required to give meaning to those pointers and depict operations to be performed on them, and also the architecture of the CPU that allows the operations to be performed in the right way (among other things). This broadly inclusive interpretation of the relevant structures in the computer would allow the causal powers of the structure to be internal by incorporating within the structures the parts that do the work.

The inclusive approach raises its own issues, so while it is a reasonable move, it only shifts the burden.

One issue with this proposal is that other parts of the program that manipulate data structures mostly serve a general purpose, so different structures will largely overlap in the portion of their subparts that perform causal work. Many of the instructions needed to step forward through a neural network will be basic routines for loading and manipulating data. These routines will be used in much the same way to manipulate every structure in the network. While not necessarily fatal, this substantial degree of overlap between parts replaces one threat to integrity with another. Substantial degrees of overlap blur the distinctness between different parts of a system.

A second potential issue is that incorporating the machinery used for updating a network into the parts of the structure will result in a clear bifurcation between the operator and the data inside of that part. As we just saw, the distinctive component of these substructures, what makes one part different from the others, remains passive. Insofar as the combined structure has the powers it does, it has them because of an internal system of, essentially, manipulating notes. The notes contain the structures that distinguish the state from others, but they are powerless in themselves. This too is strikingly different from the way that human brains work in a way that should raise some doubts.

Continuity

The continuity of a system is a measure of the extent to which its parts persist over time under identities that are separate from their functional roles. Systems without continuity may have a constantly evolving set of subcomponents built from different materials; insofar as the parts of a non-continuous system persist over time, the slices of each part at different times have a unity through their shared positions in a broader functional organization.

The neural structures that perform the sorts of functional roles in human brains underlying consciousness are mostly consistent over time. Neurons are specialized for specific tasks by their internal structure and by their connections. They maintain their structure over time, occupying the same positions in the brain and being composed out of the same materials moment-to-moment.

The same structures in human brains might be implemented in systems where they lack continuity. Consider a machine that takes as an input a human brain that has been frozen into a single state within a preservative medium and produces as an output a fully new human brain frozen in another brain state. This machine would disassemble the input brain and construct the output brain out of new atomic materials that reflected what state the input brain would have momentarily occupied were it not frozen. We might route the output brains back around to form the machine's inputs, so that it produced a constant succession of new frozen brains reflecting the states that a human brain would naturally occupy as its internal dynamics evolved over time.

The succession of brains produced by this machine could be said to have the same functional architecture as a human brain. However, this requires some interpretive creativity. The dynamism of the functional architecture is accomplished over multiple brains; the parts that play each functional role would jump from place to place and would be constantly composed of new materials. We might identify a network that forms a global workspace and hold that the brains' global workspace consists in relevant network in the latest frozen brain. As new brains are produced, the global workspace repository will jump from brain to brain. The faculties that to which information is broadcast would also exist in future brains.

Even though each frozen brains' parts would have causal connections with the previous brains' parts through the operations of the machine, their separateness poses an intuitive challenge for consciousness. It is plausible that this is an improper implementation of the architectures of consciousness because the material parts that play a functional role in a proper implementation must retain that role over time.

The regions of memory that are available for use by a computer program depend on the runtime context. When a program starts up, the operating system carves off a section of memory space for the program. The operating system then provides additional sections when the program's needs grow. Which sections of memory are provided to the program depends on which sections are currently in use. If different programs are run, or the same programs are run in a different order, they'll have access to different regions of memory. This means that when a program stores data about a node's state in binary cells in memory, the circuits that store that data change from run to run.

While running, programming languages typically handle function calls by creating data structures ('frames') that are stored separately in regions of memory. The same function, when called multiple times, may use different sections of memory for each frame. The memory space allocated to each frame can contain some basic data, they also contain pointers to other memory locations for data objects that have variable sizes. Frames located in different memory regions will use distinct binary cells to store data that is internal to a function and may point to data stored in different regions of memory. So even when a function is called multiple times within the course of a single run of the program, it is likely to store its data using distinct binary cells. The arrays of activation states of nodes will not be supported by any specific section of memory (except by chance) but may jump around to binary cells that at other times store a wide variety of different kinds of data.

Complex neural networks require so many computations that they are typically run on GPUs or similar processors. GPUs can efficiently apply the same algorithm to multiple data points at the cost of some restrictions on memory access. [Note to Derek: How does memory allocation across GPUs affect this argument?]

Most neural networks are layered and do not need to keep track of node state in layers over time. Instead, they need to remember the connection strengths between layers and the activation state of the current layer. Even when a sequence representing a list of node activation values is stored in memory and updated within the scope of a single function call, whether the result is stored in the same place will depend upon fine details of the code. Depending on the neural structure in which the nodes are embedded, there may be no reason to keep any record of its activation state after its effects on the next layer have been computed. The memory that was allocated to recording each node's state may be freed up for future use for or may be immediately used for other purposes. This means that the bits storing node state don't change over time the way the state of a neuron does. They are allocated to their role in storing a node state and given a single unchanging value for their life.

As a result of the complex and dynamic factors influencing memory use, the binary cells that at one point store the state of a node may at another time store the state of a different node, or store a different kind of data altogether. If the program does not reuse the same memory regions to record the state of a node from moment to moment, then the structures in a neural network representation that are responsible for consciousness will lack continuity. Each time a node is marked as activated, the representation of the activation will involve memory at a different location composed of different circuits.

If continuity is a constraint on what it takes for a structure to properly play a functional role, then ensuring the continuity of data address usage may be critical for consciousness.

Building Conscious Systems

The three integrity constraints raise doubts about the potential consciousness of AI systems built using current tools and techniques. These are doubts about potential consciousness, not behavioral capacities: present tools may allow us to build systems with every behavioral indication of phenomenal states. This makes these concerns particularly pernicious: we may be easily misled into attributing consciousness to the systems. The first sophisticated AI systems may be something like Searle's Chinese room: they convey all the outer appearances of phenomenal experiences without the requisite internal machinery to actually feel them.

The constraints present no principled barrier to artificial consciousness. There is no reason to think that we can't build systems whose internal parts display contrast, causal integration, and continuity. Securing all of these constraints would require rethinking how AI systems are implemented. The processors and memory on which systems run could be redesigned so as to include robust structures of transistors grounding the functional states of the mind. The particular architectures of nodes and connections might be physically implemented in circuits, rather than inferred from computations on matrices of floats.

Some exploratory steps have already been taken in this direction. Neuromorphic computers implement some of the structural features of brains in transistors. IBM's TrueNorth (Merolla et al. 2014) and Intel's Loihi (Davies et al. 2018), for instance, contain circuits of processors directly connected to one another, that can compute and transmit calculations to each other without being instructed to do so by a central instruction processing system. Such properties suggest that a descendent could satisfy the integrity constraints with an artificial brain that looks much more like our biological brains than present computers.

[Maybe add something about optical neural networks?]

Neuromorphic computers are promising insofar as they implement neural structures in hardware, but the arguments here don't cast doubt on the possibility of conscious systems that are fundamentally dissimilar from us. Different sorts of computational systems, including systems not using neural networks, could satisfy contrast, causal integration, and continuity. Neural networks are a promising way to implement the right architecture, but they are not necessarily the only way. Perhaps future alternatives will allow for more efficient implementations of consciousness architectures.

It is not clear that non-traditional processor architectures will win out. The traditional approaches developed in the 40s, 50s, and 60s may be ideal, or they may enjoy enough of a head start to never be worth replacing. However, if we care about reducing ambiguity in AI consciousness, and if we take functionalism seriously, then we should strive to reduce the doubts raised by violations of integrity constraints. This will require redesigning hardware and software to make conscious relevant structures more robust.

Bibliography

Baars, Bernard J. 1993. *A Cognitive Theory of Consciousness*. Cambridge University Press.

Chalmers, David J. 1996. "Does a Rock Implement Every Finite-State Automaton?" *Synthese* 108 (3): 309–33.

Chrisley, Ronald L. 1994. "Why Everything Doesn't Realize Every Computation." *Minds and Machines* 4 (4): 403–20.

Davies, Mike, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang

Cao, Sri Harsha Choday, Georgios Dimou, et al. 2018. "Loihi: A Neuromorphic Manycore Processor with on-Chip Learning." *Ieee Micro* 38 (1): 82–99.

Deco, Gustavo, Diego Vidaurre, and Morten L Kringelbach. 2021. "Revisiting the Global Workspace Orchestrating the Hierarchical Organization of the Human Brain." *Nature Human Behaviour* 5 (4): 497–511.

Godfrey-Smith, Peter. 2009. "Triviality Arguments Against Functionalism." *Philosophical Studies* 145 (2): 273–95.

Lycan, William. 1981. "Form, Function, and Feel." *The Journal of Philosophy* 78 (1): 24–50.

Merolla, Paul A, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, et al. 2014. "A Million Spiking-Neuron Integrated Circuit with a Scalable Communication Network and Interface." *Science* 345 (6197): 668–73.

Putnam, Hilary. 1988. *Representation and Reality*. MIT press.

Schweizer, Paul. 2019. "Triviality Arguments Reconsidered." *Minds and Machines* 29 (2): 287–308.

Schwitzgebel, Eric. 2022. "The Computerized Philosopher: Can You Distinguish Daniel Dennett from a Computer?" 2022. http://schwitzsplinters.blogspot.com/2022/07/results-computerized-philosopher-can.html.

Schwitzgebel, Eric, and Mara Garza. 2020. "Designing Ai with Rights, Consciousness, Self-Respect, and Freedom."

Shulman, Carl, and Nick Bostrom. 2021. "Sharing the World with Digital Minds." *Rethinking Moral Status*, 306–26.

Sprevak, Mark. 2018. "Triviality Arguments About Computational Implementation.” In *The Routledge Handbook of the Computational Mind*, 175–91. Routledge.