Structure and Dynamics in Implementation of Computations

Jacques Mallah[1]

Abstract: Without a proper restriction on mappings, virtually any system could be seen as implementing any computation. That would not allow characterization of systems in terms of implemented computations and is not compatible with a computationalist philosophy of mind. Information-based criteria for independence of substates within structured states are proposed as a solution. Objections to the use of requirements for transitions in counterfactual states are addressed, in part using the partial-brain argument as a general counterargument to neural-replacement arguments.

1 INTRODUCTION

Intuitively, knowing that a physical system implements a given computation tells us that the structure and dynamics of some part (or subsystem) of the system are similar 'in some way' to those that define the computation. To make this precise and meaningful, well-defined criteria must be used for such implementations. Each computation that a system implements then provides a partial characterization of the structure and dynamics of the system.

The primary application that requires that such characterization of systems must in principle be possible is computationalist philosophy of mind. That is the idea that there are some computations that, if implemented, give rise to mental states either inevitably or due to natural laws.

Defining the exact criteria for implementation has proven to be a more difficult problem than it might at first appear, because without appropriate restrictions on how to map underlying systems to computations, even trivial simple systems could be seen as implementing virtually any computation. [1,2,3]

The relationship between physical systems and implemented computations is many-to-many: After all, many systems can have subsystems that are similar in some way, a given system can have many subsystems, and a given subsystem might have many aspects to its dynamics. The fact that a variety of physical systems would be able to implement the same computation is called *multiple realizability*. Although different mappings are associated with different computations, there is no role for observers' preferences about which mapping to use for the question of whether or not a given computation is implemented by the system (since at least one successful mapping would be possible if it is), or when the whole *spectrum of computations* implemented by the system for *all* possible mappings is considered.

Along with the main proposal for implementation criteria, a few variant options will be suggested. Different options may be useful for different purposes, as different aspects of system structure and dynamics may be more important for each purpose; e.g. determining to what information-processing uses an analog machine can be put to as opposed to determining whether a computer system is similar enough to a system known to be conscious that it would have similar consciousness.

[1] Medical Physicist, Charleston Radiation Therapy Consultants, Suite B-1, 3100 MacCorkle Ave SE, Charleston, WV, 25304, USA.
Email: jackmallah@yahoo.com

An analogy for the use of such variant criteria is that a solid object has both a shape and a composition. One observer might say of the object "It is round"; while another observer might say "It is rubber". Despite these different characterizations, neither property depends on any external observer.

The main proposal is intended to be the one capable of being applied to philosophy of mind, as well as for most general purposes that may arise, especially regarding digital computers. However, it is possible that a detailed computationalist theory of the human mind would call for a more analog-focused implementation criterion to appropriately characterize brain structure and dynamics.

2 NAÏVE CRITERIA FOR IMPLEMENTATION

The following criteria for implementation will be used as a baseline to which modifications will be added to avoid the problems with unrestricted mappings between the underlying system and the computation:

1) There must exist a mapping M between the states of the underlying system and those of the computation (known as formal states). A given underlying state maps to at most one formal state, and need not map to any formal state.

2) If the underlying system is (or *were* to be) in any state that maps to a state of the computation, then its time evolution must be such that the next different state of the computation that the underlying system state will be (or *would* be) mapped to (if any) is given by the appropriate *transition rule* of the computation.

The second criterion includes the requirement of proper behavior in counterfactual states, not just in actual states; this requirement was not always present in early ideas of implementation, and was part of Chalmers' [1] counter to Putnam's critique of the implementation concept [2]. This requirement leads to some counterintuitive implications that have been the basis for critiques of computationalism, as will be discussed below in sections 14-15.

Define a *run* of a computation as a pass through the sequence of formal states appropriate to the initial condition while implementing the computation. The system will be said to implement that run. To characterize what a system is actually doing, it is important to know what runs it implements; for example, applied to computationalist views of mind, runs of the same computation that are associated with different sequences of states could give rise to different mental states.

A run need not be unending. If a system goes to an underlying state that is not part of the mapping and does not resume entering formal states, the run ends. A run of a computation can also end by entering a formal end state.

The verb 'run' may also be used as a synonym for 'implement' when referring to a computation. A *computer* is the part of the physical system that is mapped to the states of the computation.

3 SIMULATION PRINCIPLE

If the formal states for a computation are treated as though they were physical states and the transition rules for it as though they were physical laws, then one computation could be said to implement another.

One would expect the following *simulation principle* to hold, as it does so far for the above criteria: If a physical system implements computation #1, and computation #1 implements computation #2, then the physical system must also implement computation #2. In such a case, computation #2 is called a *simulation*, and computation #1 is called a *virtual machine*. The simulation principle should be interpreted as a description of a property of the implementation relation, rather than as an additional criterion, except when otherwise stated.

4 CLOCK-AND-DIAL FALSE IMPLEMENTATIONS

The traditional view is insufficient: A clock (something which reliably increments a non-repeating variable) and a stationary dial with sufficiently many states would implement any computation under the naïve criteria. The argument is summarized here; further details can be found in Chalmers' paper [1]:

Map the initial state of the clock and the actual state of the dial into the initial formal state, and use that clock state and other possible dial states to extend the mapping to counterfactual formal states. For each subsequent clock state, map the clock and the chosen dial states to the proper subsequent formal states. In this way, any desired transition rule can be reproduced.

To introduce notation that will be useful later, this mapping can be represented in symbols as (C, D) ➔ F, where C is a clock state, D is the dial state, F is the formal state of the computation, and "➔" means "maps to". The dynamics for the clock can be represented as C(t+1) = C(t) + k, where k > 0; here t refers to the time at the initial step, and t+1 is the time at the next step. Since the dial doesn't change, D(t+1) = D(t), and F(t+1) relates to F(t) according to the desired transition rule.

5 THE COMBINATORIAL STATE AUTOMATON

Chalmers [1] correctly noted that the lack of structure of computational states in the naïve criteria is what allows the clock and dial to implement any given computation; this illustrates that both structure and function are relevant to characterizing the nature of a system.

He proposed the Combinatorial State Automaton (CSA) as a type of computation that would resist inappropriate (invalid) mappings. It adds the following to the naïve criteria:

3) The formal state label is given as combination of substate values, specified in an ordered list.

This also has the advantage that 'input' is easily incorporated into the CSA concept by allowing some substates not to be covered by the transition rule, but rather to have values that can be determined by outside influences.

4) Each substate must be '*independent*' of the others. Chalmers took this to mean that its value must depend on physical variables that are in a different region of space.

The above independence requirement prevents 'clock and dial' mappings from implementing computations with substates.

6 PROBLEMS WITH SPATIAL INDEPENDENCE

However, the CSA criteria do not rule out all of the obviously false implementations [1,5,6], as pointed out by Chalmers himself. Consider a system in which the information that determines the values of all of the substates is copied at the next time step into each of the spatial regions. To make a mapping such that this system would falsely be seen as implementing any computation (within the limits of the number of substates), just map the regions to the desired formal substates based on the transition rule.

This would require an exponentially growing memory capacity for each region, so it is not practical for long runs or big computations, and Chalmers left it at that as being sufficient until a proper fix is found. But the problem is far from insignificant: If not fixed, it is enough to trivialize the notion of implementation, and thus, invalidate computationalism.

In fact, it is not difficult to find examples of systems that would suffer from such false implementations. One such example is any system that performs a weighted sum of real numbers, such as a weighing scale does if more than one object can rest on the scale. Suppose there are N weights, each with twice the weight of the previous, which can be placed on the scale, each in a spatially distinct spot. The presence or absence of each weight then can be inferred from the total deflection of the scale. The position of the scale pointer can then be mapped to any desired one-bit function of an N-bit string. For example, a prime-check function that equals 1 if the N-bit string represents a prime number in base 2 notation, and 0 otherwise, could be used in the mapping. What's more, there is no need for the absence of a weight to correspond to a 0 and its presence to a 1; this can be different for each bit. Thus, the empty scale would 'compute' that 00010001_2 (seventeen) is a prime number. This is a computer with only two distinct time steps, but in any other sense it is a nontrivial computation. Clearly, this is a false implementation.

A second problem with the use of spatial separation as an independence criterion is that it is too closely tied to physics. It certainly would not be acceptable to use a physical property such as energy, for example, as part of the fundamental definition of what a system computes. It is understandable that time may play a role in the definition because a computation is an initial-value problem, and one can generalize the role of physical time to the evolution parameter in any mathematical model of an initial-value system; it is far less obvious how the idea of spatial separation could be similarly generalized.

Also, in quantum physics (assuming no collapse of the wavefunction, or in other words, the many-worlds interpretation (MWI)), macroscopic objects have approximately well-defined positions only relative to other objects. A system could then be in a macroscopic quantum superposition such that the spatial positions of objects would overlap in the overall wavefunction. It is not plausible that this would destroy a computation since the internal structure and dynamics of the system would still be, in many ways, much the same as in the classical case. e.g. A computer that finds prime numbers would still do so even if its wavefunction overall is a spatially spread-out function; measuring its position is not necessary. This argument holds even if the MWI is not true, because the definition of implementation, being a mathematical notion, must not depend on empirical physics.

7 STRUCTURED STATES

Before proposing a better notion of independence, another aspect of implementations of computations should be dealt with, and it will play a role in the criteria for valid mappings. While a CSA has structure in the form of the ordered list of substates, that may be a poor reflection of the structure of the underlying system, which could for example involve a function on a space of more than one dimension. While there is no reason that every implementation of a computation must reflect the full underlying structure, if implementations characterize both structure and function, it would make sense that some implementation mappings would reflect more structure than a simple ordered list.

Therefore, a computation will be specified by a structured set of variables (called substates), and a transition rule. Such a computation will be called a Structured State System (SSS). Substates must be independent in a sense to be defined below. All of the naïve implementation criteria still apply.

The structured state variables may form an ordered list, or may have additional structure; for example, they may be arranged into a rectangular matrix. Any label that the identity of a state depends on will be referred to as an index; for example, for states f(x) which are a function on the space x, then for *each* value of x there is a different substate f(x), and 'x' is an index label.

The structured state framework allows for both analog and digital computations, including computations with a continuum of variables (a field). An analog transition rule may be stated as a differential equation involving time derivatives.

Substates may have no particular structural relationship to each other, in which case it will still be convenient to list them in an ordered list, but the lack of structure should be noted. It is also possible for more than one variable to be considered part of a single composite substate.

Input substates have values not determined by the past-to-future transition rule for the time step being considered, and are possibly influenced by forces outside the scope of the computation. Usually they are substates before the transition, but after the transition, some substates could also be inputs; e.g. a button that can be pressed by outside forces at any time. Any substate which has a value determined by those of past substates can be called an *output* and often serves in turn as an input for future time steps.

Transition rules can apply at the substate level, not necessarily based on a global time step, so it need not matter if states on one side of a computer complete a localized transition before those on the other side or vice versa. Thus, two otherwise identical spatially distributed computers traveling at opposite relativistic velocities could still implement the same computation.

In the following section, for convenience, the term 'physical system' will be used for describing the underlying system. However, it should be understood that one SSS may implement another, treating the formal states for the underlying computation in the same way as physical states are used in an implementation, and the transition rules for it as though they were physical laws.

A physical system is treated here as being itself an SSS, with a fixed state structure. For classical mechanics, the state structure could be the positions and velocities of the particles. For quantum mechanics it could be the wavefunction in the position basis, and for quantum field theory, the basis of field configurations that are functions of position. The criteria for implementation of computations could also be applied to mathematical structures as the underlying system, which have been proposed as the underlying basis of reality, e.g. by Tegmark [4].

8 BASIC INDEPENDENCE

The false implementations cited in section 6 above must be ruled out. Since the problems arise when the mapping rather than the values of underlying physical variables does the work of binning the combinations of substate values into the desired function, the allowed mappings could be denied access to the resources to do so. This suggests requiring limits on what information can be obtained from knowledge of the underlying variables that are mapped to a substate; that is the basis of the proposal given here:

BI #1) The first rule for basic independence is thus that it must not be possible to find the values of any of the substates (at the previous time step) that controlled what the value of a given substate is (at the current time step) from knowledge of the physical states that are mapped to the formal value of the given substate and from knowledge of the system dynamics, except when the current formal value itself provides enough information.

BI #2) In addition, the values of other substates at the same time step should not be revealed by knowledge of the physical variables that are mapped to a given substate. This provides the second rule for basic independence.

If the second rule were not required, then false implementations could be obtained in systems where a function of one physical variable is calculated, if the invalid mapping claims it is a function of many formal substates. For example, if variable A takes on integer values in the range $(0,\ldots,2^{N-1})$, it can be mapped to N bits, $B_1,\ldots,B_N$. There are many such mappings. Let C be another variable and $C(t+1)=f(A(t))$, where t indicates one time step and t+1 the next. This might then falsely implement $C(t+1)=g(B_1(t), \ldots,B_N(t))$, where for example g could be a prime-checking function for base 2 numbers, where the mapping from A to the bits is chosen to make the values of g correspond to that function. To summarize, this example invalid mapping is:

One variable ➔ Many variables

$A(t)$ ➔ $[B_1(t),\ldots,B_N(t)]$

$C(t+1) = f(A(t))$ ➔ $g(B_1(t),\ldots,B_N(t))$

Simple function of one # ➔ Complex function of many bits

This second rule for basic independence is a generalization of Chalmers' spatial independence criterion. It almost reduces to spatial independence in a case where physical variables depend on different spatial regions and do not contain the information needed to reveal the values of the substates in the other regions, except that it could allow one physical variable to partially determine many substates. Spatial non-overlap can be a useful rule of thumb.

The two rules for basic independence eliminate clock-and-dial mappings, because the clock and dial physical state which any substate depends on would reveal all formal values of the previous and current substates. They also rule out the other false implementations discussed above, such as the binary prime number checker discussed in section 5, in which the variables mapped to each substate record information that would reveal the values of the previous substates that determine its values. Yet they are too conservative in some ways, as will be seen below; basic independence provides *sufficient*, but not necessary, conditions.

FS) One modification to the basic independence criteria is suggested by considering a pointer whose position has a slight 'fine structure' dependence on substate values that basic independence would forbid, but where that dependence is not exploited by a convoluted mapping. Such a pointer does not discard information that is normally discarded by the digital substate, but there still is much about the dynamics that is reflected by the computation. To allow this type of system to

implement a digital computation, bin the values of the underlying system variable (in this case, the pointer position) into *non-overlapping ranges* (if this can be done in a well-defined way based on a continuous or discrete intrinsic, inherited, or transferred label as defined below) such that within each range it is mapped to the same formal state value. If each range is then treated as a single value of the underlying variable, less information would be available from knowing those values than is available from the actual value of the pointer position. Use only this reduced information when testing for independence. Because the ranges are non-overlapping, this could not be used for clock-and-dial-style arbitrary mappings.

9 INHERITANCE

It is sensible to allow certain cases in which formal substates share all of the physical variables that they depend on (contrary to the second rule for basic independence) when those variables carry indices that mark them as functions on a grid or space of more than one dimension: Mappings should be allowed to reflect the multi-dimensional aspects of the structure and function.

For example, consider a hypothetical underlying physical system that is a set of bits labeled by a pair of integers plus time, B(i,j,t). Suppose that only one bit has the value 1 at any given time; the rest are 0. The mapping is from these bits to a pair of integers I(t),J(t) in which I=i and J=j for the nonzero bit.

The second rule for basic independence implies that I and J would *not* be independent substates, because they both depend on the *same* set of physical variables – *all* of the bits on the grid. (The first rule may be violated as well but this depends on the transition rule.) But, intuitively, they should be considered independent; i and j are distinct aspects of the underlying structure of this system, not something imposed by the mapping.

To take this type of structure into account, treating labels on a space (here, the grid) almost on equal footing with the values of physical variables, which labels the value of a formal substate depends on must be taken into account. It is useful to define new technical terms, 'inherit' and 'disinherit', to deal with this issue.

If a label is *inherited* by a computation substate, then the formal value of the substate depends on how the physical state values are distributed among physical states with different values of that label. If there is no such dependence, then the label is *disinherited* by that substate; swapping or permuting the values of physical variables whose labels differ in only disinherited indices would leave that substate's value unchanged.

For substate independence purposes, knowledge provided by the values of disinherited labels for a given substate is to be disregarded when evaluating whether values of other substates could be revealed by knowledge of the physical variables that are mapped to the given substate.

In the example of the grid of bits (Fig.1), with a mapping to I(t) and J(t) as described above, the value of I(t) depends only on the i-label, and not on the j-label. Swapping the j-value rows would have no effect on the value of I(t). Thus, I(t) inherits i and disinherits j. Similarly, J(t) inherits j and disinherits i. In this way, I(t) and J(t) are independent, just as if I(t) had depended on one physical variable and J(t) had depended on a different variable, even though in reality they depend on the same set of bits.

There may be variables that both substates inherit. In addition, an index can be suppressed if other indices suffice to calculate substate values. Suppose that the physical system consists of bits on a 3-d grid plus time, B(i,j,k,t), and the mapping is such that each pair (i,j) is paired with a unique value of k, namely k_{ij}, for each of the bits that are included in the mapping. As before, only one bit at a time among those used in the mapping is nonzero. As before, let I(t)=i and J(t)=j for the nonzero bit. I(t) inherits i and k.. Knowing k, since it is unique, reveals the value of j. But one could still calculate I(t) and J(t) if the k-values were not used. In this case, I and J should still be considered independent (barring the special cases); I inherits i and disinherits j, and J inherits j and disinherits i, with k suppressed.

Functions of labels can themselves be labels that the physical variables are in turn functionals of, as are fields in quantum field theory. Inheritance or disinheritance can be established by considering permutations of either type of label.

10 CLASSICAL COMPUTERS IN QUANTUM WORLDS

Classical computation performed by quantum systems is a very important subject because all known systems are actually quantum, and it should be studied in depth with full awareness of constraints on implementation mappings. Inheritance could allow quantum systems to implement some computations that the classical version of a quantum system would perform, using for example the relative state (many-worlds) interpretation. In such a case, particle positions for a given relative state might be mapped to formal substates, each inheriting the label for position of the appropriate particle and not the others. (Quantum field theory, which is a more realistic model of reality, could be handled in an analogous way with inheritance of field values at key positions.)

For example, relative to an environment state B representing one of the decoherent branches, a (simplistic) mapping might be:

$$X_1(t) = C(B) \int dx_1\, dx_2\ |\Psi(x_1,x_2,B,t)|^2\, x_1$$
$$X_2(t) = C(B) \int dx_1\, dx_2\ |\Psi(x_1,x_2,B,t)|^2\, x_2$$

with transition rules such as

$$d^2X_1/dt^2 = k\,(X_2-X_1), \qquad d^2X_2/dt^2 = k\,(X_1-X_2)$$

X_1 inherits x_1 but not x_2, and X_2 inherits x_2 but not x_1; they can thus be independent although both depend on the *same* set of wavefunction physical variables.

Of course reality is not so simple, as decoherence is never complete, so a more realistic mapping would specify a specific value or range (which can vary with time) for each of those variables that determine which branch of the wavefunction is being considered. B could depend implicitly on x_1 and x_2 via the dynamics (not explicitly via the mapping) and other restrictions might also be needed (such as restricting the wavefunction to have a given form). The analog formal states given above would typically have to be binned into digital states to give exactly reliable transitions, and (as usual) such a run need not go on forever. It should be remembered that these mappings are not necessarily the only ones that might be important for considering classical computations performed by quantum systems.

11 THE SIMULATION PRINCIPLE AND LABELING

For the simulation principle that "If a physical system implements computation #1, and computation #1 implements computation #2, then the physical system must also implement computation #2" to hold, the formal substates of the computation should only be labeled with indices if each such index derives from indices (of the underlying system) inherited by the substates.

For example, suppose the physical system consists of a set of bits on a 1-d grid plus time, B(n,t). A mapping is proposed from this system to formal substates consisting of a set of bits on a 2-d grid plus time, F(i,j,t), where each individual formal bit depends on only one physical bit; the only difference is in the labeling of the bits. There is no problem with the independence of the bits. However, there would be a problem if one tried to argue that the relabeled system implements a computation involving substates I(t),J(t) where they each inherit from the corresponding label on the 2-d grid. That is not a legitimate mapping because the proposed substates are not independent when considering the labeling structure of the underlying system; both i and j depend on the value of the underlying system index n.

By contrast, suppose that the underlying system consists of a set of bits on a 4-d grid plus time, B(i,j,K,L,t). Map this to a set of substates on a 2-d grid plus time, F(i,j,t) where i and j are the same as before and each value of F depends on bits at various K-values and L-values, but only one value each of i and j. Suppose that F inherits K but disinherits L, treating L-values symmetrically. F now inherits i,j,K , but if it depends on all K values there is no point in labeling it with a K-dependent index. F(i,j,t) is a legitimate set of substates. For a specific example:

$$F(i,j,t) = \Sigma_L \, \Sigma_K \, K \bullet B(i,j,K,L,t)$$

The formal system described these substates might then simulate another computation involving a grid of bits with i,j labels.

Because many systems (at least when considered on an intermediate level, as virtual machines) don't have much intrinsic label structure, such as a group of transistors which can be assembled together in largely arbitrary ways, it is often convenient to be able to label similar components without implying that the label can play a role in determining substate independence. Such labels of convenience will be flagged using a # sign.

12 TRANSFERENCE

The restriction on formal state labeling given above may be too strong. With it, the set of allowed labels can contract by going from an underlying system to an implemented computation, but could not expand. For example, with it a Turing machine consisting of a single long tape S(n,t) and the position N(t) and state H(t) of an active head could not be legitimately mapped to a set of bits on a 2-d grid plus time.

These kinds of mappings are not like those generally thought to be relevant to human cognition, which map physical variables to neural nets, but might be relevant to artificial intelligence or to attempts to model a possible structure underlying (and implementing) known physics. Such mappings should be allowed if doing so would permit a better characterization of the structure and dynamics of the underlying system, since such a characterization is what the implementation concept provides.

One way to allow such mappings is allow some of the structure provided by the transition rules for an already legitimate computation (which is structure actually present within the system) to be 'transferred' to the substate labeling for a mapping which can then be used to implement another computation. In allowing cases like this, it must be verified that trivial systems can not be considered to perform nontrivial computations.

The *simulation principle* as stated above will not automatically hold in such cases. To preserve it, it should be turned into a *prescriptive* rather than descriptive statement; this is an additional relaxation of the independence criteria.

As an example of 'transference', suppose two of the underlying system substates, X(1,t) and X(2,t), control which among the other substates are used or updated by a subsystem; the transition rule is

$$X(3 + X(1,t) + C\,X(2,t), t+1) = [\text{some function}]$$

where $0 \le X_1 < C$ and $0 \le X_2 < (N-2)/C$, and so for every combination of X(1,t) and X(2,t)) a different variable among the X's within the appropriate range would be updated at this time step. X(1,t) and X(2,t) would have their own transition rules. (Such situations are common in artificial programs written for electronic digital computers, such as to display an image on a rectangular screen.) This suggests a sort of two-dimensional structure among the X's within that range. This 2-d structure can be allowed to 'transfer' to the label structure in a mapping:

$$X(3 + X_1 + C\,X_2, t+1) \rightarrow Y(X_1, X_2, t+1)$$

Another example in which the transition rules might be used to guide allowed label structure is for a cellular automaton (CA). In a CA, each substate's transition rule within a subset of substates depends on only a limited number of other such substates in a largely symmetric way. For example, suppose the substates are bits B(x#,y#,t). The bits are implemented by some other underlying physical mechanism, and are not intrinsically on a 2-d grid from any inherited physical index, so the # notation is used here for the labels of convenience. They are hooked up in such a way as to implement a cellular automaton, in which each bit's future state depends on its current state and on those of its nearest neighbors in the x# and y# labeling scheme. The resulting physical system may look messy, with wires going in various directions and looping around each other, and the transistors physically arranged in no particular order, but it implements the transition rules and each bit is independent of the others as defined above. The logical structure imposed by the transition rules can be allowed to 'transfer' to the label structure; thus, the # signs may be dropped and this can be considered as bits on a 2-d grid.

Transference could occur with continuous variables as well. The path length position of a bead along a wire might transfer to a continuous variable, since it implements constrained dynamics.

13 GENERALIZATION AND VARIANT CRITERIA

A time-less generalization of the implementation concept is possible. This might be necessary for use with the "frozen formalism" that quantum gravity might have if the Wheeler-DeWitt equation is true. Transition rules would be replaced by implication rules: the laws of physics must imply that if the "input" substates have particular values, then the "output" substates have values that correspond to the rules. A chain of implications can then be constructed by taking those "outputs" as "inputs" for the next step (now a logical step rather than a time step), forming an extended computation. It must still be the case that there are many possible physical states consistent with the laws of physics, so that counterfactual implications would be true.

Another issue related to implementation is that the criteria for independence put few restrictions on what function can be used in a mapping from a single continuous physical variable to a continuous formal value; e.g. any 1-to-1 function is allowed. Such aspects of the system dynamics as finding the cube of a value are not well reflected by allowing any 1-to-1 mapping.

An example of an analog mechanism that could be used to find the cube of a number is a cone that can be filled with water up to a height corresponding to the desired number. (For this conceptual example, assume that water is a continuous fluid.) The water in

the cone can then be poured into a graduated cylinder. If the diameter of the cylinder is appropriate, the height of the water column in the cylinder will equal the desired cube. If the diameter were different, then scaling the height of the water column by an appropriate factor would give the desired cube.

A linear mapping for a continuous variable to a formal value, even if within a limited range, *would* be meaningful as a characterization of the system dynamics, as in the above example. This restriction can be imposed, although this is not necessary for computation in the classic sense, which is more concerned with combinatorial properties. An otherwise valid mapping for a multi-dimensional set of differential equations would still be nontrivial without the linear restriction.

A similar issue arises for mappings to digital formal values. For example, Joslin [7] believes that only a system that has something oscillatory about it implements a 1-bit oscillator, while a monotonically increasing clock would not. One option to produce that result would be to restrict mappings to time-symmetric monotonic functions of the variables.

Without that restriction a monotonically increasing clock would indeed implement a 1-bit oscillator. Note, however, that the formal value of that oscillator can only serve as input to a non-trivial computational time step (such as putting it into a NAND gate with another substate) if there is indeed something oscillatory about the dynamics of the system, given the mapping restrictions. Therefore there is no harm in not using such restrictions for complex systems. Also, in general a time-symmetric mapping may not be possible (e.g. if the hardware changes over time).

14 COUNTERFACTUAL STATES AND CAUSALITY

In addition to the problem of false implementations of virtually any computation by trivial systems, a related line of attack against computationalism argues that counterfactual transitions - which would have happened under different initial conditions - cannot affect consciousness [8,9,10]. Requiring that counterfactual transitions would have occurred is a crucial ingredient in rejecting false implementations because, for example, without that requirement any set of inert bits can be mapped to the output string of a proposed Boolean computation.

If this attack succeeds it therefore rules out computationalism. One exception has been proposed to that statement, which is that Platonically existing computations as an underlying reality could still give rise to consciousness, producing what we consider physics as an emergent property of typical conscious experiences [11]. However, pseudo-computations without the proper counterfactual behavior should then also exist Platonically, so even Platonic computationalism would be vulnerable.

Several arguments against the use of counterfactual transition requirements have been made, but they fall into a few basic categories. In the first category, the argument relies on incredulity that the detailed properties of a potentially very complicated or 'Rube Goldberg' subsystem can matter for consciousness if it is never even activated during the computation. For example, Maudlin [9] gives an example of a computer that operates straightforwardly for one input condition, which is in fact the actual one, but is required to call on different (and in the actual case, inert) machinery for any other input.

There is of course no empirical way to verify that any system other than one's own brain is conscious. In the face of this criticism based on one intuition, therefore, a computationalist responds by appealing to the contrary intuition that if-then relationships and feedback loops, the sorts of things captured by the notion of computation, seem to be things that would be important for consciousness, and must accept that these aspects of the overall structure and function of physical systems can indeed depend on complicated "inactive" components.

For weighing these contrary intuitions about "inactive" components, it is worth pointing out that so-called "inactive" physical components still have *function* in that they still evolve in time according to dynamical equations of physics; e.g. net force = mass · acceleration still applies even when the forces cancel to zero. This gives the components "if-then" functional properties and is very different from a situation in which the components *only* sit there and have no interesting functional properties.

In some cases, inactive components can be excluded from the mapping being considered, and what would have been their output is then treated as an input to the computation. This results in a different computation than the one that would have included all of the components, but it can be closely related to that one – perhaps close enough that for a particular initial condition, if one would be conscious, the other would have the same consciousness.

Consider a computer with a 'straightjacket' such that if it departs from a pre-specified sequence, the state will be changed by an external monitor to match the sequence. If it always matches the sequence, the monitor will make no changes (and will leave the subsystem of interest physically untouched). In the actual run, the monitor makes no changes. This system seems to have the wrong counterfactual relationships because of what the monitor would have done for counterfactual states, yet part of it is physically identical to a perfectly normal computer implementing that particular run without any external interference in actual fact, so it seemingly should implement that computation after all!

This is a case where the external monitor should be excluded from the mapping and its actions treated as a fixed sequence of inputs, where the sequence of inputs happens to be such as to leave the sequence of other substates the same as they would have been with no input. This computation is presumably conscious if the one without input would have been. Also, the mapping can be restricted to situations in which the input substates must have that fixed sequence, which in effect removes them as inputs.

A 'derail-able computation' in which the computation proceeds normally for some combinations of input substate values, but enters a 'halt' state and no longer undergoes nontrivial transitions for other possible input values, is another good candidate to be considered 'closely related' to a nontrivial computation which evaluates some function for all possible input values. The 'halt' value can be treated as one value of another substate, and in that case should be independent of the others. For example, a fuse might blow if an electrical computer is in a certain set of states.

Care must be taken, however, not to consider trivial computations to be 'closely related' to complex ones. For example, suppose there is a string of 'input' bits S recorded on one set of adjustable switches, and a string of 'output' bits R recorded on another set. The output bits initialize to a default string R_0 and remain that way if not adjusted. The default output string equals the base 2 value of some nontrivial function of the actual input string, S_0. An implementation mapping can claimed to exist such that the output R(t+1) will be the desired function of S(t) if $S(t)=S_0$ OR if Bob comes by, looks at the strings, and sets the output to be the desired function of S. In fact Bob will not come by, but since $S=S_0$, the computation is not only implemented but produces an output that does have the right value for the actual

value of the input. This computation may at first seem 'closely related' to one in which the proper function of the input is actually computed, but in fact it is a trivial computation since any bits anywhere could be mapped to the output string R_0, and cannot be 'closely related' to the nontrivial computation in the sense of having the same consciousness if any.

15 NEURAL REPLACEMENT … OR ELIMINATION

Another type of argument against using counterfactual transition requirements comes down to a neural replacement thought experiment, e.g. that of Bishop [8]. Similar thought experiments have famously been used to argue in favor of computationalist views of the mind [12], so using this thought experiment against computationalism is an interesting move, and refuting it is important for the viability of computationalism.

In a neural replacement argument (NRA) scenario, small components of a brain are replaced one at a time by alternate components that behave in the same way as the old ones. The following assumptions are made:

0) The functioning of the rest of the brain is preserved.

1) Any change this procedure might make to his consciousness is not something the person can explicitly take mental note of; he could not directly notice that anything has occurred.

2) Since he can't notice any change, it must be true that the properties of the person's consciousness – namely the things he experiences, including color qualia - do not change or fade away.

3) Sudden vanishing of consciousness when a certain number of components have been replaced wouldn't happen.

If these assumptions are granted, then a brain made of the new components must be equally as conscious as one made of the old ones. This is usually taken to imply that the behavior (which can be described in terms of computations) and not the composition of the components is what matters for consciousness.

The twist that is used to attack counterfactual transition requirements is to replace the old components - one at a time as before - with new components that have behavior that is only correct for the particular initial conditions that actually occur. The new components produce a fixed series of outputs and have no if-then sensitivity to counterfactual inputs. The argument is made as above that a brain made of the new components must be equally as conscious as a brain made of the old ones; but if so, that establishes that counterfactual sensitivity does not matter for consciousness. But for computationalism to work, counterfactual sensitivity must matter to filter out false implementations.

In order to counter this argument, a computationalist must reject one or more of the assumptions. While it is possible to reject assumption 0), the view of the brain as similar to a neural net classical computer (which is a view common among computationalists) implies that assumption; the argument can then be run in terms of an artificial conscious digital neural net brain.

To reject assumption 1) would imply that the mind can make mental notes that are not aspects of the functioning of the brain. It would be a dualist position, suggesting an immaterial mind. While dualism can be compatible with computationalism, as Chalmers has argued [12], such a divergence between the activity of the mind and that of the brain would not be.

Assumption 3), while not *prima facie* undeniable, is highly plausible because brains are highly variable. Rejecting it would be technically viable but would garner few if any supporters.

Assumption 2), that the properties of consciousness would not change, must be the one that computationalists reject, despite the fact that many computationalists have been ready to accept it in the context of the original NRA.

To shed light on the issue, consider another variation of the thought experiment. [13] In this case, when each small component is removed, it is not replaced by a substitute component. Instead, the exact same inputs that would have been fed to the remaining part of the brain by the missing components are supplied externally, as boundary conditions. For simplicity, assume that these inputs are correct due to extreme luck. If the details of an artificial brain's internal functioning are predictable, the inputs can be supplied by using those predictions.

Now only part of the brain remains, and that part becomes smaller as more components are removed - until vanishing altogether. The activity in the partial brain is the same as it would have been if no components were removed, since the boundary conditions are the same for that part of the system.

In this case it is not possible for the consciousness of the brain to remain unaffected by the removal of the components, because the tiny bit of a brain remaining near the end of the process would not be complex enough to give rise to cognition. The remaining consciousness *must* be only a partial version of the original.

Assumptions 0), 1), and 3) are equally valid in this case as in the standard NRA. Assumption 2) is clearly false in this case, because the subsystems of the brain responsible for the various types of experiences – such as color vision – will at some point simply no longer exist. But this means that the mind *is* not necessarily a good judge of what it is conscious of, since it can never at any point make a mental note of any changes.

To relate this partial brain argument (PBA) more closely to the NRA, consider replacing the removed components with mentally inert components – anything that can supply the right boundary conditions to the remaining normal brain, but which cannot support consciousness. For example, in a hypothetical model in which substance dualism were true, the mentally inert components would function the same in terms of input and output, but would lack the 'magic' substance needed for consciousness.

The remaining partial normal brain would be identical to the partial brain in the PBA, and would have the same partial consciousness. Assumption 2) is just as clearly wrong in this case as in the PBA. But this case is no different than that of the standard NRA, except that it has been stated from the beginning that the usual conclusion of the NRA that the consciousness is unchanged is false. As this argument shows, that is a perfectly self-consistent possibility, thanks to partial consciousness. As a result, any NRA fails to show what it was intended to show.

Apart from the PBA, there are other reasons to think that the mind is not a good judge of its own consciousness. For example, the central part of the human visual field is much more detailed than the peripheral parts, but humans rarely notice that fact. The idea that the mind would have to be able to notice any change in its own consciousness may be a sort of homunculus fallacy, in which the mind is thought of as an observer of its own thoughts.

With the NRA no longer viable, computationalism loses one of the arguments in its favor. However, the idea that the mind is a good judge of its own consciousness must also be abandoned, and it is a source of the anti-computationalist intuition that mental qualia are hard to relate to computation. The truth or falsity of computationalism is a complicated issue; what is being claimed here is merely that the need to rule out false implementations does not falsify computationalism.

16 AMOUNTS OF IMPLEMENTATIONS

If computationalism about consciousness is true, then in order to use it to relate a mathematical model of a physical system to predictions about what observers who live in that system would observe, it is not enough to know which computations are being implemented or even to also know how to relate each computation to a particular conscious experience or lack thereof. The reason is that multiple instances or different amounts of each computation may exist [14,15]. In the system's spectrum of computations, different computations must be assigned different weights.

That is certainly the case in the many-worlds interpretation (MWI) of quantum mechanics, for example. For the MWI, one must be able to put a measure on the computations in each branch of the wavefunction and relate it to the effective probabilities for observing different outcomes. In principle, if the proper way to do that were known, one could then either falsify the interpretation (while suggesting what modifications could fix the problem) or confirm that it does give the correct predictions.

The difficulty that arises is not only that the question cannot be investigated experimentally but also that there are few obvious restrictions that must be met, since the implementations being considered do all exist in the system. By contrast, in the case of defining implementation criteria, the need to avoid the possibility of false implementations at least provides a strict restriction that guides what criteria are acceptable.

That said, a few possibilities will be mentioned here. One possibility is that the measure is proportional to the number of independent implementations, where independence is established in the same way as for substates within a single implementation. A more lenient possibly is that in this context implementations are independent as long as it is physically possible to choose their initial conditions in any logically possible combination. These possibilities may be consistent with a slightly modified MWI [14].

Since the measures are also a characterization of the structures and function within the overall system, it is also possible that parts of the system that are in some sense physically larger should have more measure. That could be consistent with the fact that branches of the wavefunction with larger amplitude in the MWI have higher effective probabilities. However, that solution of the problem, while it might be qualitatively plausible, assumes what it is meant to explain on the quantitative level (amplitude-squared for effective probabilities), so it lacks philosophical force unless independent confirmation could somehow be supplied.

17 CONCLUSIONS

Implementation of Structured State Systems is a way to characterize the structure and dynamics of physical systems in terms of computations, such as is required for computationalist philosophy of mind. Criteria based on information available in the physical subsystems that are mapped to substates suffice to rule out false implementations, and the concepts of inheritance and transference extend the usefulness of those criteria to include various valid implementations that should not be ruled out. Options exist for better characterizing the systems for different applications or for systems with laws but not dynamics.

The implementation criteria include requirements on transitions in counterfactual states. This has been a subject of controversy in regard to the application to philosophy of mind. The importance it gives to inactive machinery is counterintuitive to some people, but currently inactive components still follow the laws of dynamics and this endows them with functional capabilities, which computationalists do not see as counterintuitive. Also, in some cases, 'improper' overall counterfactual behavior need not make any difference to consciousness in the computationalist view, as such systems can implement computations which can be closely related to the 'proper' one and which would give rise to the same type of consciousness.

The neural replacement argument, which traditionally has been used to argue in favor of computationalism, can be turned around to argue against it with the use of components with pre-specified activity. To counter this, the partial-brain argument was given as a general counterargument to the neural replacement argument. With only part of the brain in existence, it becomes clear that consciousness in such cases must become more partial as the process goes further, and the same partial consciousness can apply in the neural replacement scenario. This neutralizes the neural replacement argument in both its traditional pro-computationalist form and in its anti-computationalist form.

Given that a system does implement various computations, some of which (assuming here the validity of computationalism) give rise to conscious observations, it is necessary to have a way to assign an effective probability to each observation in order to predict what observers who live in that system should expect. The way to do this remains an open question, which assumes particular importance for the evaluation of interpretations of quantum mechanics, because the quantum wavefunction has a many-worlds character and the origin of effective probabilities in that context has not been adequately explained.

REFERENCES

[1] D. Chalmers. Does a Rock Implement Every Finite-State Automaton? *Synthese*, 108:309-33 (1996).

[2] H. Putnam. *Representation and Reality*. MIT Press (1988).

[3] Searle, J.R. Is the brain a digital computer? *Proceedings and Addresses of the American Philosophical Association*, 64:21-37 (1990).

[4] M. Tegmark. The Mathematical Universe. *Foundations of Physics*, 38:101-150 (2008).

[5] D. Chalmers. The Varieties of Computation: A Reply. *Journal of Cognitive Science*, 13:211-248 (2012).

[6] M. Sprevak. Three challenges to Chalmers on computational implementation. *Journal of Cognitive Science*, 13:107–143 (2012).

[7] D. Joslin. Real realization: Dennett's real patterns versus Putnam's ubiquitous automata. *Minds and Machines*, 16:29–41 (2006).

[8] M. Bishop. Counterfactuals cannot count: a rejoinder to David Chalmers. *Consciousness & Cognition*, 11:4:642-652 (2002).

[9] T. Maudlin. Computation and consciousness. *The Journal of Philosophy*, 86:407–432 (1989).

[10] M. Muhlestein. Counterfactuals, Computation, and Consciousness. *Cognitive Computation*, 5:1:99-105 (2013).

[11] B. Marchal. The computationalist reformulation of the mind-body problem. *Prog Biophys Mol Biol*. 113(1):127-40 (2013).

[12] D. Chalmers. Absent Qualia, Fading Qualia, Dancing Qualia. In: *Conscious Experience*. T. Metzinger (Ed). Imprint Academic (1995).

[13] J. Mallah. *The partial brain thought experiment: partial consciousness and its implications*. Unpublished manuscript. **http://cogprints.org/6321/** (2009)

[14] A. Pruss. *Functionalism and Counting Minds*. Unpublished manuscript. **https://bearspace.baylor.edu/Alexander_Pruss/www/papers/CountingMinds.html** (2004).

[15] J. Mallah. *The Many Computations Interpretation (MCI) of Quantum Mechanics*. Unpublished manuscript. arXiv:0709.0544v1 [quant-ph] (2007)