Models of Neural Computation-

An Examination of David Chalmers' Causal Theory of the Mind

By
Dinyar Mistry, B.A.

A THESIS SUBMITTED TO MACQUARIE UNIVERSITY

FOR THE DEGREE OF MASTER OF RESEARCH

DEPARTMENT OF PHILOSOPHY,

FACULTY OF ARTS

MACQUARIE UNIVERSITY, NSW 2109, AUSTRALIA

OCTOBER 2015

Declaration

I certify that the work in this thesis has not been submitted for a degree nor has it been submitted as part of the requirement for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the presentation of the thesis itself has been acknowledged.

I certify that all information sources and literature used are indicated in the thesis.

Dinyar Mistry October 28, 2015

Acknowledgements

I would like to thank my supervisor Dr. Colin Klein for his help, support and encouragement while I was writing this thesis.

TABLE OF CONTENTS

Summary

David Chalmers has defended a causal version[1] of the Computational Theory of the Mind(CTM) by formulating an abstract computational object called a Combinatorial State Automata(CSA) which he argues can cover the structure of different abstract computational objects such as Finite State Automata, Turing Machines, Cellular Automata *etc*. He views implementation as the bridge between formal computation and physical computation. He defines implementation as an isomorphism between causal processes of a physical object and the formal structure of a computation. He uses his causal version of CTM to connect computation and cognition by defending a thesis of *computational sufficiency* and uses computation as an explanatory framework for cognitive processes and behaviour. In my thesis I examine Chalmers' views to see whether his argument stands up to scrutiny and whether his views are supported by the data of neuroscience. Where there are shortcomings I modify and extend Chalmers' theory to make it compliant.

[1] Chalmers D, (2011), A Computational Foundation for the Study of Cognition, ***Journal of Cognitive Science 12: 323-357***

1. Introduction

In contemporary philosophy, materialist theories of the mind have dominated discussion in the philosophy of mind, with versions of the Computational theory of the Mind (CTM) being in the mainstream. The CTM is based on the intuition that the mind is a computer, and mental processes involve computations.

The idea of the mind as a computer program was originally proposed by Putnam (1960). It was based on the abstract idea of computation derived from the work of Turing (1936) and Church (1936) and others who formalised the notion of algorithm or effective procedure in mathematics. An algorithm is an explicit step by step procedure consisting of a set of instructions. When these instructions are carried out sequentially over input values it transforms them in a deterministic way into one or more subsequent states plus intermediate values and finally into output values. This makes it possible to define solutions to problems of a class which are effectively decidable *i.e*. can be mechanically done by rote by pencil and paper alone. Such a procedure or algorithm constitutes the machine table or program of a computer.

On this view, the mind is viewed as the machine table or program of a computer. Another way to look upon the machine table is as a set of rules operating on one set of symbols to transform them into another set. For example the Rule "If *a* then *b*" implies that if input is symbol "*a*" then output is symbol "*b*". Note that this rule assigns no meaning to the symbols "*a*" and "*b*". Putnam's view of CTM is cast in this syntactic vein of not assigning any meanings to machine tables. As the machine table

or program is not a physical object the relationship between the mind and the machine table is not between the mind and a physical object but to an abstract object. This was an attraction at that time as the preceding theory of the mind, the Identity Theory of Smart (1959) and Place (1956) ran afoul due to its assertion of identifying mental states and processes with actual brain states and processes thereby unable to account for the multiple realisation of mental states such as "pains" in species with totally different physiologies. CTM was deemed more plausible and replaced the Identity Theory. The versions of CTM developed and proposed by Fodor (1975) following Putnam became for a while according to Fodor (1975, pp. 27-53) "*the only game in town*".

Fodor's (1975) main contribution was to modify CTM by marrying the Putnam view of "the mind as a computer" to the view that the way in which these computations are done are by processing symbolic representations about the world so as to give truth-value, reference and meaning to these representations in a manner similar to a natural language statement. In fact Fodor called his theory the Language of Thought Hypothesis which asserted that thought takes place within a mental language. However working out the nature of the meanings or the semantic relationship for mental representations turned out to be controversial and in some respects a vexed topic. This led to a number of moves by Fodor and his collaborators to save the day resulting in more than one version of his theory. His thesis is also known as the "semantic version" of CTM. I discuss Fodor's views further in Section 2.6 of the next chapter.

1. Introduction

The main weakness of Fodor's theory (Milkowski 2015) was that it could not offer a clear connection between computation at the abstract formal level and computation at the physical level. That is to say, it could not explain the implementation relationship between abstract computation as logically done by a Turing Machine and concrete computation as performed by a physical computer.

David Chalmers (2011, pp. 325-326) on the other hand identified the need for giving an account of the implementation relationship between computation at the abstract and physical levels as crucial to giving an account of the mind and proposed his causal account of computation. He argued that by explaining this link he was able to make computation foundational to mentality and cognition. He saw his causal version of the computational theory of the mind as providing a foundation to Cognitive Science and Artificial Intelligence and avoiding objections made to earlier versions of CTM.

Chalmers causal theory is based on the intuition that a system implements a computation when the causal structure of the physical system mirrors the formal structure of the computation. His aim is to develop an account based on a rigorous concept of the implementation relationship between abstract or formal computation and computation as realised in physical systems. While computation at a formal level was well understood from the work of Turing and Church however there was no clear account for computation as implemented in physical systems. There is clearly a need for this if computation is used to explain cognition and minds as they are realised in physical systems, namely brains.

Chalmers has developed this intuition by formulating an abstract computational object called Combinatorial State Automata (CSA) which he argues can cover the structure of all different types of formal computational objects such as Finite State Automata (FSA), Turing Machines, Cellular Automata *etc*. FSAs and CSAs are discussed in detail in Section 2.2 and 2.3 respectively of the next chapter. He then brings out the connection between computation and cognition by arguing that the right kind of computational structure suffices for the possession of a mind. Furthermore he argues that computation is an explanatory framework for cognitive processes and behaviour. I discuss in detail the link between computation and cognition in Section 2.5 of the next chapter.

1.1 Methodology

The methodology used in the thesis is to look at both the logical soundness of Chalmers' argument and to see if it is supported by the data of neuroscience. Where there are any weaknesses in the argument I endeavour to see if the argument can be modified, extended or has to be rejected. Whereas the critical analysis strand of the methodology is a standard approach in a philosophy thesis, I have also brought to bear on Chalmers' thesis the weight of empirical data from neuroscience. I do this by looking at whether extant neural processing can be looked upon as a computation and if so whether it is a type of computation that can be supported by Chalmers' thesis. I have taken this approach because Chalmers' says that the main purpose of his work is to give an account of the mind and cognition which is foundational to Cognitive Science and Artificial Intelligence. Therefore for Chalmers' enterprise to be relevant requires examining whether his theory is compliant with neuroscience as otherwise it would have to be rejected.

The outline of the rest of the thesis is as follows. In the next chapter I examine Chalmers' theory in detail. Chapter 3 is devoted to a discussion on neuroscience and whether neural processes underlying cognitive processes are either a digital computation or an analog computation, or a hybrid computation or not a computation. Chapter 4 discusses whether Chalmers causal version of CTM is compliant or can be made compliant by modification or extension with the data about neural computation discussed in Chapter 3. Finally in Chapter 4 I also consider the case if neural processes were not a computation. I close with concluding remarks in Chapter 5.

2. Chalmers on Implementation and Cognition

In this chapter I take a close look at Chalmers' thesis. I discuss his analysis of the relationship between abstract and physical computation and how he links the latter to the mind and cognition. I examine the merits of his theory and how it responds to the triviality arguments of Putnam (1988) and Searle (1992). Finally I cover the main objections raised by his critics with his replies.

The outline of the chapter is as follows. In Section 2.1 below I discuss Chalmers' causal thesis of the mind. There I look in detail at the implementation relationship between abstract and physical computation which is central to his account of the mind. Chalmers develops his theory based on an abstract computational object namely Combinatorial State Automata (CSA) which is an extension of Finite State Automata (FSA). FSA's are crucial for the development of Chalmers' views hence in Section 2.2 I first discuss the nature of FSA accompanied by a detailed example of their functioning. I outline Chalmers discussion on implementation via FSA and their shortcomings. Section 2.3 is devoted to a discussion of the definition of the CSA and the implementation relationship using CSA instead of FSA. In Section 2.4 I discuss how CSA's can also apply to connectionist networks thereby making Chalmers' theory applicable to both symbolic and sub-symbolic architectures. Having established the implementation relationship using CSA I turn in Section 2.5 to a discussion of the link between computation and cognition as envisaged by Chalmers with a discussion of some issues identified with his thesis of computational explanation. In Section 2.6 I discuss the merits of Chalmers' theory with a discussion in Section 2.7 of how it overcomes the trivialisation objection raised by Putnam and Searle. Finally in Section 2.8 I discuss objections raised to Chalmers' views with his

replies. I close the chapter in Section 2.9 with concluding remarks and prefacing the strategy for the rest of the thesis.

2.1 Chalmers' Causal Thesis of the Mind

As a background to the paper Chalmers (2012, p. 212) says that he does not see his work posing a radically original view but squarely in the roots of Putnam's (1960) original work and trying to overcome the triviality objections of Searle (1980) and the later Putnam (1988). The main subject of Chalmers causal thesis is a discussion of two questions. The first question is the nature of the implementation relationship between abstract computation (based on the mathematical theory of computation) and physical computation in physical systems, for example as in computers. The second question is about the nature of the link between physical computation and mentality in general and cognition in particular.

I will focus on the first question before the second as Chalmers (2011) views an account of the nature of the implementation relation between a physical system and an abstract computation as the key to understanding minds and computational processes in cognition. This is because while the mathematical theory of computation is well understood however it is not clear "*what is it for a physical system to implement a computation*" (Chalmers 2011, p. 323). The idea was to give a clear account of the implementation relationship between an abstract computation and a physical system. Furthermore he posits that without a clear answer to this question the foundational role of computation in cognitive science cannot be justified as cognition does not occur in the abstract but as a physical process in the brain.

2. Chalmers on Implementation and Cognition

As mentioned earlier in Chapter 1, Chalmers' main intuition is that for the implementation of an abstract computation such as a program P, the formal structure of P must be mirrored in the causal structure of the physical system.

In a detailed gloss he says *(*Chalmers 2011, p. 326*)*:

"A physical system implements a given computation when there exists a grouping of physical states of the system into state-types and a one-to-one mapping from formal states of the computation to physical state-types, such that formal states related by an abstract state-transition relation are mapped onto physical state-types related by corresponding causal state-transition relation."

I explain each of the points made in detail below.

1) A physical system has a structure *viz*., electrical for a computer, electrochemical for a nervous system and electromechanical for a manufacturing control system to give three different examples. Any such physical system will be in different physical states at the same time. For example having a current flowing or a voltage level for an electrical system; a chemical reaction occurring along with electrical activity of a current or voltage for an electrochemical system and a force or pressure exerted along with electrical activity for an electromechanical system. These different physical states occurring concurrently can be grouped by the requirements of the computation under study. For example one grouping can be by the nature of the state. All the electrical ones can be grouped together, as can all the chemical or all the mechanical ones. Similarly groupings can be done by some other features such as spatio-temporal factors such as all the states at particular location(s) in space or time(s). Another example of a grouping parameter could be all the states to produce a particular type of output for a particular type of input.

2) Sub-sets of these grouped states form a state-type such that there is a one-one mapping between each state-type of the physical system and an abstract computational state of the machine table or algorithm being implemented. The notion of machine table or algorithm was introduced in the previous chapter in Section 1.0. A one-one mapping is an isomorphic relationship between two sets or groups such that one member in one set or group corresponds to or picks out one and only one member in the other set or group. Such a mapping can be just a correlation or based on a rule. The rule can be just a "*If a then b*" type of material conditional or one that is law like obeying counterfactual conditionals such as "*If a were to happen then b would happen*" so that it has modal force. In the case of the material conditional the truth of the consequent "*b"* clause is tied up with the truth of the precedent *"a"* clause. While in the case of the counterfactual conditional the if clause is not true, while the then clause may or may not be true but certainly would be true in the counterfactual circumstance of the if clause being true. Given a set of statements about possible states of affairs the material conditional is weaker as its truth only applies to those that are realised in reality while the counterfactual conditional applies to the full set of statements irrespective of their instantiation in reality. Hence the counterfactual conditional is the one to express law like behaviour. Obviously the strongest type of isomorphism will be the one based on the rule supporting counterfactuals. Chalmers above description makes no mention of the nature of the isomorphism. But as we will see later in the more refined versions below he argues for law like rules with counterfactual force in order to overcome Putnam's and Searle's well known triviality objections. This is discussed in Section 2.7 below.

3) The states of a complex physical system change over time either due to internal changes in its structure or function or its states might change due to external environmental inputs. Chalmers' analysis is via an isomorphism or one to one mapping between groups of physical states (*i.e.* state-types) that have a causal role (or causal topology as he terms it) and formal computational states and becomes clearer in my description of Finite State Machines below.

Chalmers' above description of implementation is made all encompassing by specifying it by a schema that covers all other computational formalisms such as Turing machines, Finite State Machine or cellular automata. Chalmers introduces the notion of an abstract computational object called a Combinatorial State Automata (CSA) which has the power to mirror the states of a Turing machine (or any other computational formalism). It is implemented when the state transitions of a physical system map on to the state transitions of the CSA. This is an improvement on earlier accounts by Putnam (1967) using Finite State Automata (FSA).

2.2 Finite State Machines and its Relation to Chalmers' Thesis

I will explore the concept of FSA in a little detail as it is so central to Chalmers' account of a CSA which is a development of FSA. A Finite State Machine (Automata) (Denning, Dennis, Qualitz 1978 pp. 88-136) is a logical computational object which is made up of a set of inputs, a set of states, a set of transitions from one state to another and a set of outputs. The machine is in only one state at a time and there can be only a finite number of states. From its current state at some point in time the machine transitions to a new state plus a new output based on a triggering event which results in a new input. That is for each combination of an input and an internal state there is

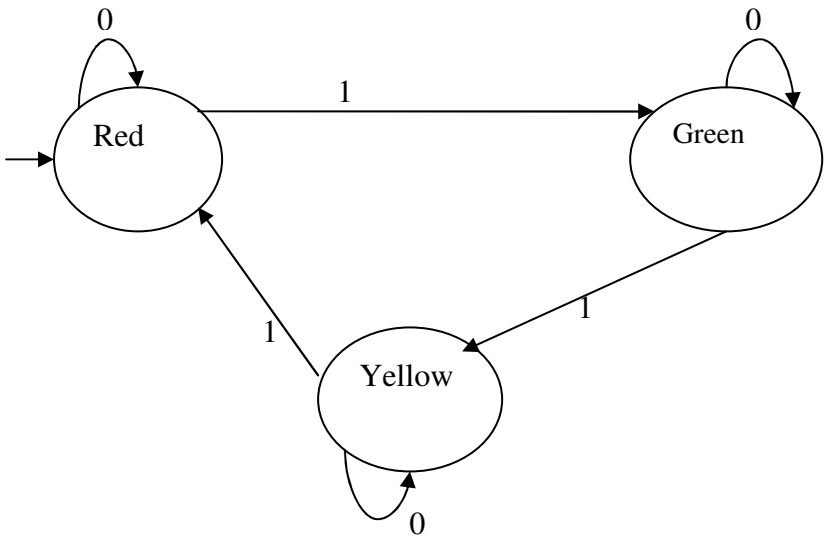
a corresponding transition function which results in a new internal state and a new output. A standard version of the machine has a starting input and state and a finishing state and output at which the machine stops. Note that variants of the FSA model involve machines which have no output and machines which do not halt but keep cycling sequentially through all their states.

A physical version of a FSA can be looked upon as a machine with input signals, internal states and output signals. Examples of FSA's used to control a device are a vending machine or traffic lights. By machine I mean that it operates in a mechanical step by step procedure or algorithm to arrive at the result. FSA's have wide uses such as parsers and lexical analysers in computing.

The following example[2] illustrates the FSA concept. Consider a set of traffic lights which cycles through the three states Red, Green and Yellow. The change in state is controlled by a 5 second delay timer. We can illustrate the operation of this FSA by the transition diagram below. The nodes represent the states (Red, Green and Yellow). The arrows from state to state represent the timer value which causes the transition and changes a state (*e.g.* from Red to Green) and the label on the arrow represents the input received (in binary bits) to change the state when time = 5 seconds. As there are only two relevant controlling values namely 5 seconds or < 5 seconds we can replace these by the binary values 0 and 1 so that any timer value < 5 seconds is shown as an input of 0 while the timer value = 5 seconds is shown as an input of 1. Note that this FSA does not halt as there is no final state and the machine keeps cycling from one

[2] This example is adapted from discussion and examples on FSA in Denning, Dennis, Qualitz (1978) and in Hopcroft , Ullman (1979)

state to the next state every 5 seconds. There are other FSAs which can have a final state and halt but the basic concept is the same.



The input arrow going to the Red state indicates the start state. The FSA stays in Red until the timer is 5 seconds when it transitions to Green and the timer initialises and starts counting again. The arrow labelled "0" curling back to the same state indicates that the timer is < 5 seconds and no transition has occurred. While the arrow from one state to the other labelled "1" indicates that the timer = 5 seconds and a state transition has occurred.

This simple FSA can be implemented by Flip Flop circuits which alternate between 0 and 1 (*i.e*. ON or OFF) for each light based on the control signal from a timing circuit. Flip Flops are circuits that can be either in one state or another based on a controlling input and hence are widely used to implement binary states of 0 or 1.

Another way of representing the above FSA is by a transition table which shows the states the inputs, the transitions as shown below.

.

STATE	**INPUT (Timer Signal)**	**NEXT STATE**
Red	0(Timer<5secs)	Red
Red	1(Timer=5secs)	Green
Green	0(Timer<5secs,)	Green
Green	1(Timer=5secs)	Yellow
Yellow	0(Timer<5secs,)	Yellow
Yellow	1(Timer=5secs)	Red

The causal structure to note in this implementation is made up of the two components namely the timer circuit and the Flip Flops. The timer circuit clocks the 5 second delay between states and controls the firing of the Flip Flops which sends binary signals to the lights based on which the lights transition from one state to another.

A physical traffic lights system has a causal structure which will be made up of a number of different physical states described as follows. Electrical, connected with the current flowing through the circuits of the system. Thermal due to the heat generated in any circuit accompanying a flow of electrons. Chemical related to the release of chemicals (*e.g*. gases) in the air detected as a smell or maybe odourless either from paint or sometimes from hot electrical components. Finally, gross material based on weight, shape, size and molecular structure.

It is important to see that the computation of this FSA is only related to the functioning of the causal structure of the electrical system. An actual traffic light will have a causal nexus inter-relating all the above different types of physical states. In an intuitive way it is easy to see for this example that the causal structure of the physical electrical circuit mirrors the formal structure of the FSA. On the formal side we have the transition table or diagram of the FSA which gives the inputs, the states and the transitions from state to state. On the physical side the causal structure of the flip flops is controlled by the timer circuit which gives rise to a state transition every time the timer = 5 seconds. Hence each transition in the transition table is mirrored by corresponding causal processes in the flip flops when the timer = 5 seconds which leads to the lights changing state. Thus the formal structure of the computation as enshrined in the formalism of the FSA is mirrored in the causal structure of the physical system.

In fact depending on the design of the circuit we ignore all groups of electrical (physical) states where the timer is < 5 seconds and are only interested in the electrical (physical) state where the timer equals 5 seconds. So we have a one-one mapping or isomorphism between physical state types and abstract states as posited by Chalmers in Section 2.1 above of what it means for a physical system to implement a computation. In this sense the causal organisation is invariant and the abstract computation of the FSA is realised in the implementation or physical computation via the output bits at each state.

Chalmers defines an FSA as follows:

"An FSA is specified by giving a set of input states I_1, ..., I_k, a set of internal

states $S_1,...,S_m$, and a set of output states $O_1,...,O_n$, along with a set of state-transition relations of the form (S, I) → (S', O'), for each pair (S, I) of internal states and input states, where S' and O' are an internal state and an output state respectively. S and I can be thought of as the "old" internal state and the input at a given time; S' is the "new" internal state, and O' is the output produced at that time." Chalmers (2011, p.326)

Chalmers definition captures the gist of the points about FSA I had explained in detail above both in the description and in the traffic light example given.

Chalmers defines the implementation of a FSA as follows:

"*A physical system P implements an FSA M if there is a mapping f that maps internal states of P to internal states of M, inputs to P to input states of M, and outputs of P to output states of M, such that: for every state transition relation (S, I) → (S', O') of M, the following conditional holds: if P is in internal state s and receiving input i where f(s)=S and f(i)=I, this reliably causes it to enter internal state s' and produce output o' such that f(s')=S' and f(o')=O'.* Chalmers (2011, p. 327)

Chalmers definition needs some expansion and explanation as follows.

In the example above the finite set of states are the traffic light states Red, Green, and Yellow. The finite input is made up of the binary values 0 and 1. The transition function is realised by the combination of the current (traffic light) state and Input = 1 *(i.e.* timer = 5 seconds) which results in the state transition to the next state. The initial or start state = Red. Finally there is no final state for this FSA as it is a cyclic process.

The thing to note is that the state transition relation in the physical system is based on a causal transition from one state to the succeeding state and furthermore such

causation must be according to Chalmers a "reliable" one. Although Chalmers does not formally develop here this notion of reliable causation I take it he means that the causation must be law like and counterfactual supporting. Earlier when I had discussed Chalmers intuition in Section 2.1 I had stated how an isomorphism can range from a mere correlation to one based on a rule that is a law like counterfactual supporting one. There Chalmers was silent about the nature of the isomorphism. Here by referring to reliable causation he has indicated that the state transitions must be law like and counterfactual supporting. By imposing the additional condition of supporting counterfactuals Chalmers' separates out correlations and coincidences from law like behaviour. That is the state transitions should not just be material conditionals of the sort "If at time t, system happens to be in state *a* then it would transition to state *b*". But rather they must be counterfactually true so that they are of the form: "*Given a formal state-transition* $A \rightarrow B$*, it must be the case that if the system were to be in state A, it would transit to state B" (*Chalmers 2011, p. 333*).*

For computation, the important cases are ones where the input was different but the machine table continues to hold true. For example if their were three state transition rules $A \rightarrow B$, $B \rightarrow C$, $B \rightarrow D$ in a machine table that branches based on a decision so that only one of $B \rightarrow C$ or $B \rightarrow D$ gets instantiated in a run, say $B \rightarrow C$, then the counterfactual conditional ensures the truth of the state transition rule $B \rightarrow D$ even though it does not get instantiated in a particular run. Similar remarks apply for state transition rule $B \rightarrow C$ in the runs when $B \rightarrow D$ gets instantiated.

In his discussion of the Putnam objection to CTM Chalmers' uses the above counterfactual conditional for a FSA. It is the truth of the counterfactual that makes the state transitions neither a mere correlation nor an arbitrary causal effect *e.g*. due to

say stray fields or cosmic rays changing the state of a circuit and tripping it thereby causing a change of state.

One of the things to note about a FSA is that the elements that make up its states, inputs or outputs are monadic scalar quantities like numbers or symbols from an alphabet *i.e.* they have no structure such as vectors or matrices do. In our above example the states were represented by the values "Red", "Green", and "Yellow" which have a value but no structure. Similarly consider another example of a FSA such as a Vending Machine where the machine is either in a state for receiving a customer's selection (the Order state) or is in the state of delivering the selection (the Delivery state). So that while there may be multiple inputs and outputs to such a machine (depending on the items stocked and selection made) however the states have no internal combinatorial structure *i.e.* they cannot be broken up into sub-states.

FSA's are simple low level computation devices and have limitations on the number of computational problems they can solve. They have limited memory as it is limited to the state and they have no control structure. Turing Machines on the other hand are a class of computational object which have more computational power. They can solve all effectively decidable problems[3] *i.e.* they can in theory generate an algorithm for a problem if one is possible. They are the model of the general purpose computer. Conceptually a Turing machine can be looked upon as a FSA with memory. It consists of an infinite tape, a Read/Write Head and a finite set of states which form its machine table. The tape is made up of cells consisting of a datum *i.e.* having a value expressed as a symbol from an alphabet set. The machine reads the tape one cell at a

[3] Note: This assumes the truth of the Church-Turing thesis.

time and then depending on its instruction set writes a value back to that cell and then moves the head one cell either to the left or the right of the tape.

2.3 Combinatorial State Automata

Chalmers (2011, p. 328) extends the concept of a FSA to one where the internal state is extended by having a combinatorial structure. By that I mean that the elements of a CSA are vectors. The vector can be made up of sub-states mirroring *e.g.* the states of a Turing machine such as a combination of Head states, Tape states and internal states. CSAs can be represented by vectors as sub-states of an overall state. Hence CSA's are a more faithful translation of Turing Machine states or the cell pattern of a cellular automata or whichever formal computation is the subject of an implementation. The other point where CSAs differ from FSAs is that the internal states can be either finite or infinite while for a FSA they are always finite. But both inputs and outputs in the case of CSAs are finite like FSAs. Chalmers mentions that for all practical purposes the finite case suffices. He needs the infinite states to ensure that it makes CSAs powerful enough to cover all computational formalisations.

As Chalmers states it a physical system P implements a CSA, *M*, when the following conditions are met:

"*If there is a vectorization of internal states of P into components* $[s^1, s^2, \ldots]$,

and a mapping f from the sub-states s^j *into corresponding sub-states* S^j *of M, along with similar vectorizations and mappings for inputs and outputs, such that*

for every state-transition rule $([I^1, \ldots, I^k], [S^1, S^2, \ldots]) \rightarrow ([S_o{}^1, S_o{}^2, \ldots], [O^1, \ldots, O^l])$ *of M: if P is in internal state* $[s^1, s^2, \ldots]$ *and receiving input* $[i^1, \ldots, i^n]$ *which map to formal state and input* $[S^1S^2, \ldots]$ and $[I^1, \ldots, I^k]$ *respectively, this reliably causes it to*

enter an internal state and produce an output that map to $[S_o^1, S_o^2,\ldots]$ *and* $[O^1, \ldots, O^l]$ *respectively*." (Chalmers 2011, p. 329)

The above can be unpacked as follows. Each internal state of P is a vector given by its components $[s^1, s^2,\ldots]$, and similarly each internal state of the CSA is a vector whose components are $[S^1, S^2,\ldots]$. A vector consists of an ordered set of sub-states or components for each internal state of both the physical system and the CSA. For example each component of the vector corresponding to an internal state of P could represent a spatial location coordinate. The inputs and outputs of the physical system have combinatorial or vectorial structure *e.g.* $[i^1, \ldots, i^n]$ and $[o^1, \ldots, o^l]$ respectively. The inputs and outputs of the CSA also have combinatorial or vectorial structure *e.g.* $[I^1, \ldots, I^k]$ and $[O^1, \ldots, O^l]$ respectively.

There is a function f that does a one to one mapping from each sub-state s_j of the internal state, of the physical system P to each sub-state S_j of the internal state of the CSA, M. along with mappings of the components of the inputs and outputs of P to components of inputs and outputs of M. This mapping is based on the state transition rule of M $([I^1, \ldots, I^k], [S^1, S^2,\ldots]) \rightarrow ([S_o^1, S_o^2, \ldots], [O^1, \ldots, O^l])$. The state transition rule specifies a unique mapping for a combination of input state vectors and internal states of M giving new internal state vectors plus an output vector of M. Furthermore the state transition rule must be such that the causal transition in the physical system P is reliable *i.e.* counterfactual supporting as discussed earlier in connection with reliable causation for FSA implementation.

By formalising the CSA Chalmers gives a formal account of his intuition about what it is for a physical system to implement a computation and how to interpret the mirroring relationship between the states of a physical system and the states of an abstract computation.

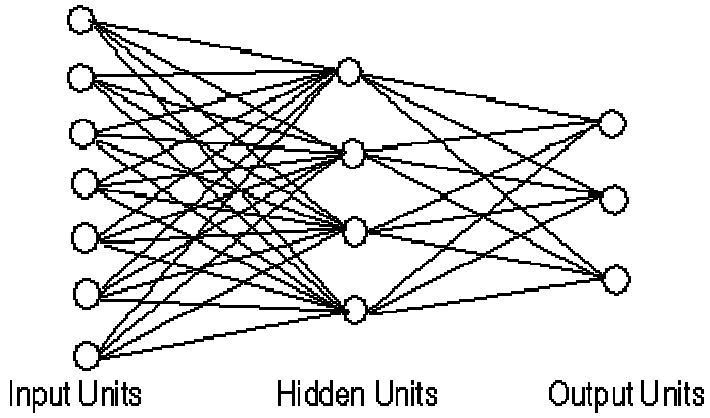
2.4 CSAs and Connectionist Networks

In this section we look at another class of computational structures called neural networks. They operate in a different way then the state based machines we have seen so far in FSAs, TM and CSAs. Chalmers CSA formalism can be extended to neural networks as neural networks can be described by a Turing Machine which in turn can be described by CSAs.

The entities of computation in the transition (machine) table that we have seen so far *e.g*. in the FSA traffic light example operate at the "symbol" level. By this I mean that the states of the FSA (Red, Green and Yellow) and the inputs (0 or 1), correspond to the status of the lights and the timer value of < 5 seconds or timer value = 5 seconds respectively. There are another class of systems whose entities of computation operate at a lower "sub-symbolic" level. To paraphrase Chalmers (2011, p. 351-352) in the state machine case the units that do the computation and the vehicles of representation are identical as both operate at the symbol level. In the neural network case the units of computation and the vehicles of representation are different. The claim is that these sub-symbolic structures and their operation is closer to mental processes in brains.

Our standard architecture of all computers since their inception is based on what is called the Von Neumann (1945) model. It is centred on serial processing and a separation between data and programs. Unlike the Von Neumann (1945) model of computers neural networks have parallel processing and have distributed data and instruction sets. Furthermore they do not have a separate declaratively held program / knowledge store. They are made up of simple computational units having no

representational link to objects in the real world. The computational units act as nodes with links between them to form a computational network. The representation to the real world is derived in a distributed way based on the pattern of activation of nodes and the links and weights to connecting nodes in the rest of the network. Its proponents (connectionists) argue that these are more suited to model and explain the mind as they are closer to the way the brain works via neurons and connections between them via synapses to form networks. However note that their computational ability is prime and not their representational ability. There can be such networks performing computations that are purely syntactic without having any meaning just as there can be Turing Machines or CSAs performing meaningless computations on strings of numbers or symbols of an alphabet. No doubt for cognition a semantic output is required but it is not required purely for computation. As mentioned above Chalmers' CSA model applies to computation at both a symbolic and a sub-symbolic level. The reason for this is that neural networks can be rendered in a Turing Machine form and therefore as a CSA. The following diagram from Garson (2015) represents a simple neural net showing the layer of input processing units, the pattern of connections between all units, the hidden layer units of processing, and the output units. It is the pattern of activation in the net that converts inputs to outputs.



Each input unit has an activation value that it sends to the hidden units which in turn send their activation value to the output units. Each layer of units gets activated based on the strength of the activation values received from the previous layer of units. The pattern of activation of the network depends on the strength of the connection or the weights between the units in the different layers. An activation functions calculates the activation value at a hidden or output unit to arrive at the resulting value for that unit taking the strength of all the inputs received into account. This simple net is called a feed forward net as the outputs are just passed on. It will produce the same result every time for the same set of inputs. More complex networks are also possible which will have connections from forward layers to back layers causing modification of results. While these more sophisticated neural nets are used to simulate cognitive functions it does not concern my thesis as my main interest is not a study of connectionist networks but rather to show that neural nets also fall under Chalmers' computation model.

I will now show how a connectionist or neural net can be rendered in terms of a CSA. According to Rummelhart (1998) the conceptual framework of connectionist architecture takes the "abstract neuron" as the fundamental processing unit. Computation occurs by interactions between these units. These processing units may represent either particular features[4] of a problem over which meaningful patterns can be defined or they can be a totally abstract set of nodes having no meaning. The units are connected to each other. It is this pattern of connectivity among units in a connectionist system which determines what the system knows and how it behaves. As mentioned earlier the units are generally structured in 3 layers namely input,

[4] such as for example a network to train phonemes of words to get their plurals

hidden and output[5]. There is a state of activation defined over the units as denoted by a vector which represents the state of the system at a point in time *t*. An output function maps each unit's state of activation into an output signal. Units transmit signals to units in their neighbouring layer and the degrees to which they affect their neighbours depend on their state of activation. A rule of activation determines how the inputs into a unit combine with the present state of a unit to give the new state of a unit. A learning rule specifies how the system changes with experience.

A connectionist net N can be rendered as a CSA M by initially ignoring the training rule and treating only the starting form of the net thereby simplifying the problem. This is because from a CSA viewpoint all the training rule is doing is changing the neural net and therefore the CSA either in terms of inputs or states or outputs or the state transition rules. Initially we want to get the form of the CSA right for the connectionist network. Once that is done the changes to generate a new CSA based on the training rule should be fairly straightforward.

In order to express a network N as a CSA we first set up a correspondence between an internal state vector of the CSA and an appropriate part of the network as follows. Each internal state s of the network N at a point in time t is based on its state of activation defined over the units and their pattern of connections and can be denoted by a vector whose components are $[s_1, s_2,\ldots s_n]$ and are made up of the states of the processing units and the weights of the pattern of connections between units. There is a function f that does a one to one mapping from each sub-state s_j of the network to each sub-state S_j of the CSA.

[5] with the input units and their connections storing the content and the hidden units plus connections the pattern

There are input, output and hidden layer vectors corresponding to the underlying layers of units and their relationships in a neural net. We assume as many hidden layers as required for a problem although generally there is only one. The inputs and outputs also have combinatorial or vectorial structure whose components corresponds to the states of the units plus the relationships between units reified as entities whose values are based on the connection values. State-transition rules are determined by the reliable causation for each element of the state-vector, a function by which its new state depends on the old overall state-vector and the input-vector, and the same for each element of the output-vector. That is there is a one to one mapping which gives an isomorphism between the states of the units and their connections and the vectors corresponding to the CSA based on the causal structure of the network. Therefore we have a vectorisation of input, internal (hidden) states and output states, and a mapping function which maps sub-states of inputs to sub-states of outputs based on a transition rule thereby enabling a CSA to represent a connectionist network.

The purpose of this section was to show that the CSA format can be extended to other forms of computation such as neural networks. This would apply to neural networks both at a purely syntactic level where the outcomes of the neural network have no meanings or to a neural network whose outputs have meanings as in the ones tied to cognition.

2.5 Computation and Cognition

I now look at the relationship between Chalmers' account of implementation and cognition. Chalmers introduces the following two theses that link his account of

implementation using CSA and cognition they are a) a thesis of computational sufficiency and b) a thesis of computational explanation. The first thesis says that having the right kind of computations suffices for the possession of a mind and cognition. On the other hand the computational explanation thesis says that computation so construed provides a framework for explanation for cognitive processes and behaviour. I discuss them one by one below.

To discuss these theses I need to explain two of Chalmers' notions on which they depend, namely abstract causal organisation or causal topology and organisational invariance. Chalmers' theory of implementation (as discussed earlier in Section 2.3 above) posits an isomorphism relationship between the computation being implemented and the causal structure of the physical system implementing the computation. Hence Chalmers describes a computation as an "abstract specification of causal organisation"[6] of the implementing system. This abstract specification of the causal organisation he refers to as the causal topology. In addition Chalmers posits that mentality and cognitive properties are organisationally invariant properties. Organisationally invariant properties of a physical system are those which depend on the causal topology alone and not on the nature of the physical system. Furthermore they remain invariant with respect to causal topology *i.e.* any change or distortion that does not affect the causal topology does not affect mentality and cognition. For example he contrasts mental and cognitive properties against the properties of digestion and flying as follows. The former do not depend on the underlying physical substrate while the latter depend on the physico-chemical makeup in the case of stomachs and factors like height and speed (among other aerodynamic ones) in the

[6] Chalmers (2011, p 331)

case of flying. The latter group stop functioning at some point: if in the stomach's case parts were replaced by metal and in the flying case if the height gets to ground level. While cognition and mental properties are dependent on the abstract pattern of causal organisation alone *i.e.* they can be specified by a CSA description. It is in virtue of implementing these "right kind" of computation that the system is cognitive and particular mental states have a one to one correspondence to computations such that implementing those computations realises those mental states. Computational sufficiency implies that a particular computation for a cognitive process can be implemented in different physical substrates as long as they have the same causal organisation and therefore they would have the same CSA description.

The thesis of computational explanation also depends on causal organisation and looks to providing an explanation of cognition and behaviour based on specifying a description of the underlying computational structure in terms of a CSA description.
In Chalmers (2012) he says that this maybe too strong and needs modification as it will not work in all cases of cognition. In his response to Egan (2012) and Rescorla (2012) Chalmers qualifies this thesis when he concedes that explanation in cognition using mathematical functions ("function-theoretic" as he calls it) like explanation of edge detection by Marr in terms of computing the Laplacian of the Gaussian of the retinal array is not captured well by his CSA model of computation. I briefly expand on this without going too deeply into Marr exegesis. In image processing edge detection is one of the important tasks (HIPR 2000). Edges carry useful information about an object than that available from other features such as texture and colour. Edges are generally drastic changes of image brightness over a small spatial distance. The challenge in edge detection is to distinguish edges from other features in the

image such as textures and especially noise, so that all edges are detected while the noise is suppressed. In Marr's technique the noise is first suppressed by smoothing the image by applying the Gaussian operator and then applying the Laplace operator for edge detection. Both the Gaussian and Laplacian operators are two mathematical operators which Chalmers concedes are not well covered by his model of computation.

The other example Chalmers gives as an exception to his explanatory framework when responding to Rescorla (2012) is that of Bayesian models in perceptual psychology. These rely on statistical theory for estimating prior and posterior probabilities to posit hypothesis about the perceptual field. The Bayesian approach assumes that cognition is approximately optimal in accord with probability theory. The mind has representations for statistical correlations and conditional probabilities. It has the capacity for probabilistic computations such as applications of Bayes' theorem. Applying probabilistic computations to statistical representations accomplishes mental tasks such as perception.

Chalmers argues that while these techniques are useful for individual areas of cognition *e.g*. vision and perception but they do not give a general account for cognition and the mind in general which is his project. Furthermore he argues that as the Marr and Bayesian theories are computing mathematical functions they are like a black-box input- output tool far removed from the internal states and mechanisms of cognition unlike his theory which is at a lower level involving states and therefore "mechanism ready". This mention by Chalmers of higher and lower levels of explanation links up with a well known meta theoretical view of Marr (1982) on cognitive architecture. Briefly, Marr posits three levels of explanation for a cognitive

function. The first is the computational level. This is the highest level which answers the What/Why question about the role of a function for an organism in its environment. The next is the algorithmic level which is at an intermediate level. It seeks an answer on how a function is performed and is akin to the software in an implemented program. Finally at the physical level is the implementation level like the hardware of a computer where the algorithm is implemented. In his discussion of Klein, Egan and Rescorla's papers Chalmers (2012, p. 222 & pp. 243-248) posits that his theory was at a lower level just above the physical neurobiological level while the ones of the above three were at an intermediate level closer to Marr's algorithmic level.

Chalmers generalises by saying that representational, function-theoretic, teleological and social explanations are "higher level explanations". They are not covered by his model of computation. But they are not incompatible with his causal model and the connections between them could be worked out possibly if one moved from a CSA framework to a framework of Abstract State Machines (ASM). This is a new machine architecture which enables the setting up of a hierarchy of machine formalisations and their connections to define higher level cognitive functions to intermediate and lower level ones with the connections between them set out. The top of the hierarchy would correspond to higher level cognitive functions and the lower levels to the ones corresponding to CSA states. I discuss ASM in more detail in Chapter 4.

Note that the theses of computational sufficiency and of computational explanation can be held independently as we can take a pluralist approach to mentality and

therefore on the one hand deny the thesis of computational sufficiency but at the same time accept the thesis of computational explanation. Doing this implies taking an instrumentalist view on the role of computation and the mind. Equally one can accept the thesis of computational sufficiency as an account of the mental and be pluralistic about explanation which is how Chalmers (2012) is heading.

2.6 Merits of Chalmers' Causal Thesis of the Mind

Chalmers' causal thesis of the mind has three key merits. Firstly it has benefits over earlier theories of the mind such as the semantic version of the Computational Theory of the Mind (CTM) developed and proposed by Fodor (1975) and his followers following Putnam (1960). Secondly it claims to overcome the triviality arguments against CTM launched by Putnam (1988) and Searle (1992). Finally it claims to provide a foundation to cognitive science by defining the role of computation in a theory of cognition. I discuss each of these below.

Fodor's (1975) version of CTM is linked with his Language of Thought hypotheses. It holds that mental states are representations or tokens in a language of thought which is like a natural language. The mind manipulates tokens of this language like a computer *i.e*. the manipulations are by computations. Fodor's account makes it possible to answer the question of how thought and higher cognitive processes such as abstract thinking and language processing are possible in terms of processing strings of symbols computationally and providing them content with referential relations to entities in the world. Fodor's (1975) version of CTM is tied up with mental representations and their semantics such as truth-value, reference, content etc.

2. Chalmers on Implementation and Cognition

According to his view cognitive states and processes such as thinking, reasoning and imagining are constituted by the occurrence, transformation and storage in the mind of mental representations of one kind or other for mental objects such as thought, mental images etc. To quote Fodor (1975, p. 198): *"Mental states are relations between organisms and internal representations, and causally interrelated mental states succeed one another according to computational principles which apply formally to the representations.* A little earlier on the same page he says. *"So having a propositional attitude is being in some relation to an internal representation. In particular having a propositional attitude is being in some computational relation to an internal representation."*

As pointed out in Section 1.0 earlier Fodor's theory had a major weakness in that it could not offer a clear connection between computation at the abstract representational level and computation at the physical level. It could not explain the implementation relationship which Chalmers (2011 pp. 323-326) has argued as crucial for making computation foundational for cognition and explaining behaviour. Fodor's connectionist critics *e.g*. Rummelhart (1998) are not convinced by his plea that his theory is at a higher level and not intended to provide neural realisations. Connectionists view his CTM as biologically implausible. The problem with the semantic version of CTM had been made worse by the advent of well supported connectionist models of cognition which operated at a sub-symbolic level (Rummelhart et el 1986).

In their response Fodor and Pylyshyn (1988) criticise connectionism by arguing that there are some features of cognition such as the productivity and systematicity of

language which require a symbol system and cannot be accommodated within a connectionist model. The productivity of thought or language is the capacity human beings have to produce indefinitely many different thoughts/ sentences of a learnt language. The systematicity of thought or language is the capacity for mastery of the syntax of one's native language. Fodor and Pylyshyn (1988) argue that this is missing from connectionist nets as the representations they have don't have the structure to be compositional. Compositionality is a feature of rich symbol systems, similar to natural language. Fodor notes that while minds are finite and therefore the number of simple ideas can be finite yet we can construct almost an indefinite number of complex ideas / thoughts. This he refers to as the productivity of thought and the mechanism for doing it is compositionality. Compositionality ensures that the contents of complex concepts are determined by the content and arrangement of their simple constituents with the semantics of the former determined by the semantics of the latter. Furthermore it is not just in the structure of the concepts but also in the content that this productivity of thought applies.

Note that Fodor and Pylyshyn's criticism does not apply to all connectionist theories. Smolensky (1990, p.166) has developed a version of connectionism which could account for higher level cognitive structures like language without including a full blown language of thought. The semantic version of CTM had a problem explaining connectionist models as the question of symbolic processing and discrete mental representation did not apply to the connectionists. Also as mentioned above it could not give an account of how physical implementation occurred thereby facing the charge of biological implausibility. However this does not pose a problem to Chalmers' thesis which operated at a purely syntactic level by appealing to an abstract

computational object namely CSAs and explaining the implementation relationship between abstract and physical computation by the causal thesis.

Chalmers also claims to overcome triviality objections against the computational theories of the mind raised by Searle and Putnam. I discuss the triviality objection and Chalmers' response in the next section (2.7).

Cognitive scientists and AI researchers make heavy use of computation in the models they develop for understanding cognition and the mind. In the course of this they believe that they are not just simulating but replicating these cognitive processes in their models. However critiques such as Searle (1980), Dreyfus (1972) and Penrose (1990) to mention three raise questions about the role of computation in a theory of cognition and throw doubt on whether a replication of a cognitive process is really possible. They suggest the models were merely at best doing simulations. Chalmers responds that if a model shares the causal topology of the cognitive process then it is replicating the process. By establishing the theses of computational sufficiency and computational explanation Chalmers feels he has established the foundational role of computation in cognition thereby answering these critics.

2.7 Does the CSA Formalism Escape the Trivialisation Objection?

Searle (1992) and Putnam (1988) argue against CTM by saying that computations have no independent reality in the scheme of things but are observer or mind dependent and exist for our convenience. They argue that a complex machine can

implement all possible computations, with the computations it implements being mind dependent. They thereby question the reality of the implementation relationship. This is the trivialisation objection and it claims that every complex object such as a wall or a rock could be described as an inputless FSA hence doing all possible computations but without any inputs or outputs. The arguments are based on the contention that the notion of a physical system implementing a computational formalism is overly liberal to the point of vacuity and hence of no use in grounding mentality and behaviour. One key assumption in both is the idea of a mapping correlation between physical states of the physical system and computational states of the FSA.

Searle claims that Putnam's (1960) original account of FSAs is so lax that any complex object such as walls could be given computational descriptions without inputs or outputs and appear to be performing a computation. In fact he claims that a complex system such as a wall implemented a WORDSTAR program. Putnam (1988) developed the formally argued objection which said that every ordinary open system implemented every inputless FSA thereby trivialising CTM. Putnam's argument is that given any machine table of a FSA, every physical system implements the FSA by having mappings of the machine table to internal states of the physical system within a given time interval. For example if the machine table stated that state A is followed by state B then every instance of A is followed by state B in a time interval. This can be described as based on the material conditional of the form "*If A then B"*.

Chalmers' (2011, p.333) response to Putnam's argument is based on an appeal to law like counterfactuals supporting the notion of reliable causation associated with the implementation of a transition from one state to another in a CSA. He argues that Putnam's argument is based on the use of weak material conditionals for the

implementation of a FSA as seen above. While Chalmers' thesis requires stronger causally reliable ones *i.e.* one's that can support law like counterfactual conditionals and are not merely coincidental or corelational.

As he says: Chalmers (2011, p. 333)

"Given a formal state-transition A → B, it must be the case that if the system were to be in state A, it would transit to state B. Further, such a conditional must be satisfied for every transition in the machine table, not just for those whose antecedent states happen to come up in a given time period."

Chalmers is asserting what I had discussed in Section 2.1 earlier that given a machine table with a set of state transition rules, the use of the counterfactual conditional makes the rules true in all cases of the machine table and not just the ones that get instantiated in a time period. The notion of reliable causation based on law like counterfactual conditionals was discussed earlier in the discussion on Chalmers statement of the implementation of CSA in Section 2.3 and in Section 2.1 above. Because of using CSA instead of FSA the computational descriptions are no longer lax and are sufficiently constrained to reduce the probability of any complex object such as walls implementing a particular computation to be almost improbable. The requirement of reliable causation in CSAs does the work of imposing the rigour required via counterfactual conditionals in preventing complex objects like walls implementing a given particular computation while no doubt it will implement some computation(s). By introducing CSA Chalmers argues that while there could still be residual false implementations however the probability of such trivial cases is dramatically reduced (Chalmers 2011, p. 331). For example Chalmers estimates the probability for an arbitrary system to meet the requirements of a CSA whose state

vectors have 1000 elements with up to 10 possible values for each element and a similar number of state transition rules as less than one in $(10^{1000})^{10^{1000}}$. While this probability is a very small number, note that for a CSA with an infinite or a very large number of vector states this probability could still amount to a large number. Therefore while CSA reduces the problem for practical purposes but in theory it does not totally eliminate it.

2.8 Objections and Replies

In the Journal of Cognitive Science (Vol. 12, 2011 and Vol. 13, 2012) a number of commentators provide a critique of Chalmers' views from a number of different angles. In general most agree with the basic intuition behind Chalmers' causal theory of the mind and see it as an improvement on a semantic view of CTM like Fodors'. While they see shortcomings and weaknesses in the detail of Chalmers' work from different aspects however in my view no one gives a knockdown argument and in return Chalmers (2012) offers a robust defence. There is only one area where Chalmers' accepts a need for revision and makes concession and that is on the thesis of computational explanation. I discussed this earlier in Section 2.5 above.

I list here some of the main issues raised by Chalmers' critics:

1) Is Chalmers' notion of implementation over liberal thereby leading to too many systems implementing too many computations without complying with their causal structure and explaining their behaviour?

2) Is CSA an appropriate formalism for capturing all different types of computation such as Turing machines, Pascal programs, etc and what are its shortcomings?

3) Does Chalmers' CSA formalism in fact manage to escape the triviality objection of Searle and Putnam?
4) Is computation sufficient for cognition and mentality?
5) Does computation provide an explanatory framework for cognition and behaviour?

In what follows I will discuss 1) and 2) and 4) below with 3) and 5) already discussed earlier in Sections 2.7 and 2.5 respectively. Note that the discussion referred to earlier on objection 3) and 5) cover the subject of the objection and not necessarily the specific arguments of particular commentator(s) from the Journal of Cognitive Science (2011 & 2012).

1) Is Chalmers' notion of implementation over liberal?

Some critics[7] argue that Chalmers' definition of CSA can lead to an over liberal interpretation of computation so that too many systems end up implementing too many computations in a way that does not reflect their real causal structure and does not explain their behaviour. Chalmers[8] had leant qualified support to such a view by saying in his paper that any sufficiently complex system will implement a number of computations. However he had gone on to say that this was benign as long as every system does not implement every possible computation or any given computation as Putnam and Searle had argued. The worry here maybe that Chalmers' views endorses panconputationalism[9] or the thesis that everything is a computation under some description with Towl (2011, p. 428) giving an example of a pool table doing computations. Chalmers agrees, rightly in my view, that his view endorses a benign version of Panconputationalism. Such a view says that every complex physical system

[7] Ritchie (2011) and Towl (2011)
[8] Chalmers(2011) pp. 331-332
[9] I discuss what I see as the main issue and not individual points of Ritchie and Towl.

may implement multiple computations however he does not see it as endorsing blatant permissiveness (i.e. not every possible computation). This is due to the constraints imposed by the rigour of CSA definitions via the requirement for reliable causation and related counterfactual requirements of state transition rules which do not permit random mappings of physical states to computations. Furthermore it does not make his causal theory of the mind trivial because of the strong thesis of computational sufficiency based on causal topology and organisational invariants defining the mental so that it is not attributing mentality to all matter. Chalmers is open to the idea that there maybe a need to put constraints on how implementation is to be applied if it provides a more precise definition and has hinted at constraints such as spatial, functional and teleological ones among others without developing these ideas in detail (Chalmers 2012, p.242)). His theory involves law like transitions in his talk of "reliable causation" and not mere mappings.

2) Is CSA an adequate formalism for different types of computation?

Chalmers says that he chose CSAs for his implementation thesis as it was abstract enough and powerful enough to capture all different types of computational formalisms such as Turing Machines, FSAs, cellular automata and probably can be extended to other models such as PASCAL programs, declarative and imperative programming languages etc.

Chalmers' critics[10] argue on two fronts on this issue namely that:

a) CSA's are too powerful and not a purist computational model;

b) CSA's are too general and miss out the detail in the architecture of a Turing machine;

[10] Sprevak (2012) and Milkowski (2011)

On the first point Chalmers agrees that an unrestricted CSA is effectively a "super-Turing Machine" able to compute computable and non-computable functions thereby not a purist computational model like Turing Machines. However he argues that he is not advancing the theory of computation nor is there anything that says that a computational model of the mind should be based on a purist model of computation when all that he is hoping to achieve is a consistent and coherent account of the implementation. Chalmers also says that by restricting CSAs to finite starting states and inputs we can get a sub-set of CSAs which correspond to Turing Machines. Furthermore he argues he sees nothing wrong in giving an account covering both super-Turing Machines and Turing Machines.

The other criticism made against CSA's by Sprevak (2012, p. 126) was that they obscured the architecture of a Turing Machines which were conceived as having a head and a tape with the head moving serially a step at a time reading and writing one square of a tape. Furthermore there was a distinction between control and data. All this detail was lost in a CSA which was just a giant table with a state or states covering each of these factors. While a physical parameterisation could handle some of the features it was unlikely to represent all the concepts within different architectures. Chalmers agrees that while this is a more worrying difficulty than the previous one however he points out that to some degree it is to be expected as the CSA was a more abstract computational object than the other formalisms. It was only because of the CSA's abstractness and generality it was able to provide translations of the different computational formalisms such as Turing machines, FSAs etc. However the translations are not to be seen as replications of the target architectures. While Chalmers sees this mattering at the level of models and theory however at the

implementation level he does not see this loss of distinctions and detail being important. In Chapter 4 I discuss Abstract State Machines which Chalmers (2012) endorses in the face of criticisms of CSA. As will become clear in my discussion there a move to ASMs would answer Sprevak's critique of CSAs as ASMs are far more granular and flexible.

Milkowski raises the issue of breakdown saying that all mechanical devices are subject to breakdown in their lifetime and the CSA translation of a TM will not have any state representing a head or paper tape breakdown and therefore is not an adequate representation of real live implementation. Chalmers argues that on his account the device stops computation at breakdown. He is giving an account of implementation for some points in time and not for all points in time *i.e.* forever. Since Chalmers' views are developed to give a framework for a theory of the mind in general and cognition in particular then the reservation I have about Chalmers' reply is that his view (to stretch the analogy) does not apply when there is a breakdown in physical structure or function *i.e.* in the brain or mind either as a result of accident or birth. This would be a weakness in his position as accounting for defective function should be part and parcel of a general theory of the mind and cognition.

4) Issue with Computational Sufficiency[11]

The main issue people have with Chalmers' computational sufficiency thesis is the idea of abstract causal organisation where in the pattern of interactions of the parts of a system one could abstract away to a fine grained level where all one has left are computational properties related to mentality. How such abstraction occurs is a worry as Chalmers offers no clues. Piccinini (2010, Note8, p. 275) argues that unless an

[11] Note: Issues 3 & 5 as mentioned above were covered in Sections 2.7&2.5 respectively.

explanation of such abstraction is forthcoming from Chalmers we are left with the belief that either such abstraction will also include material properties *e.g*. related to bodily processes such as digestion or it will capture some but not all mental properties. Note that the detractors here are raising a worry about how such an abstraction process occurs so as to be plausible and are not giving a knock down objection. To reiterate, in the computational sufficiency thesis Chalmers is drawing a line between cognition and the mental on the one hand and the rest of material things on the other hand. He does this by saying that for cognition and mentality computation is a sufficient property because of the organisation invariant properties of cognition and mentality while for other things like his stock examples of flying and digestion, computation is insufficient and requires other physical properties for constituting them. I think a valid objection to his thesis would be for the detractors to argue that computation is insufficient for mentality and cognition and this has not been done.

2.9 Conclusion

This chapter has discussed Chalmers theory of implementation based on his account of the computational object of a CSA and its relation to mentality and cognition. I have argued that Chalmers' theory though not perfect has advantages over previous theories of the mind and it meets the trivialisation objections of Putnam and Searle plus has been robustly defended by Chalmers.

In the next chapter (3) I discuss the nature of neural computation based on current accounts of the functioning of the brain from neuroscience. In Chapter 4 I will look at

whether Chalmers computational model of the mind can be made compatible with neural computation as discussed in the next chapter.

3. Chalmers' Causal Theory of the Mind and Neuroscience

This chapter discusses the nature of neural computation based on current accounts of the functioning of the brain from neuroscience.

3.1 Introduction

Cognitive Science and related disciplines such as Cognitive Neuroscience are the fields that address questions about cognitive functions and their performance by the brain. The brain is the organ that realises mental activity (Piccinini and Bahar, 2013). It does this via signals consisting of action potentials or spikes which are organised in sequences called spike trains resulting from the activity of neurons firing. For large organisms the performance of cognitive functions is done by the firing of population groups of neurons and their patterns of firing in well defined regions of the brain.

McCulloch and Pitts (1943) working in mathematical bio-physics and influenced by Turing were the first to propose that neurons are a binary switch, neural activity is computational and that neural computation explains cognition (Piccinini and Bahar, 2013, p. 467-468). This is the earliest version of a computational theory of the mind and looked upon the brain as a computer with computations undertaken by neural activity to carry out mental processes. Their methods and techniques are superseded by modern ones. However the general notion of the processing of spike trains as computation has survived in the Cognitive Science and Neuroscience disciplines. When people talk of neural computation in the literature they are basically referring to the processing of spike trains.

If we ask whether neural processes perform computations in the sense employed in Chalmers' theory then at the most basic level the question is whether spike trains are:

i. a digital computation **or**
ii. an analog computation **or**
iii. a type of computation not covered by our current paradigms of computation **or**
iv. not a computation.

Of the above four cases only the first case supports Chalmers' theory. The second and third one would require a revision of Chalmers' theory and the fourth case negates Chalmers' theory. I discuss this in detail in Section 4.2 below but briefly digital computation assumes discrete values in the input, state and output parameters making up the CSA while analog computation would assume continuous parameters. Chalmers (2011 p. 347) has implied that the CSA model as it stands would have to be modified to handle continuous parameters. The third case above is a hybrid case which requires a modification of Chalmers current model to handle both discrete and continuous parameters. Finally the last case is a straightforward negation of Chalmers' view as it is denying any role for computation.

While at this stage of my thesis it is an open question which type of computation occurs in neural processes however it would be a significant result for Chalmers' theory of physical computation (implementation) if it can be shown to be compliant with the data of neuroscience. If on the other hand Chalmers' computational views cannot accommodate neural computation then one of the things to look at is if it can be modified so that it applies at a higher abstract level *i.e.* at a neural circuit or

function level. I discuss the compliance and modification of Chalmers' theory in the next chapter.

3.2 Neural Structures

Nervous systems are found in all multi-cellular animals except very simple ones like sponges. While they vary greatly in complexity, the basic structure is widely homologous across species reflecting a common evolutionary ancestry. The Central Nervous System (CNS) is an electrochemical system consisting at the cellular level of nerve cells or neurons with their structures such as dendrites, synapses and axons. The CNS[12] consists of the brain and the spinal cord in vertebrates and is responsible for controlling all cognitive functions.

Neurons work like a switch (Byrne 2013). They are quiescent as long as the electrical voltage across their membrane is below a threshold. They get electrically excited and fire when the voltage reaches a threshold level under electrochemical stimulation. This excitation occurs by electrical or chemical signals from other neurons or as a result of sensory stimulation. The electrical signal resulting from a neuron firing is called an action potential or spike as recordings of it are shaped like a spike. These spikes are all or nothing and flow across nerve fibres or axons and are passed on via the neuron's synapses which act like a terminal in an electric circuit. The signal or action potential is passed on down the axon and is a brief event of about a millisecond only. Electrical activity across synapses is facilitated by chemical activity of neurotransmitters and hormones. There is a short refractory period after a spike when

[12] The introductory paragraphs on neuroscience are based on Piccinini & Bahar (2013), Dayan and Abbott (2001) and Kandel, Schwartz and Jessell (2000).

neurons cannot get electrically excited. Neurons recover after the refractory period for firing the next signal when the input voltage again reaches the threshold.

A sequence of spikes fired by a neuron as a result of stimulation is called a spike train. A longer duration stimulus to a sensory organ will result in a succession of action potentials the frequency of which will depend on the intensity of the stimulus. Spike trains from a single neuron do not have sufficient useful information for a neural function. The minimal neural processing unit for carrying out a neural function are a few dozen groups of neurons working together according to estimates of Shadlen and Newsome (1998). Smooth operation of a neural function is achieved by neurons working in inhibitory and excitatory neighbouring groups by sending inhibitory or excitatory signals to manage the dynamics of neural networks. Neurons can also be distinguished by their firing pattern. The firing pattern from individual neurons can either be a regular train of spikes or in bursts of spikes. The firing pattern can also be either in phase with (*i.e.* synchronous) or out of phase with (i.e. leading or lagging) other neighbouring neurons. Phasic firing pattern generally occurs when they are carrying out a common neural function.

Information about the outside world is acquired by the sensory organs in various sensory modalities such as visual, auditory, olfactory to name a few. The brain has developed multiple coding systems which render this sensory information into a suitable code. This is done by using a feature of the information about the world captured in the sensory modality of sounds or sight or smells and coding it into a variable in the neuron's action potential or spike. This process is called modulation and it can be explained further by a reference to radio engineering where the term

originated. Note that in my discussion I look at the two main coding systems in the brain namely rate coding and temporal coding.

In wireless transmission modulation is a means of encoding information in a carrier signal (Terman 1960). Carriers are signals using different waveforms such as sine waves, saw-tooth waves, square waves and pulses. Different waveforms are used for different applications such as radio, telephony, TV and radar. The process of encoding information in the carrier is called modulation. The information encoded can be of different formats such as audio, video, text and messages. Modulation involves varying one of the variables of the carrier such as the amplitude, the frequency or the phase using the information for transmission as the parameter. For example in Pulse Frequency Modulation (PFM) the carrier waveform are pulses of constant amplitude and duration but whose frequency is varied to encode the transmitted information.

Neuronal spike trains can be viewed as a modulation with the spike firing rate encoding the neuronal information while spike amplitude and duration are constant and not used for modulation. The rate of firing of the neuron is varied and is one of the main means of coding information in the neural signal and transferring it across the brain[13]. This coding method is called rate coding. It is akin to radio signals using pulse frequency modulation of a carrier wavefront to convey information where the firing rate of pulses are varied to code the information to be transmitted. This analogy makes no commitment to neural signals either as a PFM or more broadly as an analog process. In rate coding the rate is a measure of the average number of spikes over a duration measured in milliseconds.

[13] (Piccinini & Bahar 2013, p. 462), Byrne (2013 Fig.4)

The other main coding scheme called temporal coding applies when the neural information is coded in the precise timing of the spike train pattern. It is generally used when coding rates are high (*e.g*. in vision) so that the stimuli have to be processed more quickly than rate coding allows. When information is based on high frequency fluctuations in firing rates it could easily be mistaken for noise in a rate coding model but in the temporal coding model the information is better discriminated. For example if 1 indicates a spike and 0 = no spike then a spike pattern of 011001111001 has different information from a spike pattern of 011101110001 even though the average rate is 6 spikes per 10 milliseconds.

There are other coding schemes. However for our purposes just illustrative discussion of the two important ones is sufficient. Overall it seems that the information in the spike train is mainly conveyed in some combination of their dynamical properties such as the average firing rate, or the timing of the spikes (i.e. the inter-spike interval or phase shifts).

3.3 Nature of Neural Computation

In this section I discuss the nature of neural computation. The discussion is mainly based on the work of Piccinini and Bahar (2013) who argue that "neural computation" is a hybrid type of computation, neither a digital nor an analog computation.

To address this issue the structure of this section covers the following five points:

1) What is digital computation?

2) Based on Neuroscience data can neural activity be considered a digital computation?

3) What is an analog computation?

4) Based on Neuroscience data can neural activity be considered an analog computation?

5) If neural computation is neither digital nor analog then is it *sui generis?*

Each of the above 5 points is addressed one by one below.

3.3.1 What Is A Digital Computation?

A digital representation is one which uses discrete set of elements or values of entities or variables represented by symbols of a finite alphabet or natural numbers or integers. Digital computation uses discrete step wise changes over a set of elements according to an abstract rule over them to solve or process changes (Piccinini and Bahar 2013, pp. 459-461). In its physical implementation digital computation has three main characteristics.

The first characteristic is the processing of input strings from a finite discrete alphabet to suit the hardware circuitry underlying the physical device. Digital computation generally uses a code such as binary or octal or hexadecimal numbers to give a few commonly used examples. The second characteristic of digital computation is a finite number of internal states and symbols of the machine and a finite number of instructions or rules in terms of those symbols. The internal states are set up as a machine table to manipulate the input strings (as in the FSA machine description in the previous chapter). Furthermore the rules and vehicles or entities which form the alphabet strings are medium independent. They are insensitive to any concrete properties of the physical medium or hardware on which they are implemented thereby making multiple realisability possible *i.e.* the same computation being implemented on different media or machines such as for example mechanical, electronic or magnetic. The third characteristic of digital computation is the

production of output strings resulting from the manipulation of the input strings. The output is in the same or a different alphabet from the input.

The key to abstract digital computation is the ability to convert all input, rules and output discrete elements so that it can be interpreted as a number for use in the internal inputs and states of the physical machine. A practical device will have all sorts of bells and whistles such as memory, storage communications networks etc. depending on the architecture on which it is designed. These however are not relevant for our purpose.

3.3.2 Why Neural Computation is not Digital Computation

We have seen that neural spikes have an all or nothing character as they are either present or absent. This gave rise to the consideration of an analogy between spikes and digits (originally by McCulloch and Pitts)[14]. Digital computation requires the manipulation of strings of digits hence in the neural case continuing with the analogy the candidate would be spike trains either from a neuron in their temporal order or from synchronous neurons within a suitable time interval. Synchronous neurons are the ones that are anatomically or functionally located and participate in a neural circuit to work in unison. However there are strong dissimilarities between spikes and digits.

Piccinini and Bahar (2013 p. 469) argue that spikes are not digits and manipulation of spike sets is not digital computation[15]. Their assertion in a nutshell is that while digital computation requires the manipulation of strings of digits *i.e.* the concatenation of representations like natural numbers or integers, they note that: *"Neural spikes are not*

[14] Piccinini & Bahar (2013 p. 467)
[15] Note: I do not discuss all the arguments presented by Piccinini & Bahar (2013).

digits, and even if they were digits, spike sets would not be strings". (Piccinini and Bahar 2013 p. 469*)"*

Three proposals for neural activity as digital computation are considered by Piccinini and Bahar (2013) and then rejected on the evidence available. They are 1) Single neuron spike activity; 2) Spike rates as digits; 3) Fitting Spikes into Strings. I discuss each of these in detail below.

1) Single neuron spike activity:

One proposal is to consider the firing of a neuron as digital computation as this gives rise to two states: the presence or absence of a spike. This superficially could be mathematically typed with two digital values and investigated further as a digital computation. However Piccinini and Bahar (2013, p. 470) provide two reasons why this digital typing is not correct.

The first reason is based on not being able to identify a functionally significant time interval for neural activity. For digital representation only a finite number of digits can be typed within a given time interval. Therefore the typing of the presence or absence of a spike as digits is based on the assumption of a fixed time interval between spikes in order to map a finite number of digits to spikes. But this assumption is questionable as neural firing in vivo occurs with a high degree of variability (Piccinini and Bahar 2013, p. 471) and is not based on a fixed time interval. This leads to a move for spiking from a deterministic to a probabilistic one. While digital computation can be probabilistic, events are temporally discrete so that for a fixed time interval there can only be a finite number of outcomes with different degrees of probability such that a finite number of digits can be assigned to each

outcome. But in the neural case there is no fixed functionally significant time interval for counting spikes. Proposals like using the synaptic delay or a master time interval were found to be wanting as in the former case there was lot of variation in the time and there is no empirical evidence for the latter. Spike counting would then have to be done in real time and spike probabilities assigned over an infinitesimal period. This leads to the possibility of an uncountably infinite number of times in a finite time interval during which a spike could occur. For a non-deterministic set up this leads to an uncountably infinite number of probabilities for the occurrence of spiking events. If spikes are typed as digits this would require the assignment of an uncountably infinite number of digits as there is a background assumption that different times in an interval correspond to different digits. However by definition of digital processing there can only be a finite number of digits in an interval for a digital value to be correctly assigned. Hence the analogy between spikes and digits breaks down and the presence and absence of a spike for digital typing is not workable.

Next they argue that while the presence or absence of a spike has a differential impact in a spike train, what is of functional significance neurally is the rate at which spiking occurs. In many cases neuroscientists assess the functional significance of a neuron by computing the average firing rates over many trials (Piccinini and Bahar 2013, p.472). Furthermore, especially in rate coding, individual spikes may be removed or added to a spike train without losing their functional significance (Dayan and Abbott, 2001; Chapter 1). Hence they argue we cannot attach significance to the presence or absence of a spike so as to treat that as a digit.

2) Spike rates as digital states:

Von Neumann's proposal was that we take the spike rate as a digital value. As the spike rate varies with the stimuli received the digital value would also vary. This proposal had the merit that it proposed using the functionally significant variable from a neural viewpoint. However as spike rates vary continuously there is no functionally significant time interval (Piccinini and Bahar 2013 p. 472-473) which can be used to parse *i.e*. quantize the spike rate in a reliable manner such that a digitisation is possible.

Piccinini and Bahar (2013, p. 473) report that Von Neumann states that there is a limit to the precision of spike rates thereby making a digitisation from spike rates unreliable. Hence they argue the idea of using the spike rate as the basis for digitisation has to be given up.

3) Fitting Spikes into Strings

In the previous two cases Piccinini and Bahar (2013) argue why spike trains from single neurons cannot be considered as a digital computation. In the case discussed below Piccinini and Bahar (2013, p. 473) argue why even the functioning of groups of neurons cannot be looked upon as a digital computation.

A basic uncontroversial assumption in digital computation is that any non-trivial computation requires a finite number of strings of digits concatenated together to carry out a function whether a program instruction or some other operation such as a FSA internal table state. A single digit or even a pair of digits as a rule serves no useful purpose. If neural computation were a digital computation then it should be

possible to unambiguously decide which sets of spikes belonged to which strings of digits. However Piccinini and Bahar (2013, p. 473) argue that this is not possible.

They cite two sub-cases here namely:

a) Spike train synchrony

Consider sets of spikes from neurons associated with a cortical column engaged in carrying out some neural circuit or function. These columns are as a rule spatially and anatomically contiguous and have a neural function performed by groups of neurons working in synchrony *i.e*. in the sense that there spike trains begin and end in unison. The proposal was to consider the spike trains from such synchronous neurons as strings of digits. The complication here is the fuzziness in defining neural structures. Determining the boundary for a neural structure on a neuron by neuron basis is not possible. However the main issue is that such neural synchrony is a matter of degree and therefore the spike trains of neurons from neural structures will be in phase over some time intervals and out of phase (either leads or lags) over other time intervals, with no meaningful way of finding a functionally significant time interval where they will be in perfect synchrony. Therefore looking at spike trains from synchronous neurons as concatenation of strings of digits is not possible.

b) Spike trains from individual neurons.

A spike train has a temporal order but in order to treat a spike train as a string we need to identify the start of the string *i.e*. the first spike and the end of the string *i.e*. the last spike. In neural activity there is a high stochastic or random background or "noise" signal such that the neural signal cannot be parsed out from the noise in a functionally significant way to identify the start and end of a spike train. I am not totally convinced by this argument of Piccinini and Bahar (2013) as in theory the signal should always be distinguishable from the power spectrum of the noise. Furthermore

in non-linear circuits (including neural) a stochastic resonance (Piccinini and Bahar 2013, p. 472) effect occurs where the noise can aid and enhance the signal.

The second point they make is probably more significant and that is that spike trains from individual neurons are not functionally significant. Their significance is discovered by averaging out spike trains from single neurons over many trials and then use the average spike trains as units of functional significance. Colin Klein has remarked (in conversation) that this is a confusion between the functional significance of a spike train and the process to discover the functional significance of a spike train. Based on the above three points Piccinini and Bahar (2013) reject spike trains as a digital computation.

The purpose of this section was to discuss reasons why spikes and spike trains are not a digital computation. It was based on reasons given in Section 7 of Piccinini and Bahar (2013). While the reasons are of varying degrees of conviction however all up they make a case that spikes and spike trains are not a digital computation.

3.3.3 What Is An Analog Computation?[16]

An analog structure is one which uses continuous representations such as the real numbers so that the values of a function using these representations can vary continuously with respect to some variable(s). Analog processing could use these continuous representations to solve or process a continuous function. Some of these functions may be set up as differential equations of multiple variables varying over real values. Analog computation is the application of analog processing either in nature or

[16] Piccinini & Bahar 2013, p. 461

by design in a device. While some analog computers are systems which are setup for finding solutions to differential equations this need not be so for all cases of analog processing. To give a stock example of a looser kind of analog processing even the old AM wireless radio is an example of an analog processor. It detects the audio signal from the broadcast radio frequency carrier which is a continuous signal (or representation). The audio signal, again a continuous representation, is amplified and converted to a sound wave for listening to a radio broadcast.

There is a clear contrast between digital and analog computation which can be intuitively grasped and it is as follows. In a digital computation there will be a clear difference from one state to the next state with no intermediate states. On the other hand in an analog computation, theoretically there will be an uncountable number of intermediate states between two states as it uses continuous functions ranging over real numbers.

3.3.4 Why Neural Computation Is Not An Analog Computation?

Piccinini and Bahar (2013, pp. 465-467) argue that neural processing is not an analog computation for the following reason. They concede that there are neural processes of a continuous nature over real time. These are the release and uptake of neurotransmitters and hormones, plus continuously variable voltages transmitted by dendrites and some axons. These may look like analog processes and in this loose sense the brain may appear to be an analog machine. But they argue that for the brain to be looked upon as an analog machine in a strict sense its functionally significant signals must be "irreducibly" continuous variables. Now from earlier discussion in this chapter we know that spikes are the main functional signal of neural activity.

Furthermore what is of value is the dynamical attributes of a spike such as its firing rate or spike timing and not any physical attribute such as its threshold voltage or the rise / decay time. The mathematics of the operation of these functionally relevant neural factors (viz. firing rates and spike timing) as set out in theoretical neuroscience does not correspond to the mathematics of analog computation. A new mathematics had to be invented to account for their operation (Dayan and Abbott, 2001). Hence Piccinini and Bahar (2013, pp. 465-467) argue neural computation is not strictly an analog computation.

3.3.5 Is Neural Computation Hybrid?

Based on the survey of the neuroscientific evidence Piccinini and Bahar (2013, pp. 476) conclude that neural processing is neither strictly a digital computation nor an analog computation but has characteristics of both. This leaves two choices namely: neural computation is not a computation, or it is a unique hybrid one which has some attributes of both digital and analog computation. Furthermore it is specific to the brain itself.

Piccinini and Bahar argue for the latter view by concluding that:

"*In a nutshell, current evidence indicates that typical neural signals, such as spike trains, are graded like continuous signals but are constituted by discrete functional elements (spikes). Therefore, typical neural signals are neither continuous signals nor strings of digits; neural computation is sui generis.* (2013, pp. 476)

Such a hybrid computation is intended to allow for both continuous variables as well as discrete digits and it therefore has a broader range than digital computation.

Piccinini and Bahar introduce a "generic" concept of computation to cover this hybrid form of neural computation and define it as:

"the processing of vehicles (defined as entities or variables that can change state) in accordance with rules that are sensitive to certain vehicle properties and, specifically, to differences between different portions (i.e., spatiotemporal parts) of the vehicles. A rule in the present sense is just a map from inputs to outputs; it need not be represented within the computing system" Piccinini and Bahar (2013, pp. 458)

One other point they stress as a corollary of this definition is that the processing involves physical properties that are abstracted away to be independent of the medium or substrate on which they are implemented and referred to as "medium independent". Piccinini and Bahar (2013, pp. 458) refer to the work of the neo-mechanists such as Craver (2006) for their main assumptions behind generic computation namely the talk of vehicles and their portions with association rules between inputs and outputs and their fractionating into parts. While Craver does talk about capacities and operations of parts and taxonomy of levels however as my thesis is on Chalmers' theory hence I will not go further into an analysis of the notion of generic computation nor use the notion of generic computation further. I will accommodate hybrid computation by revising Chalmers' model in the next chapter.

3.4 Support for Neural Computation as Hybrid Computation from other Authors

Ratification of a hybrid view of neural processing is also independently provided by Sengupta, Stemmler and Friston (2013, p. 7). Their reasons are different from Piccinini and Bahar (2013) however they argue against neural computation being

analog processing as noise is endemic to analog processing'. In analog processing the noise signal is transmitted with the information signal while with digitisation only the information is transmitted. If neural circuits are analog then noise would be transmitted with the signal instead of being suppressed. The inbuilt threshold mechanisms of spike trains attenuate noise and do not transmit it down the line. However they argue the pre-production processing of the neural signal via neurotransmitters and hormones plus the continuously variable voltages across dendrites is like analog processing thereby making the overall neural computation a hybrid one.

I have discussed a number of views about the nature of neural computation using Piccinini and Bahar (2013) as the main foil for the discussion. The argument presented indicates that neural computation is a hybrid type of computation which has both discrete and continuous properties. In the next chapter I will look at how Chalmers' views can account for hybrid computation or need to be modified.

4. Evaluation of Chalmers' Theory

In this chapter I will discuss how well Chalmers' thesis of cognition and the mind outlined in Chapter 2 stands up to scrutiny in the face of known empirical facts of neuroscience discussed in the previous chapter.

4.1 Chapter Approach

The discussion in the previous chapter on neurons, their activity and functions was done with the aim of asking the following questions in this chapter.

1) Does Chalmers' implementation scheme apply to neural computation?
2) Does Chalmers' thesis of Computational Sufficiency hold in the light of the facts of neuroscience?
3) Does Chalmers' thesis of Computational Explanation hold in the light of the facts of neuroscience?

4.2 Does Chalmers' Implementation Scheme Apply To Neural Computation?

Chalmers' (2011) original paper would be consistent with digital computation described in Section 3.3.1 above as he sets up his CSA as a state based machine. His machine is like a FSA but with vectors for inputs, outputs and internal states. There can be an infinite number of internal states unlike FSA's which can only have a finite number of internal states. The components of the vectors range over discrete parameters such as integers or symbols of an alphabet. He then argues that other computation models such as Turing Machines and cellular automata for example can be represented using CSA. In Section 2.4 of Chapter 2 I argued that his analysis covers both symbolic and sub-symbolic or connectionist architectures. Chalmers does

not use the term digital computation as I have described in Section 3.3.1 above following Piccinini and Bahar (2013) but calls it symbolic computation.

In order to judge how Chalmers' views have to be modified if neural computation is a hybrid type of computation as argued in the previous chapter I consider how a CSA can be set up to handle physical processes with continuous causal organisation. Chalmers' makes some remarks in this direction in his discussion on "Continuous and Discrete" in Chalmers (2011, p. 347) and also in his replies to commentators (Chalmers, 2012, p. 227).

I mentioned above that the CSA has inputs, internal states and outputs that are made up of vectors whose components are discrete values such as integers or symbols from an alphabet. Now if we posit the inputs, internal states and outputs of CSAs as corresponding to real valued quantities then we cover continuous causal processes in the world. Hybrid processes will then be covered by a combination of real valued quantities and discrete quantities and therefore cover combinatorial structures with components having both discrete and continuous values. One other point is that Chalmers' expresses the causation as happening reliably by appeal to counterfactuals to avoid Putnam-Searle type of trivialisations. He posits a spatial separation in the components of the input and state vectors of the CSA keeping head states and tape states of Turing Machines in mind as having different physical locations. For handling continuous structures this will have to be achieved by extending spatial to the spatio-temporal field in order to capture time slices of continuous variables (if need be) in a continuous manner.

Next we consider how the state transition function has to be setup to generate continuous values. There are two options here according to Chalmers. The first is a very close approximation which employs polynomial functions over the real numbers to do a step wise continuous change from state to state with an acceptable level of error. This would work in practice for biological structures and therefore cognitive processes most of the time. This is because approximations can be made as close as possible to a continuous variable (Blum, Shub and Smale 1989). We need to keep in mind that there is noise in biological systems which would make it unlikely that any cognitive process would be so sensitive as to require changes which need tracking to the tenth decimal place. Where it may not work is in the case of chaotic and random signals which can happen in neural circuits in unusual cases *e.g.* stimuli resulting in epileptic seizures (Manganotti, Tamburin, Zanette, and Fiaschi 2001).

The second option is an exact one based on differential equations which generates continuous values (MacLennan 1990). Note that as the state transition function has to cover both discrete and continuous values as far as inputs, internal states and outputs go hence two different sets of rules are required with discrete state transition functions applied to discrete structures and one of the two options discussed above used to handle continuous structures.

The above extension of CSA's will work provided the continuous and discrete structures in the inputs, states and outputs are kept separate. By this I mean that there are two cases. The first case is where there are some tuples or vectors with all discrete components and some with all continuous components and none with mixed components. For example if the input vector *I* has components $[i^1, i^2, \ldots i^n]$ then in the

first case all the components of the vector $[i^1, i^2, \ldots i^n]$ have discrete values for one set of input vectors I_j and have continuous values represented by real numbers for another set of input vectors I_k. In the second case the input vector I has mixed value components $[i^1, i^2, \ldots i^n]$ such that some components have discrete values and some have continuous ones. I have referred to input vectors but the same remarks can apply to the composition of internal state vectors and output vectors of the CSA.

The above extension to Chalmers' CSA model may not work for the mixed case which requires simultaneous modelling of discrete and continuous data in the one tuple. This case cannot be ruled out on an a priori basis and has to be considered[17]. For this case the CSA as a computational paradigm will have to be replaced by one which works at the component level and handles the individual component variables separately instead of combining them within a vector. I don't think the CSA as a tool can handle it as the processing is done at the tuple or vector level and not at the component level of the vector. The only way the mixed case can be handled is to change the paradigm from a CSA processing at a vector or tuple level to a hierarchical computational structure which is setup as a tree with the functional and complex states at the top of the hierarchy and simpler and closer to physical ones lower down until you get a separation of discrete and continuous at a branch and leaf level of the tree. What we are trying to do is break it down to a fine grained level so that processing happens at the component level on the one hand and yet keep the connection at the higher level but hide the detail.

[17] An actual example of this does not come to mind. Colin Klein has suggested one could happen along modulation lines either by attention or by top-down cognitive control to an underlying continuous cognitive process that is enhanced (or inhibited) in a binary fashion.

4.3 Is ASM the Solution?

There maybe a way out for Chalmers if we develop a line of thought advanced by Milkowski (2012, pp 369-371) and endorsed by Chalmers (2012, pp 228) [18]. Milkowski argues that Chalmers use of the CSA is not as effective as the computation model developed using a new class of machines called Abstract State Machines (ASM) which would allow us to model any computation in Cognitive Science. ASMs were developed based on the thesis that one can use them to model any algorithm in any framework to any level of abstraction (Gurevich 1995). In particular any given algorithm can be step-for-step simulated by an appropriate ASM.

The notion of an algorithm in ASM is defined by the following axioms (Milkowski 2012, p 369):

"I. An algorithm determines a sequence of computational states for each valid input.
II. The states of a computational sequence are structures.
III. State transitions in computational sequences are determinable by some fixed, finite description."

A fourth axiom can be added to exclude oracle machines[19] *viz.*

IV. Only undeniably computable operations are available in initial states.

An ASM therefore consists of a sequence of finitely many transition rules of the form

***"if** Condition **then** Updates which transform the abstract states.*

The Condition (also called guard) under which a rule is applied is an arbitrary predicate logic formula without free variables whose interpretation evaluates to true or false. *Updates is a finite set of function updates (containing only variable free terms) of form f (t1, . . . , tn) = t whose execution is to be understood as changing (or*

[18] The other technique which enables simultaneous modelling of discrete and continuous structures in dynamical systems is Time Scale Calculus but use of that is not compliant with a computational framework.
[19] The Church-Turing thesis has been shown to be derived by extending the axioms to cover computable operations in the initial state to avoid non-classical computations (Dershowitz and Gurevich, 2008).

*defining, if there was none).. the occurring functions f at the indicated arguments to the indicated value". (*Borger and Stark, 2003 pp. 28-29).

The rule can be unguarded *i.e.* without the *"If..then"* conditional. The states of an ASM are structures which are domains consisting of sets their members, functions over the members and relations. Without loss of generality one can treat predicates as characteristic functions.

The ASM approach provides a way to describe algorithmic issues in a simple abstract pseudo-code which can be translated into a high level programming language source code in a quite simple manner. ASMs have been extended for algorithms over continuous structures over space and time (Bournez and Derschowitz 2010). Hence they can be used to model hybrid architectures which is what we need to cover neural computation at different levels. ASMs can describe the causal dynamics of systems at different levels of data abstraction. ASMs can work with different data structures unlike Turing Machines or FSAs which work with symbols or numbers alone. Hence they could be used to specify a hierarchy of machines with different levels of data abstraction from very low level ones with detailed states and state transitions specified to higher level ones specifying functions and commands in a higher level language.

For example for a two level hierarchy of machines the top level can be rendered as a state machine whose state transition rules reference the machine at the lower level. As the states are structures i.e. domains of sets their functions and their relations hence they would include the relationships between the different levels of machines. This enables us to give a general account of when low level machines implement high level models.

As ASM states are based on structures whose elements can have functions and relations over the domains they range over hence ASMs can combine both the Turing paradigm of conventional imperative programming with the Church paradigm of functional programming.[20] The higher level ASM could then have declarative command(s) format (akin to the Church paradigm) and its functional descriptions and relations get realised in lower level(s) of state machine(s) (akin to the Turing paradigm) where the states take whatever desired level of abstractness right down to a physical level. Klein (2012) has argued that the Church paradigm maybe more important for understanding the architecture of the brain at higher cognitive levels. According to Klein CSA's had a problem covering this however ASMs have no such problem.

ASMs would be able to give a general account of when a physical system implements different types of computations including ones involving virtual hardware and software states where the original states are being emulated and no longer physically exist. Many computational models have difficulties with working over multi-level structures especially ones which include combinations of levels of virtual hardware and software without direct spatio-temporal or causal relationships *e.g*. consider a PC running Windows7 where the causal structures of the hardware are related to the

[20]The Turing paradigm is captured by the following quote from Klein (2012, pp. 168) "*Computations are specified by specifying state-transition rules. These prescribe, in precise detail, the transitions that an implementing machine must undergo for any particular combination of substates"*. Imperative programming languages are based on the Turing paradigm and programs consist of a sequence of source language instructions which direct a computer how to solve a problem by following a strict sequence of instructions which get compiled into a sequence of machine code *i.e*. the transitions undertaken.
The Church paradigm on the other hand does not tell a computer the strict sequence of instructions *i.e*. how to solve problem. It states the problem that is to be solved. It does this by setting up the functions (mathematical) that are to be evaluated. To quote Klein (2012, pp. 169): *"A program thus describes which functions are to be computed, but neither constrains nor guarantees the order in which functions are evaluated.*" The connection to Church is because these functional programming languages are based on the lambda calculus. See Klein (2012) for a discussion on the Turing and Church paradigms of imperative and functional programming.

underlying formal structures of the computation and then a hierarchy of virtual machines running on this platform emulating various hardware and operating systems *e.g*. a Mac running MAC/OS, which in turn emulates an old IBM 486 PC running DOS and so on with each of these virtual machines running their virtual application software. This is a relatively common occurrence with the inability to run old applications on newer hardware and operating systems A possible example of a virtual machine applying to minds and therefore to neural computation is given by Colin Klein of simple sets of rules communicated in a higher level language like English which "runs" on the brain machine (or mind) like a compiler on a computer and getting translated into a lower level machine code type of language before being actioned. His example is a set of three instructions such as: "First staple these two pages". "Then add them to the folder"; "Then write the delegate's name on them". For processing it requires translating those English instructions presumably by something like compiling them to neural instructions prior to being actioned[21]. As ASMs can be defined by inputs and states from very abstract to physical ones hence virtual states are not a problem plus the link from abstract to concrete physical states can also be defined in the one machine.

Because ASMs can cover functions and relate them to lower level states, they would not have the issues Chalmers' thesis of Computational Explanation had with Marr's and Bayesian theories as discussed earlier in Section 2.5. The functional operators in

[21] Another case is given by Hamburger & Crain (1984) who describe a compile and plan neurolinguistic study on children. It involves carrying out "Do as I say" instructions on words to check if the child has for example correctly understood the referent of a word. Their thesis is that when a chid is asked to show its comprehension on a sentence by an appropriate action, the syntactic structure is first converted into a plan and it is the plan that is actioned. The plan is made up of one or more algorithms and this process of acquisition of the plan is akin to a compilation of a program on a computer. This plan specification and acquisition process could be construed as a case of a hierarchy of virtual machines as the neural representation has no causal structural relationship to the underlying physical mechanism since one does not directly exist.

those theories are not covered well by CSAs as Chalmers had conceded. As mentioned before in this section Gurevich (1995) originally worked on ASM as a general framework covering all algorithms. The proofs he has developed enable ASM to cover all computational formalisms such as Turing machines, FSAs *etc.* In replacing CSA by ASM we no longer have the objections raised in Section 2.8 to CSA as an adequate formalism covering all computational formalisms.

Chalmers (2012, pp 228) has agreed with Milkowski's suggestion to replace CSAs by ASMs and said that that would be the way to go to get a general model of computation which overcame many of the objections raised to his views. Note that the development of the formalism to replace CSAs by ASMs is a non trivial exercise, beyond the scope of this thesis and remains a project for the future. In what follows below I give a brief informal outline.

A definition for a basic ASM can be rendered in the same manner as the one done for FSA with the proviso that the states are abstract structures and the state transition rules can reference other machines as the structure can contain functions and relations:

"A basic ASM is specified by giving a set of input states I_1, ..., I_k, a set of internal states S_1,...,S_m, and a set of output states O_1,...,O_n, along with a set of state-transition relations of the form (S, I) → (S', O'), for each pair (S, I) of internal states and input states, where S' and O' are an internal state and an output state respectively. S and I can be thought of as the "old" internal state and the input at a given time; S' is the "new" internal state, and O' is the output produced at that time."

4. Evaluation of Chalmers' Theory

As mentioned earlier ASMs' states are structures hence they can have functions to any desired level of abstraction. In addition as mentioned before ASMs have systems of finitely many transition rules either with conditional form ***"if*** *Condition* ***then*** *Updates"* or unconditional form which transform the abstract states.

We extend the concept of the implementation by CSA to one by ASM where the internal states, transition rules and outputs replace vectors by structures. In moving from scalars to vectors Chalmers had provided a combinatorial complex structure to the CSA. The move to an ASM deepens the complexity and granularity of the resulting machine with hierarchy and functional command structures as discussed earlier. The other point is that ASMs (like CSAs) differ from FSAs in that their internal states can be either finite or infinite while for a FSA they are always finite. However ASM's inputs can be constrained to be finite by the last axiom above to adhere to the Turing Church hypothesis. For all practical purposes the finite case suffices.

A physical system P implements an ASM, *M*, when the following conditions are met:
"If there are internal states of *P* into structures $[s^1, s^2,\ldots]$, and a mapping f from the sub-states s^j into corresponding sub-states S^j of *M*, along with similar mappings from inputs and outputs, such that for every state-transition rule $([I^1, \ldots, I^k], [S^1, S^2,\ldots]) \rightarrow ([S_o{}^1, S_o{}^2, \ldots], [O^1, \ldots, O^l])$ of *M*: if *P* is in internal state $[s^1, s^2, \ldots]$ and receiving input $[i^1, \ldots, i^n]$ which map to formal state and input $[S^1S^2, \ldots]$ and $[I^1, \ldots, I^k]$ respectively, this reliably causes it to enter an internal state and produce an output that map to $[S_o{}^1, S_o{}^2,\ldots]$ and $[O^1, \ldots, O^l]]$ respectively."

The state transition rules given above are unguarded however they can be set to be conditional with the conditions largely determined by the context (*e.g*. another related machine), the subject matter and the environment to name at least three factors. As FSAs and CSAs are sequential,[22] the ASM description given above is that of a sequential ASM however this need not apply in all cases. ASMs can be defined with parallel, distributed, encapsulated and multi agent formats *(*Borger, E. and Stark. R., 2003, pp. 88-282).

The above ASM formulation can be unpacked as follows. Each internal state of P is a structure given by its components $[s^1, s^2,\ldots]$, and similarly each internal state of the ASM is a structure whose components are $[S^1, S^2,\ldots]$. A structure consists of an ordered set of sub-states or components for each internal state of the physical system and the ASM plus their functions and relations. For example each component of the structure corresponding to an internal state of P could represent a spatio-temporal coordinate. The inputs and outputs of the physical system have structures *e.g*. $[i^1, \ldots, i^n]$ and $[o^1, \ldots, o^l]$ respectively. The inputs and outputs of the ASM also have structures *e.g*. $[I^1, \ldots, I^k]$ and $[O^1, \ldots, O^l]$ respectively.

There is a function f that does a one to one mapping from each sub-state s_j of the internal state, of the physical system P to each sub-state S_j of the internal state of the ASM, M. along with mappings of the components of the inputs and outputs of P to components of inputs and outputs of M. This mapping is based on the state transition rule of M $([I^1, \ldots, I^k], [S^1, S^2,\ldots]) \rightarrow ([S_o{}^1, S_o{}^2, \ldots], [O^1, \ldots, O^l])$. The state transition rule specifies a unique mapping for a combination of input state structures and internal states of M giving new internal state structure plus an output structure of M.

[22] Klein (2012) covers this point in detail in his discussion of the Turing paradigm mentioned earlier.

Furthermore the state transition rule must be such that the causal transition in the physical system P is reliable *i.e*. counterfactual supporting as discussed earlier in connection with reliable causation for FSA implementation.

The above description gives an account of what it is for a physical system to implement a computation and how to interpret the mirroring relationship between the states of a physical system and the states of an abstract computation. In this section I looked at how Chalmers CSA model can be modified if neural computation is a hybrid form of computation. I found that while Chalmers model can be extended to handle continuous causal processes however in the hybrid case it can only handle one of the two possible cases. Only a move to an ASM model can handle both cases.

4.4 If Neural Processes Are Not Computations

At the start of Chapter 3 before investigating the nature of neural computation I set out the four possible logical alternatives. One of these was what if neural processes are not a computation. If that were true then clearly Chalmers theory would have to be rejected. However we have found at the end of the discussion in Chapter 3 that there is sufficient evidence that neural computation is a hybrid type of computation. Hence this alternative has to be discounted and would merit no further discussion about overall theories of the mind and cognition. No doubt as discussed earlier in Chapter 2 Section 2.5 there are theories in specific areas like Marr's model of vision and the Bayesian model of perception which did not fit in well with Chalmers' explanatory computational framework. These were based on mathematical functions and not a computational framework. However Chalmers (2012, pp 243-4, 246) has convincingly argued that they are not general theories of the mind and cognition but

cover particular areas only and the links to his model could be worked out if a move were made to ASMs.

4.5 Does Chalmers' Thesis of Computational Sufficiency and Computational Explanation Hold If Neural Computation is Hybrid?

In the assessment of neuroscientific data that I discussed in Chapter 3 I had concluded that neural computation is neither a digital nor an analog computation but a hybrid type of computation. This would potentially pose a problem for Chalmers' CSA model which is a digital one. This was overcome in Section 4.2 above by modifying CSAs to handle both discrete and continuous structures using real numbers with suitable changes to the state transition function. The question I wish to consider here is how Chalmers' thesis of computational sufficiency which links computation to cognition gets affected (if at all) by neural computation being hybrid. We have seen in Section 2.5 that this thesis states that the right kind of computations is sufficient to have a mind. The straight forward point is that the extensions to CSA in Section 4.2 above make it possible to cover cognitive processes which are both discrete and continuous. If the brain has a massively parallel and massively distributed architecture and its circuits to carry out a function are like artificial neural networks then there is a separation between the task of individual neurons which are like a switch passing encoded / decoded data and the overall processing by the network. The individual neurons are mainly responsible for low level computation while networks of neurons are responsible for the high level computation of cognitive processes. The work of the network results in meaning or action on the data to deliver the function of the network of neurons. In general a cognitive process is not at the level of an individual neuron but at the level of a network of neurons and hence debating about the nature of

processing at the neuron level (whether analog or digital or hybrid) maybe too low a level when we are giving an account of cognition. Chalmers (2011 p. 337) says that his account applies to a "neural level or higher depending on just how the brain's cognitive mechanisms function" provided it results in the causation of behaviour. Only by reading this as meaning at a neural network level and by "higher" as meaning an abstract causal organisational level does one avoid confusion. Hence there is no impact on the thesis of computational sufficiency.

To recall as discussed in Section 2.5, Chalmers' thesis of computational explanation says that computation as construed by him provides an explanatory framework for cognitive processes and behaviour. This thesis is not affected by neural computation at the neuron level being hybrid as hybrid computation was shown to be compliant with an extension to Chalmers' computational model, both the extended CSA one or the ASM model.

5. Conclusion

In this thesis I have examined David Chalmers' causal theory of the mind. I have outlined the theory and then discussed the strengths and weaknesses of his approach. Chalmers developed his theory by first defining an abstract computational object namely a CSA and its implementation via an isomorphism (or mirroring) with the causal processes of the physical system in which the computation is realised. Next he brings out the connection between computation and cognition via the theses of computational sufficiency and computational explanation. In Section 2.7 I go on to show how Chalmers' argument does not fall afoul of the Putnam-Searle trivialisation objection by appealing to counterfactual conditionals. Next I discussed some key issues and objections made by his critics and concluded that while there maybe some difficulties, however no knockdown argument was advanced by any of his detractors on the one hand while Chalmers on the other hand in most cases made robust responses to objections. Furthermore I argued in Chapter 4 that a move away from CSA to ASM as hinted by Chalmers (2012) overcomes the main difficulties with CSA as a computational model and as an explanatory framework for cognition.

By looking at neural computation in Chapter 3 I concluded that the evidence suggests that it is of a hybrid variety covering both discrete and continuous structures causing a potential problem for Chalmers' theory. Again by replacing CSA by ASM I have proposed how Chalmers' thesis can be extended to comply with the hybrid nature of neural computation. By doing this I have shown that it is still a live theory in the philosophy of mind.

References

Blum, L., Shub, M. and Smale, S. 1989. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions, and universal machines. *Bulletin (New Series) of the American Mathematical Society* 21(1): 1-46.

Borger, E. and Stark. R. 2003. *Abstract state machines, A method for high-level system design and analysis*. Springer-Verlag, Berlin.

Bournez, O. and Dershowitz, N. 2010. Foundations of analog algorithms. *Proceedings of the Third International Workshop on Physics and Computation (P&C), Nile River, Egypt*, pp. 85–94. Available at http://nachum.org/papers/Analog.pdf

Byrne, J. 2013. Introduction to neurons and neuronal networks. *Neuroscience Online,* University of Texas Medical School, Houston, Texas. http://nba.uth.tmc.edu/neuroscience/s1/introduction.html

Chalmers, D.J. 2011. A computational foundation for the study of cognition. *Journal of Cognitive Science,* 12(4): 1-21.
,
Chalmers, D.J. 2012. The varieties of computing: A reply. *Journal of Cognitive Science***,** 13: 211-228.

Church, A. 1936. 'A Note on the Entscheidungsproblem'. *Journal of Symbolic Logic*, 1: 40-41.

Craver, C. 2006. When mechanistic models explain. *Synthese*, 153(3): 355-376

Dayan, P. and Abbott, L. F. 2001. *Theoretical neuroscience, computational and mathematical modelling of neural systems.* MIT Press, USA.

Denning, P., Dennis, J. and Qualitz, J. 1978. *Machines, languages and computation.* Prentice-Hall, NJ.

Dershowitz, N., and Gurevich, Y. 2008. A natural axiomatization of computability and proof of Church's Thesis. *The Bulletin of Symbolic Logic*, 14(3): 299-350.

Dreyfus, H. L. 1972. What computers can't do. New York: Harper & Row.

Egan, F. 2012. Metaphysics and computational cognitive science: Let's not let the tail wag the dog. *Journal of Cognitive Science*. 13: 39-49.

Fodor, J.A. 1975. *The language of thought*. New York: Thomas Crowell.

Fodor, J.A. and Pylyshyn, Z. W. 1988. Connectionism and cognitive architecture: A critical analysis. Haugeland J. (Ed), *Mind design II*, MIT Press, 1997.

References

Garson, J. "Connectionism", *The Stanford Encyclopaedia of Philosophy* (Spring 2015 Edition), Edward N. Zalta (ed.), http://plato.stanford.edu/archives/spr2015/entries/connectionism/

Gurevich, Y. 1995. Evolving algebras 1993: Lipari guide. In E. Börger (Ed.) *Specification and validation methods* (pp. 231-243). Oxford: Oxford University Press.

Hamburger, H. and Crain, S. 1984. Acquisition of cognitive compiling. *Cognition*, 17: 85-136.

HIPR 2000. Hypermedia image processing reference. Department of Artificial Intelligence, University of Edinburgh. http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm

Hopcroft, J.E., Ullman, J.D. 1979. *Introduction to automata theory, languages and computation*. Addison-Wesley, Reading, Mass. USA.

Kandel, E., Schwartz J.H., Jessell, T.M. (eds.) 2000. *Principles of neural science.* McGraw-Hill, New York, USA.

Klein, C., 2012. Two Paradigms for individuating implementation. *Journal of Cognitive Science*. 13: 167-179.

MacLennan, B. 1990. Field computation: A theoretical framework for massively parallel analog computation, Parts I-IV. *Technical Report CS-90-100. Computer Science Department, University of Tennessee*.

Manganotti, P., Tamburin, S., Zanette, G., Fiaschi, A. 2001. Hyperexcitable cortical responses in progressive myoclonic epilepsy: a TMS study. *Neurology*. 2001 Nov 27; 57(10): 1793-9.

Marr, D. 1982. *Vision*, New York: Freeman Press USA.

McCulloch, W. and Pitts, W. 1943. A logical calculus of ideas immanent in nervous activity. *Bulletin Of Mathematical Biophysics*, 5: 115–133.

Miłkowski, M. 2011. Beyond formal structure: a mechanistic perspective on computation and implementation. *Journal of Cognitive Science,* 12: 359-379.

Milkowski M. 2015. *Internet Encyclopaedia of Philosophy* (IEP), ISSN 2161-0002 http://www.iep.utm.edu/compmind

Penrose, R. 1990. Precis of the emperor's new mind. *Behavioral and Brain Sciences* 13: 643-655.

Piccinini, G. 2010. The mind as neural software. Understanding functionalism, computationalism, and computational functionalism. *Philosophy and Phenomenological Research*, 81:(2) 269-311.

Piccinini, G. and Bahar, S. 2013. Neural computation and the computational theory of cognition, *Cognitive Science*, 34: 453–488.

Place, U. T. 1956. 'Is consciousness a brain process? *British Journal of Psychology*, 47: 44–50.

Putnam, H. 1960. "Minds and machines." in *Dimensions of Mind, ed. Sidney Hook,* New York University Press, USA.

Putnam H. 1967. The nature of mental states. in *Mind and Cognition: An Anthology, 2nd edn. eds. W Lycan,* 1999. Malden, Ma, Blackwell.

Putnam, H. 1988. Representation and reality. The MIT Press, Cambridge, USA.

Rescorla, M. 2012. How to integrate representation into computational modeling, and why we should. *Journal of Cognitive Science*, 13: 1-38.

Ritchie, J.B. 2011. Chalmers on implementation and computational sufficiency. *Journal of Cognitive Science*, 12: 401-417.

Rummelhart, D., McClelland, J.L. and the PDP Research Group. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models*, Cambridge, MA: MIT Press, USA.

Rummelhart, D. 1998. The architecture of the mind, a connectionist approach. *Mind readings*, 207-238.

Searle, J.R. 1980. Minds, brains and programs. *Behavioral and Brain Sciences* 3: 417-457.

Searle, J.R. 1992. *The rediscovery of the mind.* Cambridge, Mass. MIT Press, USA.

Sengupta, B., Stemmler M.B. and Friston K.J. 2013. Information and efficiency in the nervous system— A Synthesis. *PLOS Computational Biology,* 9: (7) 1-12.

Shadlen, M. N., and Newsome, W. T. 1998. The variable discharge of cortical neurons: Implications for connectivity, computation and information coding. *Journal of Neuroscience*, 18: 3870–3896

Smart, J.J.C. 1959. Sensations and brain processes. *Philosophical Review*, 68: 141–156.

Smolensky, P. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems, Artificial Intelligence 46: 159-216

Sprevak, M. 2012. Three challenges to Chalmers on computational implementation. *Journal of Cognitive Science*, Volume 13: 107-143.

Terman, F.E. (1960) *Electronic and radio engineering*. McGraw-Hill, India.

Towl, B. 2011. Home, pause, or break: a critique of Chalmers on implementation. *Journal of Cognitive Science*, 12: 419-433

Turing, A.M. 1936. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2* 42: 230-65.

Von Neumann, J. 1945. "First Draft of a Report on the EDVAC," Contract No. W-670-ORD-4926, Between the United States Army Ordnance Department and the University of Pennsylvania Moore School of Electrical Engineering, University of Pennsylvania. June 30, 1945.