“Words Lie in our Way”

(Position Paper for Symposium, “What is Computing?”)

Bruce J. MacLennan

Computer Science Department
University of Tennessee, Knoxville
maclennan@cs.utk.edu

June 20, 1994

Abstract

The central claim of computationalism is generally taken to be that the brain is a computer, and that any computer implementing the appropriate program would ipso facto have a mind. In this paper I argue for the following propositions: (1) The central claim of computationalism is not about computers, a concept too imprecise for a scientific claim of this sort, but is about physical calculi (instantiated discrete formal systems). (2) In matters of formality, interpretability, and so forth, analog computation and digital computation are not essentially different, and so arguments such as Searle's hold or not as well for one as for the other. (3) Whether or not a biological system (such as the brain) is computational is a scientific matter of fact. (4) A substantive scientific question for cognitive science is whether cognition is better modeled by discrete representations or by continuous representations. (5) Cognitive science and AI need a theoretical construct that is the continuous analog of a calculus. The discussion of these propositions will illuminate several terminology traps, in which it's all too easy to become ensnared.

> *Words lie in our way!* Whenever the ancients set down a word, they believed they had made a discovery. How different the truth of the matter was! — They had come across a problem; and while they supposed it to have been solved, they actually had obstructed its solution. — Now in all knowledge one stumbles over rock-solid eternalized words, and would sooner break a leg than a word in doing so.
>
> — Nietzsche (The Dawn, 47)

1 Introduction

The central claim of *computationalism* is that the brain is a computer, and that any computer implementing the appropriate program would ipso facto have a mind. In order to evaluate this claim we must be clear about what is meant by "computer," and that, I take it, is the principal purpose of this symposium. Related to this primary issue are questions of the role of analog computation in computationalism, Searle's Chinese Room Argument and symbol grounding.

In this paper I will argue for the following propositions:

1. The central claim of computationalism is not about computers, a concept too imprecise for a scientific claim of this sort, but is about *physical calculi* (instantiated discrete formal systems).

2. In matters of formality, interpretability, and so forth, analog computation and digital computation are not essentially different, and so arguments such as Searle's hold or not as well for one as for the other.

3. Whether or not a biological system (such as the brain) is computational is a scientific matter of fact.

4. A substantive scientific question for cognitive science is whether cognition is better modeled by discrete representations or continuous representations.

5. Cognitive science and AI need a theoretical construct that is the continuous analog of a calculus.

The discussion of these propositions will illuminate several terminology traps, in which it's all too easy to become ensnared.

2 Analog vs. Digital

2.1 History

Harnad has used terms such as "analog processing," "analog sensory projection," "analog input," "analog system" and even "analog world," and has claimed that

Searle's Chinese Room Argument applies to digital computers but not to analog computers (e.g. Harnad 1990, in press-a, in press-b).[1]

Therefore we must begin by asking what "analog" means in these contexts. On one hand, the term "analog" suggests that there is some special relationship (an analogy) between that state of an analog device and the system it's modeling; on the other hand, in most people's minds the terms "analog" and "digital" are synonymous with "continuous" and "discrete."[2] To avoid confusion it will be necessary to disentangle these two senses of "analog," and this requires a historical digression.[3]

The analog vs. digital dichotomy was already well-established in the U.S. by 1946 (*OED Suppl.*, s.v. analogue). In 1948 Stibitz contrasted the two as follows:

> The ordinary desk calculator is a digital machine. If the numbers are not broken down into digits, but are represented by physical quantities proportional to them the computer is called 'analogue'. A slide rule is an 'analogue' device. (*OEDS*, loc. cit.)

Thus, in a simple analog computer the representing variable (the variable in the computer model) is proportional to the represented variable (the variable in the modeled system); so the analogy is obvious (it is literally $\tau\grave{o}\ \grave{\alpha}\nu\acute{\alpha}\lambda o\gamma o\nu$ = a proportion).[4]

In traditional applications of analog computing, the represented variables were continuous physical quantities, and so the analog computer usually made use of continuous quantities (states of the analog device, such as voltages or currents) to represent them. In digital computers, in contrast, the discrete states of the computer and the values of the modeled variable are not related by a simple proportion. We can see how analog/digital came to be identified with continuous/discrete.[5]

2.2 Similarities and Differences

In spite of this history, there is no longer any reason to suppose that the analog/digital distinction consists in the fact that analog computation is based on an "analogy"

[1] Although I think Searle also has digital computers in mind, I do not think his arguments depend in any essential way on digital computation, and I will present later (section 5.2) a version of the Chinese Room for analog computers.

[2] Consider "analog" and "digital" watches. People sometimes claim that the term "analog watch" alludes to an analogy between the motion of the hands and the rotation of the earth, but this is absurd, for there is nothing in the motion of the earth to correspond to the separate hour and minute hands. The hands on a watch go round because they use gears, not because they help us compute the motion of the earth. An orrery, in contrast, is an explicit analogy to the solar system.

[3] These issues were much better understood when analog computation was more familiar; see for example von Neumann (1956, 1958).

[4] Even for the slide rule — everyone's favorite example of an analog computer — the relation is more complex than a simple proportion, since length on the slide rule is proportional to the logarithm of the number.

[5] In spite of its puerile style, Truitt & Rogers (1960, Ch. 2) has an enlightening discussion of analog computers and other analog devices, and of their relation to the continuous/discrete distinction. The reader is referred to it for further explanation.

between two physical processes, whereas digital is not. Although the "analogy" is less apparent in digital computation, it is nonetheless there (Lewis 1971, p. 323). In particular, the relation in digital computation between the values of a model variable and the values of a system variable is not a simple proportion, but there is still a one-to-one relation (defined by a binary positional notation). Therefore, in both analog and digital computation there is an isomorphism (a one-to-one structure-preserving map) between the representing and the represented and so, in this sense, an analogy.[6]

In summary, a formal correspondence (an "analogy") between two systems is central to both kinds of computation, because both are based on a formal structure that underlies two systems, one in the computer, the other the system of interest.

The conclusion we should draw is that the difference between analog and digital computers consists in whether the representation is continuous or discrete. The terms "continuous computation" and "discrete computation" would be more accurate, but history has given us "analog computation" and "digital computation," and so I will stick with them.

2.3 Complementarity Principle

This continuous/discrete distinction brings us to the edge of a terminology trap, so let me try to show the way around it. Digital computers are perhaps our best examples of discrete dynamical systems, yet according to the laws of physics, which are differential equations, we know that the state of a digital computer must change continuously. There can be no instantaneous voltage changes, for example. On the other hand, although we think analog computers manipulate continuous quantities, a closer look shows that the charge on a capacitor, for example, is an integral number of electrons. Unless, of course, we choose to look closer yet, and take account of the wave nature of the electron.

The key point is that the continuity or discreteness of these basic physical processes is irrelevant to the operation of a computer under normal circumstances. The analog computer operates with charges and currents sufficiently large that the discreteness of electrons can be ignored, and the state change of a transistor in a digital computer is rapid enough to be considered instantaneous. What we care about is — not the ultimate nature of matter — but how the devices behave at the relevant scale of observation.

Whether electrons behave more like waves or more like particles surely can't matter to the business at hand: developing a decent model of cognition. (Or if it does matter, then that is a claim requiring empirical justification.) So let's avoid the

[6] The analogy is most apparent in that most literal kind of digital computation: counting on your fingers. It may be objected that a one-to-one relation is impossible in the case where a digital computer is modeling a continuous system, since the computer provides only a finite set of values to correspond to a continuum in the modeled system. However, a closer look at analog computers shows that they too have finite precision. To be precise we would have to say there is a homomorphism (a many-to-one structure-preserving map) from the modeled system to the modeling system.

terminology trap of worrying whether cognition (or computation for that matter) is *really* continuous or *really* discrete; what matters is: which sort of model is better, continuous or discrete, at the appropriate level of analysis?[7] (See also MacLennan 1990c.)

Elsewhere (MacLennan in press-c, in press-d) I've proposed a methodological principle aimed at avoiding this trap. It is called the Complementarity Principle, and states that continuous and discrete models should be complementary, that is, an approximately-discrete continuous model should make the same macroscopic predictions as a discrete model, and conversely an approximately-continuous discrete model should make the same macroscopic predictions as a continuous model.[8]

3 Continuous and Discrete Computation

3.1 Computation in Brief

Having considered the similarities and differences between analog and digital computation, I will now consider more abstractly the process of computation, whether analog or digital, with the aim of elucidating concepts such as formality, syntax, representation, transduction and interpretation.

In its most basic sense, computation is the process of mechanically transforming mathematical entities.[9] (Think of long division.)

However, since mathematical entities do not exist in the physical world, they cannot be directly transformed by a mechanical process. Instead we must manipulate physical surrogates, specifically, concretes that correspond to the mathematical abstractions. Since the concretes are surrogates for the abstractions, we pay attention to only those physical characteristics of the process that correspond to the properties of the abstractions; all their other physical characteristics are irrelevant at best, and noise or error at worst. Choice of physical instantiation is an engineering decision, not a matter of principle, so long as the concrete system instantiates the abstract; the "multiple realizability" of computations is ultimately a consequence on the multiple

[7]It is precisely this terminology trap that leads Lewis (1971) to reject Goodman's (1968, Ch. IV) explication of the analog/digital distinction, which is quite accurate, and to seek a more complex criterion grounded in "primitiveness" or "almost primitiveness" relative to some language of physics. Haugeland (1981) points out that the analog/digital distinction is essentially an *engineering* distinction.

[8]In the absence of complementarity, you have a theory that makes different macroscopic predictions depending on whether physical reality is, in mathematically absolute terms, discrete or continuous, a possibility which is unlikely, though not impossible. Elsewhere I call this the Nobel Prize argument, because if you have such a theory, you can (in principle) set up the experiments, establish the *absolute* nature of reality (not just a better approximation to it), which would be an unprecedented scientific accomplishment that would undoubtedly earn a Nobel prize.

[9]This in no way limits computation to numerical data, since the strings, sequences, sets, trees, etc. manipulated by nonnumerical programs (such as AI programs) are also mathematical entities.

instantiability of mathematical entities. In computation, the substance (matter and energy) is merely a carrier for the form (mathematical structure). Given this overview I'll turn to a more detailed analysis of computational systems.

3.2 Formal Systems

3.2.1 State Space.

A computation is characterized at each point in time by its *state*, which corresponds to the collective state of all the devices comprised by the computer's memory. The only difference between digital and analog computation is that for analog computation the state space is practically discrete, whereas for analog computation the state space is, for practical purposes, a continuum. The state space is the basic representational resource of a computer.

3.2.2 Process.

The actual computation is a sequence of states, which may be completely determined by the initial state, or determined in part by variables external to the computer (i.e., input). The principal difference between analog and digital computation is that digital computation is viewed as a discrete-time process, with the successive states forming a discrete series, whereas analog computation is viewed as a continuous-time process, with the states forming a continuous trajectory in state space. In general terms, the state space provides the "substance" for a computation, upon which a "form" is imposed by the path through state space.

3.2.3 Autonomy.

Computational systems may differ in their *autonomy*, that is, in the degree to which the system behaves on its own, as opposed to the computation being "driven" by some outside agent. In the most autonomous case the process and the initial conditions are both fixed, so that once the computation is started it proceeds to completion independently of external causes; an example would be a program to compute the square root of 2. A less autonomous system fixes the process but allows the initial conditions to be determined externally; an example is a program to compute the square root of a given number. Less autonomous yet is a system that is sensitive to external input throughout its execution; examples include any interactive program (in digital computation) and any feedback control system (in analog computation).

In mathematical terms, the future state $S(t')$ of a computational process is a function of the current state $S(t)$ and the current input $X(t)$ to the process, $S(t') = F[t, S(t), X(t)]$.[10]

[10]For discrete-time processes, $S(t')$ represents the state after the next discrete state transition, $S(t+1)$; for continuous-time processes, $S(t')$ represents the state at the next "instant," $S(t+\mathrm{d}t)$. I am

In other words, S comprises the dependent variables and X comprises the independent variables. The three classes of computational autonomy are then defined by F; in the first case F is independent of X; in the second case F depends on X only at $t = 0$; in the third case F may depend on X at any time. There is also a class of processes that depend on X but not on S; they are completely reactive, with no memory.

It is worth remarking that axiomatic systems, such as studied in formal logic, unlike conventional programs, do not define a unique trajectory. Rather, they define constraints (the inference rules) on allowable trajectories, but the trajectory over which a given computation actually proceeds (i.e. the proof generated) is determined by an outside agent (e.g. a human).[11]

3.2.4 Concrete Realization.

In mathematical terms, a concrete realization of an abstract computation must have all the structure of the abstract process. Of course, it will also have more structure, since mathematics can be no more than a finite abstraction from the infinite concreteness of reality. In mathematical terms, a physical system is a concrete realization of an abstract computation when there is a homomorphism (a many-one, structure-preserving map) from the concrete process to the abstract process.[12]

3.2.5 Program.

The topology of the computational process (i.e., discrete path or continuous path) determines the form of the programs used to express it. Analog computations are described by *differential* equations, which specify the continuous change of the dependent variables. Digital computations are described by *difference* equations, which describe discrete, incremental changes to the dependent variables.[13]

---

ignoring here the possibility of nondeterministic computations, which are of course very important, especially from a theoretical viewpoint, and have a role in analog as well as digital computation (e.g., Rogers & Connolly 1960, Ch. 7). Nondeterministic processes permit certain bifurcations in the computation path that are noncausal (at the relevant level of analysis). Note also that, as usual, the state includes all the system's "memory," and so determines the extent to which prior states can influence present behavior.

[11] Computations defined by axiomatic systems are often taken to be nondeterministic (as defined in the previous footnote), rather than partially determined by an external agent. The present view is more accurate, because we are usually interested in the theorem that is proved, and because the guidance of the computation is a critical part of realistic proof-generation processes.

[12] Specifically, there is a map $\mathcal{H}$ from the concrete to the abstract system such that if (1) t is time; and (2) s is a concrete state, x is a concrete input, and f is the concrete state transition operator; and further (3) F is the abstract state transition operator; then (4) $\mathcal{H}\{f(t,s,x)\} = F(t, \mathcal{H}\{s\}, \mathcal{H}\{x\})$. In practice, imprecision and other physical limitations cause most realizations to be imperfect.

[13] The distinction between (infinitesimal) differential equations and (finite) difference equations is familiar enough from numerical programming, but applies equally well to nonnumerical programming. Indeed, digital computer programs are just generalized difference equations (MacLennan

3.2.6 Programmability.

In a special-purpose computer (whether digital or analog), the equations are fixed by the structure of the computer; in a general-purpose or programmable computer the equations can be changed relatively easily, for example with a plug-board or by loading a program into memory. In a stored-program computer, the program is represented in the computer's state space (which in the case of an analog computer implies that the equations are represented in a continuous "language," an idea which has not been systematically explored).[14]

3.2.7 Universality.

Theoretical consideration of general-purpose computers leads to the question of the existence of *universal machines*, that is, whether there are computers that can be programmed to simulate any other computer. For the digital computer, this question is answered by the Universal Turing Machine and its many equivalents. There is as yet no corresponding theory of universal analog machines, though various notions of universality have been proposed and are being explored along with the related computability questions (e.g., Blum 1989; Blum & al. 1988; Franklin & Garzon 1990; Garzon & Franklin 1989, 1990; MacLennan 1987, 1990d, in press-a, in press-b; Pour-El & Richards 1979, 1981, 1982; Stannett 1990; Wolpert & MacLennan submitted).

3.2.8 Formality.

Each variable in the equations may represent a physical quantity or a pure number. In conventional terms, each variable has an associated *dimension*.[15] If all the variables are pure numbers, so that none of them refer to physical quantities, then the system is completely *formal*, since its behavior is determined by the structure of the equations themselves and not by any specific physics. Such a system can be called "syntactic" because its behavior depends on the formal interrelations among the variables rather than on any specific physical interpretation of the variables (their "semantics").

To the extent that the variables do refer to specific physical quantities, the equations are *material*, rather than formal, since they refer to a specific physical instantiation. In this case we can divide the equations into two sets: the *formal equations*, which contain no physical variables, and the *material equations*, which do. The formal equations specify an implementation-independent computation, whereas the material equations specify an implementation-specific *transduction*. Intuitively, the formal

1989, 1990b, pp. 81, 193). Nonnumerical analog computer programs are probably best treated as differential equations over Banach spaces.

[14] Such continuous languages might be used to describe "second-order analog" systems (Haugeland 1981). See MacLennan (in press-a, in press-b, in-press-f) for some steps in this direction.

[15] This discussion is simplest if put in terms of numerical computation, but applies equally well to nonnumerical, in which case the question is whether a variable refers to an abstract mathematical object or to a physical *quality*.

equations are the program, the material equations are the input/output relations to the real world.

I must stress that even the formal variables and equations may have an interpretation (physical or otherwise) associated with them, as when they represent logical propositions and inference rules (respectively), but they are formal because the interpretation is not a necessary part of their definition. Different physical instantiations would work as well, and that, of course, is why we can build computers. The material equations, in contrast, are by definition bound to a specific physical implementation, and thus are grounded directly with the laws of physics.[16]

3.2.9 Calculus vs. Simulacrum.

The idea of a *calculus* has been understood since antiquity, although its full significance was discovered only in this century through the work of Church, Post, Turing, Gödel and others. Originally, "calculus" meant a pebble, but it is also applied to counting tokens, game pieces, voting tokens, and so forth (*OLD*, s.v.).[17] It is a token, a nonspecific "something," the properties of which are unimportant so long as it can be distinguished from other tokens (and so counted etc.).

In its modern sense, "calculus" embodies the idea of formal *digital* computation, and so also formal *logical* inference, and the theory of *digital* computation is in essence the theory of calculi and their properties. Traditional "symbolic" theories of knowledge representation and inference in AI and cognitive science take as a given that knowledge and inference are to be represented as some kind of calculus. The idea of a calculus thus becomes the central unifying principle of these theories.

I've argued elsewhere (MacLennan 1988, in press-a, in press-b, in press-f) that connectionism, which is oriented toward continuous knowledge representation and inference, suffers from the lack of a unifying theoretical construct corresponding to the calculus. In MacLennan (in press-a, in press-b, in press-f) I have proposed the *simulacrum* as a possible theoretical construct to fulfill this role, that is, to be a model of possible forms of *continuous* information representation and processing.[18] I will not go into details here, but note only that corresponding to the *formulas* of calculi, simulacra have *images*, and that transformations of images are required to be continuous.

[16]As will be discussed in more detail later (section 5), Harnad's "symbol grounding" is established by the material equations, which connect the cognitive agent with its environment.

[17]The corresponding Greek word, ψῆφος, appears in the generalized sense by the sixth century BCE (*LSJ*, s.v.), where it can also mean a number, a pebble for divination, a mosaic tile, etc. and, more abstractly, a vote or a judgement.

[18]"Simulacrum" (si-mu-lá-crum) is derived by analogy with "calculus," and means a likeness, image, representation, etc. (*OLD*, s.v.). While the term "image" is intended to include visual and auditory images, it is not limited to these, but can be any element of a topological continuum. Previously I have called these images "continuous symbols," because they are the continuous analog of the usual discrete symbols. However, I have found that the term "symbol" so strongly connotes discreteness, that the phrase "continuous symbol" is more confusing than helpful.

3.2.10 Idealization.

I must stress that the simulacrum is an *idealized* computational system, which means that it assumes the states and processes are perfectly continuous (i.e., mathematically so), just as a calculus assumes that states and processes are perfectly discrete. Such idealization is appropriate for mathematical models, and in accord with the Complementarity Principle. We must remember, however, that physical instantiation of a calculus or simulacrum is rarely perfect, and so in any particular case we must pay attention to whether the idealization is a good enough approximation to the reality.

3.3 Interpretations

3.3.1 Syntax vs. Semantics.

One commonly distinguishes between an *uninterpreted calculus* (a calculus proper) and an *interpreted calculus*. Both are formal computational systems, but in the former case we consider only "syntactic" relations, those internal to the system, whereas in the latter we take into account "semantic" relations, which associate some meaning in an external domain with the states and processes of the calculus, thus making them *representations*.[19]

In exactly the same way we must distinguish uninterpreted and interpreted simulacra. In this case the "syntax" is determined entirely by the internal structure of the continuous states and the internal dynamics of the processes, and in this sense an uninterpreted simulacrum is viewed purely "syntactically." A simulacrum is interpreted, or given a semantics, in much the same way as a calculus: by defining mappings from its images and processes onto the states and processes of some domain of interpretation. In this way it becomes a *continuous representational system.*

Although I've stated these ideas abstractly for the sake of generality, they are really quite familiar. A formal analog computer program is just a set of differential equations (perhaps with boundary conditions); they can be instantiated in one physical system (the computer) and interpreted to tell us about any other system that obeys the same equations. In particular, the variables of the physically instantiated formal system can be interpreted as the variables pertaining to some other system with the same formal structure (i.e. defined by the same differential equations). For example, voltages, currents, and conductances in an electronic analog computer can be interpreted as pressures, flow rates and cross-sectional area in a hydraulic system.

[19]Note that here we are talking about meaning attributed to the system by an external observer (a human interpreter), not intrinsic semantics, that is, meaning inherent in the system. Analogously, we distinguish systems that are intrinsically representational (e.g., presumably, brains) from ones that we may *view as* representations (typical computer programs, formulas in mathematics or logic, etc.).

3.3.2 Systematicity.

What we normally require of discrete representational systems, and what allows them to reduce meaningful processes to syntax, is that the interpretation be systematic, which means that it respects the constituent structure of the states. To find an analogous concept for continuous representational systems we need only look at systematicity more abstractly. Constituent structure merely refers to the algebraic structure of the state space (e.g., as defined by the constructor operations; see, e.g., MacLennan 1990b, Chs. 2, 4). Systematicity then simply says that the interpretation must be a homomorphism: a mapping that respects the algebraic structure (though perhaps losing some of it).[20]

The point is that these ideas are as applicable to continuous representational systems as to the better-known discrete representational systems. In both cases the representations (physical states) are arbitrary so long as the "syntax" (algebraic structure) is preserved.

3.4 Computation and Computational Systems

In the light of the preceding discussion, let me suggest the following definitions:

> *Computation* is the instantiation of a formal process in a physical system to the end that we may exploit or better understand that process.

"Formal," as before, refers to the fact that the process is determined entirely by the mathematical structure of the equations, and does not depend on any particular physical instantiation of their variables. This definition covers both automatic and hand computation, whether digital or analog.

> A *task* is *computational* if its function would be served as well by any system with the same formal structure.

Thus, computing the square root and unifying propositions are computational tasks, but digesting starch is not.

> A *system* is *computational* if it accomplishes a computational task by means of a computation.

> A *computational system* comprises a formal part (e.g., a calculus or simulacrum) and, usually, an interpretation.

[20] Indeed, as explained in more detail in MacLennan (in press-c), systematicity in both the analog and digital cases can be defined as continuity, under the appropriate topology in each case.

A complete interpretation is not necessary, since useful computations may pass through uninterpretable states. This occurs in mathematics, for example, with the use of differential notation in the infinitesimal calculus.[21] A computational system usually instantiates either a calculus or a simulacrum, though hybrid discrete+continuous systems are also possible.

4 Computationalism

4.1 Computation and Computational Systems

Searle (1990) has said, "Computational states are not *discovered within* the physics, they are *assigned* to the physics." That is, a physical system is not *intrinsically* a computer in the same sense that it may be intrinsically a brain, a star, or a rutabaga. Although I agree with Searle in the general case of physical systems, I will argue that we can discover computational systems within *biology*. This is because computational tasks and processes are defined in terms of their function, and we can often establish functions for biological systems. For example, if we could show that the visual cortex would function as well whether it were implemented in neurons, silicon, hydraulics or any other physical realization of some set of equations, then we would have shown that the visual cortex is computational. That is, we would have shown that the visual cortex is operating on mathematical entities *by means of* physical processes. On the other hand, function is much harder to establish for nonbiological natural systems, and so it would be difficult to show that they are intrinsically computational; in this case I agree with Searle.

4.2 Computers

Although I think it makes sense to ask if the brain is a computational system or even if the mind is a computation, I think it is sloppy to ask if the brain is a computer, for in normal usage a computer is a *tool* used, or intended to be used, for computation. Though we can use it in an extended sense to mean a computational system, such usage is likely to lead to more confusion.

4.3 Computationalism

Rather than asking whether the brain is a computer, a better strategy is to formulate the hypothesis of computationalism in terms of the notion of computation and computational systems, which is more susceptible to precise definition. This strategy implies a research agenda:

[21]Strictly speaking, for example, in standard analysis, $\mathrm{d}y = 2x\mathrm{d}x$ is an equation between two uninterpretable formulas, since neither side of the equation stands for a number. The use of divergent series as generating functions is another example. See also MacLennan (1990b, p. 421-6).

1. Characterize the dynamics of the brain in terms of equations or other mathematical relations.

2. Determine if some of these equations are *formal*, that is, independent of their material instantiation.

3. Determine whether differential or difference equations are a better model of the processes (at the relevant level of abstraction), that is, whether the representations and processes are continuous or discrete.

Or, in brief, develop a mathematical model of the brain, determine if it is computational, and, if so, whether it is analog, digital or hybrid.

5 Intentionality

5.1 Mutatis Mutandis

Much has been made of analog computation in connection with Searle's Chinese Room Argument, and it has been claimed that the argument applies only to digital computation (Harnad in press-a, in press-b). Although I think analog computation (in the broad sense) plays a critical role in cognition, I do not think Searle's arguments are any less applicable to it. To argue this, I have to take a somewhat unusual position, which may confuse the reader. I am not convinced by Searle's argument, and I think a version of the "systems reply" is correct (MacLennan in press-e). Nevertheless, my purpose here is to argue only that the argument applies as well, mutatis mutandis, to analog systems as to digital. That is, if you accept it in the digital case, then you must also accept it in the analog case, and conversely if, like me, you do not accept it in the digital case, then neither should you in the analog case.

Before presenting a specific analog version of the Chinese Room, I would like to consider the argument in the terms introduced above. Suppose we characterize the brain in terms of formal equations and material equations. The material equations describe specific physical transduction process, such as the conversion of light energy into nerve impulses, and the conversion of nerve impulses into muscular contractions. The formal equations describe the computational process, which can in principle be instantiated by any physical system obeying the same equations. In particular, Searle himself can, in principle, instantiate the formal equations. Therefore, the argument goes, there must be more to understanding Chinese than just implementing the right formal equations, since if there weren't, Searle could instantiate these equations, yet without the subjective experience of understanding Chinese.[22]

[22]Harnad has correctly observed (in other terms) that a situated intelligence requires material equations as well as formal equations, and that, although Searle can in principle instantiate the formal ones, he cannot instantiate an arbitrary set of material equations.

5.2 The Granny Room

With this background established, I can turn to a presentation of an analog version of the Chinese Room. Since the notion of continuous computation is less familiar than the notion of discrete computation, I'll present the example in some detail. I call it "the Granny Room," because its purpose is to recognize my grandmother and respond, "Hi Granny!" A (continuous) visual image is the input to the room, and a (continuous) auditory image is the output.[23]

Inputs come from (scaleless) moving pointers. Outputs are by twisting knobs, moving sliders, manipulating joysticks, etc. Various analog computational aids – slide rules, nomographs, pantographs, etc. — correspond to the rule book. Information may be read from the input devices and transferred to the computational aids with calipers or similar analog devices. The person in the room (Searle, say) implements the analog computation by performing a complicated, ritualized sensorimotor procedure — the point is that the performance is as mechanical and mindless as symbol manipulation. Picture an expert pilot flying an aircraft simulator. Now, when the system correctly replies "Hi Granny," Searle can honestly claim that he doesn't recognize the woman. Indeed, he may even be unaware that a face has been "seen." Therefore, the argument goes, formal equations are not sufficient for the mental phenomenon of face recognition. In spite of the correct behavioral response, no true recognition was involved.

It may be objected that this argument is not immune to the Systems Reply, since the dials, levers, slide rules, etc. are an essential part of the computation. Searle's answer, in the digital case, is to have the person memorize all the rules, thus internalizing the computation and *becoming* the system, but the same move is possible in the analog case. Instead of memorizing a (vast) number of rules and manipulating them mentally, like a mathematician or calculating prodigy, Searle must instead memorize a incredibly complex continuous process, and carry it out mentally, as might an expert choreographer. (Of course, we're talking principles here, not practicalities.) Mutatis mutandis, the argument applies as well in the continuous case as in the discrete.

It may be objected that even in this case Searle must still see the input and produce the output, and that these are transductions, and so the process is not purely formal. But the same applies in the digital case. Even if Searle memorizes all the rules, there must still be some way to get the input to him and the output from him. If a slip of paper bearing the Chinese characters is passed into the room, then he must look at it before he can apply the memorized rules; similarly he must write down the result and pass it out again. How is this different from him looking at a continuous pattern (say on a slip of paper), and doing all the rest in his head, until he draws the result

[23] I've selected this example because face recognition is a characteristically connectionist task. However, by the Complementarity Principle, I could as easily pick understanding Chinese as the task. The relative discreteness or continuity of the task is not essential.

on another slip of paper? Whatever move is made in the discrete argument, the same can be made in the continuous. The only difference is that *inside Searle's head* the processing will be discrete in one case and continuous in the other, but very little hangs on that difference.

Let's consider this in more abstract terms. To subject a system to the Turing Test, as supposed in Searle's argument, requires that the inputs and outputs be represented physically. Thus at least some of the equations defining the system must be material. A testable system cannot be pure computation; there must be transduction somewhere, the only question is where.

Searle could instantiate the whole process, if it is possible for him to instantiate the material as well as the formal equations. (This would be possible, for example, if the input were in the form of visible light.) In this case, though, Harnad would claim that the system is not purely formal and so, properly speaking, not computational. If, on the other hand, we suppose that Searle instantiates only the formal equations, then the input must be delivered to Searle as formal variables, that is, as quantities whose physical instantiation is irrelevant. Therefore, the material equations must be instantiated elsewhere, and some other device (or person etc.) must convert the physical input into its formal representation. In this case Searle does not instantiate the entire system, and so the Systems Reply can be made (as Harnad has observed).

In summary, we can construct an exact continuous analog of Searle's discrete argument, and so the continuous/discrete (analog/digital) distinction cannot be crucial to the presence or absence of original intentionality.

5.3 Synthetic Ethology

Harnad (1991; Hayes & al in press) has argued for the importance of Searle's Argument on the grounds that it provides a loophole though an otherwise "impenetrable other-minds barrier," and therefore allows us to determine that computational systems could not be intentional. He also argues for a "Total Turing Test" as the only means of determining the intentionality of noncomputational systems (Harnad 1989, 1991, in press-a, in press-b). In taking this essentially behaviorist stance he has given up more than necessary. From a scientific standpoint the other minds barrier is not impenetrable, for psychologists and ethologists routinely determine empirically whether to attribute mental states and intentionality to other organisms. An approach, which is consistent with current scientific practice in neuropsychology and neuroethology, is to ground intrinsic meaningfulness of external signals and internal representations in terms of function, and to cash out function in terms of a tendency to contribute to inclusive fitness (e.g., Burghardt 1970). Though such studies are difficult to conduct in a natural environment, they can be approached through the methods of "synthetic ethology" (MacLennan 1990a, 1992; MacLennan & Burghardt submitted).

6 Conclusions

Although it is commonly thought that analog computation differs from digital in that the former has a special relationship (the "analogy") with its subject matter, in fact both kinds of computation depend on a systematic relationship between two systems, specifically, that physical phenomena in the computer are instances (to sufficient accuracy) of given mathematical relationships, which may also apply to other systems of interest. In fact, the principal difference between analog and digital computation lies in the former having continuous states that change continuously and the latter having discrete states that change discretely. A methodological guideline, the Complementarity Principle, tells us that what matters is the behavior (continuous or discrete) at the relevant level of analysis, not in some ultimate mathematico-physical sense.

Computation refers to the transformation of mathematical entities by means of physical processes having the same formal structure. Thus multiple realization is a necessary characteristic of computation. The concepts *formality* ("syntax"), *interpretation* ("semantics"), *systematicity*, *program*, and *universality* apply as well to analog computation as digital, though the analog domain is not so thoroughly investigated as the digital. The simulacrum has been proposed as a unifying theoretical construct to fulfill a role for analog computation corresponding to that fulfilled by the calculus for digital computation.

A system is computational when its function would be served as well by any other with the same formal structure. This presupposes that the function of a system can be identified, but that is often possible for biological and artificial systems. Thus (contra Searle) it is a legitimate scientific question to ask whether the brain or individual brain systems are computational. A secondary question is whether individual systems are more analog or more digital in their operation. Progress in connectionism is showing the strength of continuous representations.

Although Searle's Chinese Room Argument is presented in the context of discrete (digital) computation, it applies as well, mutatis mutandis, to continuous (analog) computation. As Harnad has noted (in different terms), an often neglected aspect of Searle's thought experiment are the material equations that interface the physical world of the Turing Test to the (purported) formal equations of cognition. Although I agree with Harnad that such "grounding" of representations is necessary for an effective embodied intelligence, I do not agree that it depends in any essential way on the continuous/discrete distinction, or that it is the only way out of Searle's conclusions.

It is likely that continuous representations of high dimension will provide better models for many cognitive processes than the discrete, "symbolic" representations that have been commonly used. However, this is an empirical question relating to continuity or discreteness of cognitive processes at the relevant level of analysis, and does not gain any support from Searle's Chinese Room argument or from the need for symbol grounding. The nature and source of original intentionality is nevertheless a critical question for cognitive science and AI.

7 References

Blum, L. (1989) *Lectures on a theory of computation and complexity over the reals (or an arbitrary ring)* (Report No. TR-89-065). Berkeley, CA: International Computer Science Institute.

Blum, L., Shub, M., & Smale, S. (1988) On a theory of computation and complexity over the real numbers: NP completeness, recursive functions and universal machines. *The Bulletin of the American Mathematical Society* **21**: 1–46.

Burghardt, G. M. (1970) Defining 'communication.' In J. W. Johnston Jr., D. G. Moulton and A. Turk (Eds.), *Communication by chemical signals* (pp. 5–18). New York: Century-Crofts.

Franklin, S., & Garzon, M. (1990) Neural computability. In O. M. Omidvar (Ed.), *Progress in neural networks* (Vol. 1, pp. 127–145). Norwood, NJ: Ablex.

Garzon, M., & Franklin, S. (1989) Neural computability II (extended abstract). In *Proceedings, international joint conference on neural networks* (vol. 1, pp. 631–637). New York, NY: Institute of Electrical and Electronic Engineers.

Goodman, N. (1968) *Languages of art: An approach to a theory of symbols.* Indianapolis, IN & New York, NY: Bobbs-Merrill.

Harnad, S. (1989) Minds, machines and Searle. *Journal of Theoretical and Experimental Artificial Intelligence* **1**: 5–25.

Harnad, S. (1990) The symbol grounding problem. *Physica D* **42**: 335–346.

Harnad, S. (1991) Other bodies, other minds: A machine incarnation of an old philosophical problem. *Minds and Machines* **1**: 43–54.

Harnad, S. (in press-a) Grounding symbols in the analog world. *Think.*

Harnad, S. (in press-b) Artificial life: Synthetic vs. virtual. In *Artificial Life III.*

Haugeland, J. (1981) Analog and analog. *Philosophical Topics* **12**: 213–225.

Hayes, P., Harnad, S., Perlis, D., & Block, N. (in press) Virtual symposium on the virtual mind. *Minds and Machines.*

Lewis, D. (1971) Analog and digital. *Noûs* **5**: 321-327.

MacLennan, B. J. (1987) Technology-independent design of neurocomputers: The universal field computer. In M. Caudill & C. Butler (Eds.), *Proceedings, IEEE first international conference on neural networks* (Vol. 3, pp. 39–49). New York NY: Institute of Electrical and Electronic Engineers.

MacLennan, B. J. (1988) Logic for the new AI. In J. H. Fetzer (Ed.), *Aspects of artificial intelligence* (pp. 163–192). Dordrecht: Kluwer.

MacLennan, B. J. (1989) *The calculus of functional differences and integrals* (Technical Report CS-89-80). Knoxville, TN: Computer Science Department, University of Tennessee.

MacLennan, B. J. (1990a) *Evolution of communication in a population of simple machines* (Technical Report CS-90-99). Knoxville, TN: Computer Science Department, University of Tennessee; submitted for publication.

MacLennan, B. J. (1990b) *Functional programming: Practice and theory.* Reading, MA: Addison-Wesley.

MacLennan, B. J. (1990c) The discomforts of dualism. *Behavioral and Brain Sciences* **13**: 673–674.

MacLennan, B. J. (1990d) *Field computation: A theoretical framework for massively parallel analog computation; Parts I – IV* (Technical Report CS-90-100). Knoxville, TN: University of Tennessee, Computer Science Department.

MacLennan, B. J. (1992) Synthetic ethology: An approach to the study of communication. In C. G. Langton, C. Taylor, J. D. Farmer and S. Rasmussen (Eds.), *Artificial Life II* (pp. 631–658). Redwood City: Addison-Wesley.

MacLennan, B. J. (in press-a) Characteristics of connectionist knowledge representation. *Information Sciences*, to appear.

MacLennan, B. J. (in press-b) Continuous symbol systems: The logic of connectionism. In Daniel S. Levine and Manuel Aparicio IV (Eds.), *Neural networks for knowledge representation and inference.* Hillsdale, NJ: Lawrence Erlbaum.

MacLennan, B. J. (in press-c) Field computation in the brain. *Proceedings, 1st Appalachian conference on behavioral neurodynamics: Processing in biological neural networks.* Washington, DC: International Neural Network Society.

MacLennan, B. J. (in press-d) Information processing in the dendritic net. *Proceedings, 1st Appalachian conference on behavioral neurodynamics: Processing in biological neural networks.* Washington, DC: International Neural Network Society.

MacLennan, B. J. (in press-e) Grounding analog computers. *Think*.

MacLennan, B. J. (in press-f) Image and symbol: Continuous computation and the emergence of the discrete. In V. Honavar & L. Uhr (Eds.), *Artificial intelligence and neural networks: Steps toward principled integration, Volume I: Basic paradigms; learning representational issues; and integrated architectures*. New York, NY: Academic Press.

MacLennan, B. J., & Burghardt, G. M. (submitted) Synthetic ethology and the evolution of cooperative communication.

Pour-El, M. B., & Richards, I. (1979) A computable ordinary differential equation which possesses no computable solution. *Annals of Mathematical Logic* **17**: 61–90.

Pour-El, M. B., & Richards, I. (1981) The wave equation with computable initial data such that its unique solution is not computable. *Advances in Mathematics* **39**: 215–239.

Pour-El, M. B., & Richards, I. (1982) Noncomputability in models of physical phenomena. *International Journal of Theoretical Physics* **21**: 553–555.

Rogers, A. E., & Connolly, T. W. (1960) *Analog computation in engineering design*. New York, NY: McGraw-Hill.

Searle, J. R. (1990) Is the brain a digital computer? Presidential Address. *Proceedings of the American Philosophical Association*.

Stannett, M. (1990) X-machines and the halting problem: Building a super-Turing machine. *Formal Aspects of Computing* **2**: 331–341.

Truitt, T. D., & Rogers, A. E. (1960) *Basics of analog computers*. New York NY: John F. Rider.

von Neumann, J. (1956) The general and logical theory of automata. In J. R. Newman (Ed.), *The world of mathematics* (Vol. 4, pp. 2066–2098). New York, NY: Simon & Schuster.

von Neumann, J. (1958) *The computer and the brain*. New Haven & London: Yale University Press.

Wolpert, D., & MacLennan, B. J. (submitted) A computationally universal field computer which is purely linear.