CHAPTER 1

BODIES — BOTH INFORMED AND TRANSFORMED EMBODIED COMPUTATION AND INFORMATION PROCESSING

Bruce J. MacLennan

Department of Electrical Engineering & Computer Science
University of Tennessee, Knoxville
E-mail: maclennan@utk.edu

Post-Moore's Law computing will require an assimilation between computational processes and their physical realizations, both to achieve greater speeds and densities and to allow computational processes to assemble and control matter at the nanoscale. Therefore, we need to investigate "embodied computing," which addresses the essential interrelationships of information processing and physical processes in the system and its environment in ways that are parallel to those in the theory of embodied cognition. We briefly discuss matters of function and structure, regulation and causation, and the definition of computation. We address both the challenges and opportunities of embodied computation. Analysis is more difficult because physical effects must be included, but information processing may be simplified by dispensing with explicit representations and allowing massively parallel physical processes to process information. Nevertheless, in order to fully exploit embodied computation, we need robust and powerful theoretical tools, but we argue that the theory of Church-Turing computation is not suitable for the task.

1. Post-Moore's Law Computation

Although estimates differ, it is clear that the end of Moore's Law is in sight; there are physical limits to the density of binary logic devices and to their speed of operation. This will require us to approach computation

in new ways, which present significant challenges, but can also broaden and deepen our concept of computation in natural and artificial systems.

In the past there has been a significant difference in scales between computational processes and the physical processes by which they are realized. For example, there are differences in spatial scale: the data with which programs operate (integers, floating point numbers, characters, pointers, etc.) are represented by large numbers of physical devices comprising even larger numbers of particles. Also, there are differences in time scale: elementary computational operations (arithmetic, instruction sequencing, memory access, etc.), are the result of large numbers of state changes at the device level (typically involving a device moving from one saturated state to another). However, increasing the density and speed of computation will force it to take place on a scale (spatial and temporal) near that of the underlying physical processes. With fewer hierarchical levels between computations and their physical realizations, and less time for implementing computational processes, computation will have to become more like the underlying physical processes. That is, post-Moore's Law computing will depend on a greater assimilation of computation to physics.

In discussing the role of physical embodiment in the "grand challenge" of non-classical computing, Stepney writes,

> Computation is physical; it is necessarily embodied in a device whose behaviour is guided by the laws of physics and cannot be completely captured by a closed mathematical model. This fact of embodiment is becoming ever more apparent as we push the bounds of those physical laws. [Stepney, 2004, p. 29]

Traditionally, a sort of Cartesian dualism has reigned in computer science; programs and algorithms have been conceived as idealized mathematical objects; software has been developed, explained, and analyzed independently of hardware; the focus has been on the *formal* rather than the *material*. Post-Moore's Law computing, in contrast, because of its greater assimilation to physics, will be less idealized, less independent of its physical realization. On one hand, this will increase the difficulty of programming since it will be dependent on (or, some might say, contaminated by) physical concerns. On the other hand, as I will argue here, it also presents many opportunities that will contribute to our understanding and application of information processing in the

future. To understand them, I will make a brief digression through non-Cartesian developments in philosophy and cognitive science.

2. Embodied Cognition

Johnson and Rohrer trace the theory of *embodied cognition* to its roots in the pragmatism of James and Dewey, both of whom stressed the importance of understanding cognition as an embodied biological process [Johnson, Rohrer, 2007]. Dewey's *Principle of Continuity* asserts that there is no break from our highest, most abstract cognitive activities, down through our sensory and motor engagement with the physical world, to their foundation in biological and physical processes. Cognition is the emergent pattern of purposeful interactions between the organism and its environment (including other organisms). Psychologists, such as Piaget and Gibson, and philosophers, such as Heidegger, Polanyi, and Merleau-Ponty, have made similar points.

Hubert Dreyfus and others have stressed the importance and benefits of embodiment in cognition. As Dreyfus observed, there are many things that we (implicitly) know simply by virtue of having a body [Dreyfus, 1979, pp. 248–250, 253]. Therefore, in embodied cognition, embodiment is not incidental to cognition (or to information processing), but essential to it. For representative recent work see [Clark, 1997; Pfeifer, Scheier, 1999; Iida, Pfeifer, Steels, Kuniyoshi, 2004; Pfeifer, Bongard, 2007]. Finally, from [Brooks, 1991] onward there has been increasing understanding of the value and exploitation of embodiment in AI and especially in robotics [Iida, et al., 2004].

3. Embodied Computation

Pfeifer, Lungarella, and Iida provide a concise definition of *embodiment*: "the interplay of information and physical processes" [Pfeifer, Lungarella, Iida, 2007, p. 1088]. On this basis we define *embodied computation* as information processing in which the physical realization and the physical environment play an unavoidable and essential role.

3.1. Physics for Computational Purposes

Embodied computing can be understood as a natural consequence of the decreasing size and cost of computing devices. Historically, *offline* computer applications were most common. Interaction with the environment could be characterized as *input—process—output*. That is, physical input (e.g., punched cards, magnetic tape) was presented to the computer and converted into internal, computational representations, and in this effectively abstract form it was processed. As abstract results were generated they were converted into specific physical representations (e.g., printed paper, punched cards, magnetic tape) for use after the program terminated. It is easy to see offline computation as the evaluation of a mathematical function on an argument, which is the way it is treated in the traditional theory of computation.

As computers became smaller and less expensive, it became feasible to embed them as controllers in larger systems. *Embedded computations* are in ongoing interaction with their environments, are typically non-terminating, and have to be of a physical size and real-time speed compatible with the physical systems in which they are embedded. Their basic structure is *sensors—controller—actuators*, in which, however, there are critical real-time feedback loops through the physical environment from the actuators back to the sensors. Nevertheless, the basic model is similar to offline computing in that the sensors and actuators perform the conversions to and from the computational medium, which is effectively abstract (largely independent of specific physical realization). Physical considerations are confined to the embedding device and its environment, the transducers (sensors, actuators), and basic physical characteristics of the control computer (size, weight, electrical requirements, clock rate, memory capacity).

The difference between *embedded computing* and *embodied computing* is that in the latter there is little or no abstract computation; the computation must be understood as a physical system in continuing interaction with other physical systems (its environment). The strength of embodied computing, like the strength of embodied cognition, resides in the fact that information representation is often implicit in the computation's physical realization and in its environment. Representations and information processes emerge as regularities in the

dynamics of the physical systems that allow the computational system to fulfill its function.

Another significant advantage of embodied computing is that many computations are performed "for free" by the physical substrate. For example, diffusion occurs naturally in many fluids, such as liquids and gases, and in other media. It can be used for many computational processes, including broadcasting of information and massively parallel search, such as in path planning through mazes, optimization, and constraint satisfaction [Khatib, 1986; Miller, Roysam, Smith, O'Sullivan, 1991; Rimon, Koditschek, 1989; Steinbeck, Tóth, Showalter, 1995; Ting, Iltis, 1994]. Diffusion is expensive to implement by conventional computation, but it comes for free in many physical systems.

As is well known, many artificial neural networks are based on matrix-vector multiplications combined with simple nonlinear functions, such as the *logistic sigmoid*, $1/[1+\exp(-x)]$. Also, many universal approximation theorems are based on linear combinations of sigmoids and similar functions [Haykin, 1999, pp. 208–94]. Computing a sigmoid on a conventional computer requires computing a series approximation to a transcendental function (e.g., exp, tanh) or approximating the sigmoid by table look-up and linear interpolation. However, sigmoidal behavior is typical of many physical systems, for it results from an exponential growth process that gradually saturates. For example, available chemical receptors may become occupied or the supply of signaling molecules may become exhausted. In general, sigmoidal response comes for free because physical resources become saturated or depleted. In embodied computing we do not need to program sigmoid functions explicitly; we can exploit common physical processes with the required behavior.

Further, many self-organizing systems depend on positive feedback for growth and extension and on negative feedback for stabilization, delimitation, separation, and the creation of structure (in space or time). In embodied computation negative feedback may be implemented by naturally occurring physical processes such as evaporation, dispersion, and degradation of chemicals. These processes will occur anyway; embodied computation makes productive use of them.

One final example must suffice. Many algorithms, such as simulated annealing [Kirkpatrick, Gelatt, Vecchi, 1983] and stochastic resonance [Benzi, Parisi, Sutera, Vulpiani, 1982], use randomness for productive purposes, including escape from local optima, symmetry breaking,

deadlock avoidance, exploration, etc. Such randomness comes for free in physical systems in the form of noise, uncertainty, imprecision, and other stochastic phenomena.

In summary, with conventional computing technology we often "torture" the physical substrate so that it implements desired computations (e.g., using continuous electronic processes to implement binary logic), whereas embodied computation "respects the medium," conforming to physical characteristics rather than working against them.[a] The goal in embodied computation is *to exploit the physics, not to circumvent it* (which is costly).

3.2. Computation for Physical Effect

We have seen how embodied computation exploits physical processes for the sake of information processing, but embodied computation also uses information processing to govern physical processes. That is, typically we think of computation as a physical system in which the physical states and processes represent (perhaps imperfectly) certain abstract states and processes, which constitute a desired information system. In mathematical terms, there is a (perhaps imperfect) homomorphism from the concrete physical system onto the abstract information system [MacLennan, 1994 a; MacLennan, 2004]. But we can look at computation from a different perspective, since an information system (and, in a general-purpose computer, the program) governs the flow of matter and energy in the physical computer (subject, of course, to the computer's structure and the laws of physics). This is in fact an essential function in natural embodied computation (including embodied cognition), which governs physical processes (e.g., growth, metabolism) in an organism's body and its physical interactions with other organisms and their environment. Often, the result of embodied computation is not information, but *action*, and even self-action, self-transformation, and self-construction.

When our purpose is information processing, then the goal is often to represent the information with as small a quantity of energy or matter

[a] The metaphors of "torturing" and "respecting the medium" were suggested to me by Christof Teuscher and Peter Dittrich, respectively.

(e.g., electrical charge) as possible — consistent with reliable operation — so that state changes will require as small a movement of energy or matter as possible, for the sake of minimizing state-transition time and heat dissipation. Indeed, the (unattainable) goal has been a sort of *disembodied* computation and communication, in which pure form is represented, transmitted, and transformed without need of material realization. On the other hand, when embodied computation is applied to the control of matter and energy, we may want to move *more* rather than *less*. This is because, in contrast to conventional embedded computers, in embodied computation there may be no clear distinction between the processors and the actuators; the physical effects may be a direct consequence of the computational process (as opposed to being *controlled* by them). Therefore embodied computation may involve the movement of relatively large amount of matter or energy compared to traditional computation, such as large molecules, large electrical quantities, etc. For example, in *algorithmic assembly* DNA computation is used to assemble nanostructures [e.g., Barish, Rothemund, Winfree, 2005; Rothemund, Papadakis, Winfree, 2004; Rothemund Winfree, 2000; Winfree, 1998], and our own work explores the use molecular computation based on combinator reduction for nanostructure synthesis and control [MacLennan, 2003 a].

Further, embodied computation can be applied to the implementation of active materials, that is, materials that have a complex behavioral repertoire. Thus, embodied computation might be used to implement an artificial tissue that can recognize environmental conditions and open or close channels in response to them, or otherwise transport matter or energy across the membrane, perhaps transforming it in the process. Embodied computation might be used to implement a material, analogous to cardiac tissue, capable initiating and controlling organized patterns of contraction.

Much current nanotechnology has a materials orientation, by which I mean that it is most successful at producing bulk materials with a desired nanostructure or microstructure; to create macroscopic structure we must resort to more traditional manufacturing methods. Yet morphogenesis and pattern formation in embryological development show us that embodied computational processes can coordinate the proliferation, movement, and disassembly of cells, macromolecules, and smaller molecules to produce highly complex systems with elaborate hierarchical structure from the nanoscale up to the macroscale. This is an inspiring

model for future nanotechnology: using embodied computation to control the multistage self-organization of complex, functional, and active hierarchical systems, that is, *artificial morphogenesis* [MacLennan, 2009 a, in press].

4. Related Work

Several authors have discussed embodied computation and related concepts. There is not space here for a complete review (which, in any case, would be premature at this point in time), so I will limit myself to a few similarities and differences.

According to Hamann and Wörn an *embodied computation system* consists of at least two levels, with adaptive self-organization and collective behavior at the higher levels resulting from spatially local interactions among "microscopic control devices," which are embodied devices comprising sensors, actuators, a processor, and memory [Hamann, Wörn, 2007]. There are several aspects to their embodiment, including a lack of separation between processor and memory and an essential dependence of the computation on the physical world (e.g., spatial position). They name simple robots, cells, and molecules as examples of microscopic control devices. Their definition has much in common with our own, but seems to conflate embodiment with issues of adaptation, self-organization, robustness, loose coupling, etc., which are related to embodiment, but not essential to it.

Stepney discusses the ideas of *material computation* and *in materio* computers, that is, systems in which the physical substrate "naturally" computes [Stepney, 2008]. These concepts are very similar to embodied computation as I have presented it, with perhaps two differences. First, she advises that we focus on non-living substrates in order to understand material computation, since biological systems are so much more complicated. Second, it appears that she is primarily concerned with the use of physical materials to implement computations, and less concerned with the use of computational processes to organize and control matter and energy. (Indeed, this difference is suggested by the terms *material* computation and *embodied* computation, since the latter connotes a self-organizing, self-regulating body and an organicist approach.) Stepney considers a variety of physical media that might be used for computation. She also cautions us against an ill-advised application to material

computation of notions from Turing computation, a topic that I address later (Secs. 9–10).

5. Computation and Information Processing

An obvious question is whether embodied computation is a kind of computation at all, that is, whether it is appropriate to apply the terms *computation* and *information processing* to these physical processes. A brief consideration of the usage of these terms may prove helpful.

We may begin with *computation*, which — like *calculation* — originally referred to a human activity. Calculation was an embodied human activity performed primarily with concrete objects (Lat., *calculi* = pebbles). Computation was more abstract (Lat., *computare*, referring to the reckoning of accounts, etc.), but even *putare* (to think over, reckon, etc.) has a primary sense of tidying up physically [*OLD*, s.vv. computo, puto]. As is well known, before the middle of the twentieth century, a *computer* was a human occupation, which was later extended to certain artificial systems, *automatic computers* [*OED*]. More recently the term has been transferred back to natural systems, and we use *natural computation* to refer to computation occurring in nature or inspired by it. What are the features common to human and non-human natural computation and to artificial computers? I will address this later.

The ordinary sense of *information* is derived from the verb *to inform*, which means to give *form* to something or to oneself [*OED*, s.vv.]. That is, it is the shape or configuration that is relevantly altered, not its substance or material. Like computation, the activity of *informing* (another or oneself) and the abstraction *information* refer originally to human activities. By extension they are naturally applied to non-human animals. In both cases the effect of information is to reshape the pattern of (internal or external) activity of an organism or group of organisms. Also like computing, the ideas of information and information processing have been transferred to artifacts. Shannon (and before him, Hartley) facilitated this extension in part by taking human relevance and meaning out of the definition of information.

In order to expand our concepts of embodied computation and information processing it is especially fruitful to understand natural (non-technological) information processing systems. In addition to teaching us information processing paradigms distinct from binary digital logic, they can show us how to exploit embodiment for more effective information

processing. This raises a demarcation issue: how do we discriminate information processing (and computation) from other natural processes? To decide, we must look to the function (or purpose) of the process, a topic to which I now turn.

6. Function and Structure

6.1. Function

Function, purpose, and other teleological notions are problematic in science, but I will argue that they are largely unproblematic in the context of computation and information processing.

On one hand, the function of an artifact is generally easy to determine: ask the designer. That is, artifacts are designed for a purpose, which is often explicitly stated or easy to determine in a contemporary or historical cultural context. The function of some specific historical and prehistoric artifacts may be unclear, but that fact does not invalidate the general principle.

Similarly, although teleological notions, such as purpose and function, are problematic in a biological context, and are ultimately grounded in inclusive fitness, they are unproblematic in the context of particular biological subsystems. For example, biologists routinely and objectively investigate and describe the functions of the digestive, reproductive, immune, and nervous systems. Therefore, we can, in principle at least, establish criteria for whether or not the function of a biological system is information processing and computation, or something else.

However, in biology, but also in technology, structures and processes may serve multiple functions. Although such combinations of function may interfere with our understanding of the system, they improve system efficiency. (In an engineering context it is called the *Shanley Principle*; see Sec. 9).

I believe that the fundamental criterion distinguishing information processing from other physical processes is that only the abstract form is relevant to the purpose; it could as well be realized by other physical systems. This is the root of the *mutual realizability* commonly taken to be characteristic of computation. Furthermore, information, as formalized by Shannon, depends on contrasts and distinctions, that is, on

form, and is independent of the physical substrate supporting the contrasts.

In order to explore the essence of information processing, especially in the context of embodiment, it will be convenient to use the Aristotelian notions of *form* (Grk. *eidos*, Lat. *forma*) and *matter* (Grk. *hulê*, Lat. *materia*) which correspond, in more modern terms, to organization and energy. That is, it is the abstract shape or structure that is relevant to information processing, and its physical substrate is relevant only insofar as its suitability to support the form.

Computation is a physical process and therefore it takes place in a physical medium, a substratum supporting its formal structures and their transformations. For computation qua computation, the specific properties of the medium are relevant only insofar as they support the formal organization and process that constitutes the computation or information process.[b] If we abstract away from these irrelevant specific properties, we are left with a kind of generic matter, or neutral substrate, not infinitely malleable or indeterminate, but able to support the formal structures and processes that fulfill the computational system's purpose. For example, any modern computer architecture can be understood as a (relatively) neutral medium (i.e., an array of bytes and certain primitive operations on them) that can support a wide range of formal information processes (i.e., programs).

In the context of the form-matter dichotomy it is natural to think of matter as something fundamentally formless and simple, but form and matter are relative to each other and to an appropriate level of system analysis. In fact computational media are often quite complex. For example, while the concept of a bit is simple, a modern digital computer architecture is quite complex. Embodied computation, in particular, often makes use of complex matter, which exhibits a wide repertoire of complicated and interrelated, but not necessarily functional, behaviors. Computation recruits, organizes, and coordinates these complex behaviors to achieve the computation's purpose. Examples of such complex embodied computational media include neurons in the nervous system, social insects in colonies, cells in morphogenesis, and proteins in intracellular regulation [Tokuriki, Tawfik, 2009]. Many computational

[b] Obviously, the specific instantiation may be relevant to non-computational issues, such as physical robustness, energy requirements, physical compatibility with the rest of the system (e.g., a biological computational medium in a living organism, an electronic computational medium in a robot).

media belong to Wolfram's dynamical class IV, which resides on the border between static and periodic behavior (classes I, II) on one hand and chaotic behavior (class III) on the other [Wolfram, 2002]. As Wolfram has stressed, rich, complex behavior can emerge from very simple mechanisms [Wolfram, 2002, ch. 12]. This insight is summarized in Stuart Kauffman's slogan, "order for free."

6.2. Regulation and Causation

A principal goal of natural information processing is *regulation* of other physical processes to some end. That is, information is extracted from the larger physical system, processed formally, and used to control or influence physical processes. This includes the regulation of other systems (i.e., in the environment of the computational system), but especially self-regulation (homeostasis and development). Because computation is a formal process, and essentially independent of physical magnitude, it can regulate physical processes that involve more matter or energy than the computation itself.

Regulation is for some purpose or end, and therefore it is always future-directed. Its goal is either to maintain a current state into the future or to alter the current state in the future in order to pursue some goal. As a consequence, artificial computation, like natural computation, is functional; it is directed toward some *end* (in the sense of purpose, not final state; cf. Aris., *Phys*. 194b32–33). Teleology is unavoidable in computation, whether natural or artificial.

As a consequence, all four of the Aristotelian primary "causes" (Grk. *aitia*, Lat. *causa*) can be applied profitably to descriptions and explanations of computation and information processing, especially in nature (on the four causes, see Aris., *Phys*. II 194b–195a, *Met*. 983a–b, 1013a–1014a).[c]

The *formal explanation* appeals to "the form [*eidos*] or pattern [*paradeigma*]; that is, the essential formula [*logos*] and the classes that contain it" (Aris., *Met*. 1013a26–28). The form is correlative to the matter in the physical state of the computational system, and it is the

[c] The conventional translation "cause" does capture the ancient terms' ranges of meaning, which include responsibility, motive, occasion, theme, category; i.e., explanation, answer to "how? and why?" (LSJ, OLD).

form that governs the computational process, which is understood as a process of *transformation* (change of form). In some cases, such as morphogenesis and algorithmic self-assembly, the end and goal of transformation is the creation of some final form (structure, organization).

The *material explanation* accounts for computation in terms of its *matter*, that is the computational medium, whether specific to a particular realization, or generic for a class of computational systems. The medium must be able to support the formal structures and processes of the computation and be suited to the system's purpose.[d] Aristotle (*Phys*. 194b9) observes that "the conception of 'material' [*hulê*] is relative, for it is different material that is suited to receive the several forms." The material is the *substratum of form and its transformation*. The material aspects assume a much larger importance in embodied computation than they do in traditional computation.

The *efficient explanation* appeals to the agent or mover (Grk. *kinoun*, Lat. *efficiens*, *movens*), that is, "to something to initiate the process of change or its cessation when the process is completed" (Aris., *Phys*. 194b29–31). This is both the initial *informing agent* (which physically imposes the initial formal state on the medium), and the *transforming agent* of the initiation and completion of each successive step of computation (each step imposing new form on the medium). Computational systems are dissipative physical systems and their state changes require energy. A few computational processes can be initialized in a nonequilibrium state and allowed to compute to equilibrium, but most computational processes must be fueled or powered so long as they continue. Energy issues, both its provision and its dissipation, are more important in embodied computation than they were in conventional computation.

The *final explanation* focuses on the function, purpose, end, or completion (Grk. *telos*, Lat. *finis*) of a process, and indeed the formal, material, and efficient causes together constitute the *means* to achieve the *end*, which is the final cause. The final explanation is relevant to artifacts, which serve our ends, as well as to organisms, which serve their own. In particular, we have seen that purpose is fundamental to the

[d] "In the crafts, then, it is we that prepare the material for the sake of the function it is to fulfill, but in natural products Nature herself has provided the material. In both cases the material is commanded by the end to which it is directed." (Aris., *Phys*. 194b8–10, tr. Wicksteed & Cornford).

definition of natural and artificial computational systems, since their function is information processing, as manifest in multiple realizability.

In summary, all four of Aristotle's "causes" or kinds of explanation are relevant to computation and information processing, but in embodied and natural computation the material, efficient, and final aspects play a more important role than they do in traditional computation (for which the formal aspect dominates).

6.3. Structure

Conventionally we think of the physical realization existing for the sake of realizing an abstract computation. That is, we have some system of forms and transformations (an information process) in which we are interested, and we arrange (by construction or programming) the physical process to instantiate the abstract process of interest. However, as we have seen, sometimes the function of a computational process is regulation, that is, its purpose is to *inform* (impose a form on) matter, or to organize energy.

This is apparent in natural computation systems, where information processing is often devoted to the maintenance of an organism, colony, or species. For example, information-mediated regulatory processes control tissue growth and repair, embryological development and morphogenesis, social organization, and colony construction and maintenance. It is significant that in many of these cases the information process is modifying (transforming) its own physical realization or, to put it in other terms, the computation is recomputing its physical implementation. Embryological morphogenesis is a clear example, since information processes regulate the development of the embryo, and the structure of the embryo reciprocally governs the information process. Similarly social insects coordinate their behavior to construct a physical colony, which in turn conditions the collective behavior of the insects.

This sort of mutual determination of information processes and physical processes will be increasingly important in artificial systems as well, since we can use automatic information processing to fabricate or manufacture systems that are too small or too intricate for traditional techniques. For example, self-repairing or self-healing artificial systems, like natural systems, may use intrinsic information processes to detect damage, recruit repair resources, and coordinate (re-)construction. In

various self-assembly processes, such as artificial morphogenesis [MacLennan, 2009 a, in press], elementary physical components, with primitive information-processing capacities, organize into a physical substrate for further, more coordinated behavior, which creates more complex physical structures that in turn structure more complex information processes. Finally, radically reconfigurable computers and reconfigurable robot collectives can adapt themselves to changing requirements by disassembling themselves into neutral physical components and then reconfiguring themselves into a new physical structure, in a process analogous to the metamorphosis of a caterpillar into a butterfly. Thus information processes can reorganize matter and energy, including the physical substrate realizing the information processes, which is perhaps the best example of embodied computing.

7. A Mathematical Model of Embodied Computation

In this section I will provide a more precise, mathematical characterization of the difference between information processing and other physical processes. The key criterion, as previously discussed, is multiple realizability in a teleological context.

Before developing a mathematical model it is essential to recall that each model is suited to answering a certain class of questions, and therefore exists in a *frame of relevance* [MacLennan, 2003 b; MacLennan, 2009 b], determined by the model's intended use, and that in conformity to its frame, a model exists at a corresponding level of description, and that it incorporates factors relevant to these questions and excludes the rest for the sake of simplicity.

7.1. States and Trajectories

Therefore we will consider the set S of states for a closed system (which I'll also call S). Since computation is relative to a purpose, of an artifact, in an organism, in a colony, etc., we will decompose S into two subsystems, A an agent involving information processing, and E, its environment. In terms of the state space, $S = E \times A$.

However, A is typically only partially computational, and so as a first approximation, we might factor the agent into a body B and a computational part C, that is, $A = B \times C$. The computational part is

devoted to information processing, as will be explicated shortly. This decomposition is suitable for conventional computation, which is internal to the agent, but in embodied computation the information is partially externalized to the environment. Therefore it is more accurate to factor the complete system into a physical part P and a computational part C, that is, $S = P \times C$. The computational state C has exterior and interior parts, C_{E} and C_{A}, which reside in the environment and agent respectively: $C = C_{\mathrm{E}} \times C_{\mathrm{A}}$.

An additional complication that arises in embodied computation is that the state spaces might not be fixed through time. Think of embryological morphogenesis; as cells proliferate, the state space of the embryo increases in dimension (degrees of freedom), including the computational state space, which increases to accommodate the information processing of the proliferating cells. Conversely, in embryological development *apoptosis* (programmed cell death) decreases the dimension of both computational and non-computational state spaces.

The easiest way to treat this possibility is by defining the state spaces (S, C, P, etc.) to be large enough to accommodate the highest dimension required. An increasing effective state space then corresponds to a state trajectory rising out of a lower dimensional subspace into higher dimensions. Conversely, a contracting state space corresponds to the trajectory confining itself to progressively lower dimensional subspaces.

Since we are concerned with information processes, which govern behavior, we must consider trajectories in state space over some defined time interval $T = [t_{\mathrm{o}}, t_{\mathrm{f}}]$. Thus we may consider the state $s(t) \in S$ at time $t \in T$, and its computational component $c(t) \in C$.

However, at the appropriate level of modeling the trajectories may be nondeterministic, and therefore each trajectory has a probability distribution over states at a given time. The probabilities of the trajectories themselves are conditional on their initial states and also, for open systems, on their boundary conditions, but for simplicity we restrict the presentation here to closed systems. Therefore we interpret a trajectory as a function $\tau : S \times T \times S \to [0,1]$ so that $\tau(s_0, t, \sigma) = \Pr\{s(t) = \sigma \mid s(t_0) = s_0\}$ is the probability that the trajectory beginning at s_0 is in state σ at time t. Of course probabilities are required to be normalized, $\int_S \tau(s,t,\sigma)\,\mathrm{d}\mu(\sigma) = 1$ for all $s \in S$ and

$t \in T$, where a measure μ appropriate to the state space is chosen. In specific cases there will of course be other restrictions on the trajectories, but that is not relevant to our present discussion.

7.2. Multiple Realizability

With this background we can formulate the property of multiple realizability, which is fundamental to information processing. The idea we want to express is that all physical systems realizing the same abstract computational process will generate the same distribution of trajectories, but it is not so simple as this, because the computational state is part of the total state, and so any change of its realization will change the state space.

A certain subspace I of C constitutes the interface between computational system and the physical system P. Thus we write $C = I \times H$, where H is the "hidden" (non-interface) subspace of the computational state. Thus, I represents the physical inputs and outputs of the computational system, which must have a specific physical representation in order for the computation to fulfill its purpose. For example, its purpose may be to detect a gradient in the concentration of some particular chemical and move in the corresponding direction. In more conventional terms, the relation between I and H is that between the input/output transducers and the rest of the computational system.

Therefore consider a potentially alternative realization of C with physical state space $C' = I \times H'$. We are concerned with the trajectories generated by computations in C and C' on the visible state space $V = P \times I$. Each trajectory τ in S generates a projected visible trajectory $\upsilon : V \times T \times V \to [0,1]$ defined by:

$$\upsilon(v_0,t,v) = \int_H \int_H \tau\left[(v_0,h_0),t,(v,h)\right] \mathrm{d}\,\mu(h) \cdot \Pr\{\mathrm{h}_0 \mid v_0\} \mathrm{d}\,\mu(h_0),$$

where v_0, $v \in V$ and $\Pr\{h_0 \mid v_0\}$ is the probability of initial state $s_0 = (v_0,h_0)$ given v_0; they are the initial states h_0 of the computation consistent with observable state v_0.

Then, if we have two computational realizations C and C', the condition for their realizing the same abstract computation (and hence serving the same purpose) is the equality of their visible projections: $\upsilon = \upsilon'$. This defines an equivalence relation on physical realizations of

computations, and the corresponding equivalence classes correspond to abstract computations.

We have described how the same abstract computation can be realized in different physical state spaces C, C', C'', … There are also abstract computational state spaces C^*, in which, in effect, all the components are dimensionless numbers as opposed to real physical quantities. All physical realizations C of this computation generate the same observable behavior as C^*, that is, $\upsilon = \upsilon^*$.

The foregoing are necessary conditions for multiple realizability, but what are the sufficient conditions? To establish them, we need to know something about the structure of the computation. To illustrate, we will restrict our attention to a system that can be specified by a simple differential or difference equation, $s(t') = F\big[s(t)\big]$, where t' represents the next "instant" of time: $t + \Delta t$ in the case of a difference equation, $t + \mathrm{d}t$ in the case of a differential equation.[e] The initial state is $s(t_0) = s_0$. Next, we separate the non-computational from the computational components of the system:

$$v(t') = F_{\mathrm{V}}\big[v(t), h(t)\big],$$
$$h(t') = F_{\mathrm{H}}\big[v(t), h(t)\big].$$

F_{V} incorporates output transduction and F_{H} incorporates input transduction. Similarly, the abstract computation is described:

$$v(t') = F^*_{\mathrm{V}}\big[v(t), h^*(t)\big],$$
$$h^*(t') = F^*_{\mathrm{H}}\big[v(t), h^*(t)\big].$$

A sufficient condition for the physical system to realize the abstract computation is that there exists a mapping $r : H \to H^*$ satisfying the following homomorphism conditions [cf., MacLennan, 1994 a; MacLennan, 2004]:

$$F_{\mathrm{V}} = F^*_{\mathrm{V}} \circ \big(i \times r\big),$$
$$r \circ F_{\mathrm{H}} = F^*_{\mathrm{H}} \circ \big(i \times r\big).$$

[e] It is straight-forward to put these differential equations into standard form.

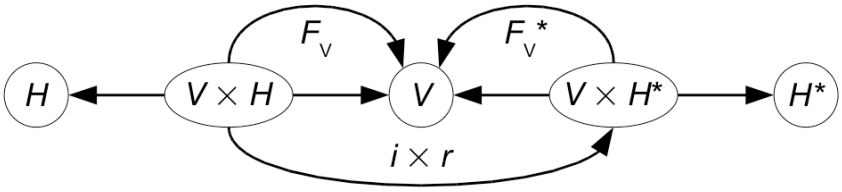

Fig. 1. Commutativity diagram for visible state.

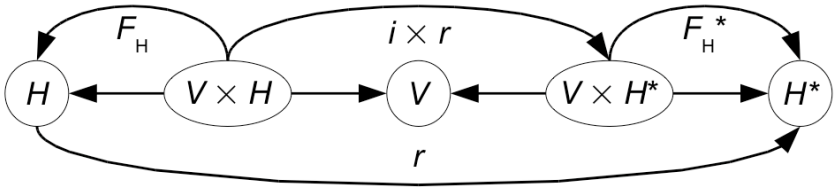

Fig. 2. Commutativity diagram for hidden state.

where $i : V \to V$ is an identity function; see the commutativity diagrams (Figs. 1 and 2). For the initial state the condition is $h_0^* = r(h_0)$.

When stated in such abstract terms it is easy to mistake multiple realizability for a mathematical property, but it is not; it is physical property or, more properly, a property of physical systems in a teleological context. Therefore, it is helpful to consider an example.

Ants lay down pheromone trails when they return to their nests with food [Camazine, Deneubourg, Franks, Sneyd, Theraulaz, Bonabeau, 2001]. In addition to showing the way to the food source, these trails convey other information, such as the quality of the food source. The competition between reinforcement of the trail as the ants use it, and its disappearance as the pheromone dissipates and degrades, ensures that the path structure is adaptive and efficient in guiding the ants to and from their food sources. The primarily computational function of this system is evident in the fact that the pheromone could be replaced by other physical substances that would work as well. The transducers would have to be replaced appropriately; that is, the ants would have to be able to produce and detect the new signal substance. Of course, to fulfill its function adequately, the new substance would have to have the same

rates of increase and decrease (although it is likely that any substance with the same ratio of increase rate to decrease rate would work as well). The ready availability of such a substance and the difficulty of ants producing or detecting it are not the important issues. What is important is that we can see that the functioning of the process does not depend in an essential way on any specific substance. The process is essentially formal not physical; it is information processing.

We may contrast this with a non-computational process that has some superficial similarities: the transport of food to the nest. If we think of it in purely mathematical terms it might seem to be multiply realizable. We can imagine a sort of input transducer that converts a portion of physical food into a number (encoding the food's quality and quantity in some way). This number is conveyed by a formal process (perhaps modeling the movement of the ants to the nest), where an output transducer converts the encoding into corresponding physical food.

Certainly, such a system could be constructed, but we can see that it would not fulfill the same purpose as the original transport process, which was to convey energy (stored in specific substances) from one physical location to another. In order to actually implement the alternative realization described, it would be necessary for the output transducer to be able to take the input signal (encoding the amount and kind of food) and use it to guide the synthesis of physical food that could be used by real ants. Naturally, this output transducer would consume energy and raw materials to fuel this synthesis, which would defeat the purpose of the transport process, which was to bring resources to the nest. Since the purpose of the transport process is to convey real, physical matter and energy from one place to another, we can see that a formal process is not suitable.

Therefore, when we are considering a supposed alternative realization, we must do so in the context of physical reality, not mathematical structure. It is an alternative realization only it fulfills the same physical function as the original system. There will, of course, be borderline cases, and we know that some systems may not be purely computational, but such complications do not invalidate the general concept. Categories may be useful even though neither nature nor engineering is compelled to conform exactly to them.

8. Design of Embodied Computation Systems

One of the challenges of embodied computation is that we have very little experience doing it. Much of our programming has been done in the idealized worlds of perfect logic and implementation-independent programming languages; unavoidable interactions with physical reality have been relegated to the periphery. Fortunately nature provides numerous examples of effective embodied computation, from intracellular genetic regulatory circuitry to the swarm intelligence of social insects and other animals. Therefore we can look to nature to learn how computation can cooperate with physics, rather than opposing it, and how information processing systems can fruitfully interact with the physical embodiment of themselves and other systems.

Since embodied computation is a new computing paradigm, it may be worthwhile to say a few words about how embodied computation systems might be designed. The first step is to *understand* how information processing occurs, and interacts with physical reality, in natural systems. We may benefit both from studies of specific systems relevant to some application of interest, but also from more general information about embodied computation in nature [e.g., Camazine, et al., 2001].

The second step is to *abstract* the process, so far as possible, from the specifics of its physical realization. In practical terms, this often amounts to developing a mathematical model of the relevant aspects of the system (i.e., the embodied information processing). This might seem like a return to disembodied, abstract models of computation, but it is not, for it incorporates physical processes in their essential form. For example, a natural system might exploit the diffusion and degradation of some pheromone, but its mathematical description would be in terms of the diffusion and degradation of *some* substance (with appropriate relative rate constants). That is, once we understand the computational principles, a *specific* quantity can be replaced by a *generic* quantity. Of course, some natural embodied computational systems will be more dependent on specific realizations (e.g., particular physical quantities) than others, and the more generically realizable ones will be the more generally useful to us.

The last step in developing an embodied computation system is to *realize* the abstract computational principles in an appropriate medium by selecting substances, forms of energy, quantities, and processes

conformable to the mathematical model and the purposes of the system. This, of course, is more difficult than the disembodied computing with which we are familiar, but it will be necessary to master these techniques as we enter the post-Moore's Law era and attempt to apply computing principles more widely.

In the end, the process of designing an embodied computation system is not so different from designing a conventional computation system. The designer develops an abstract dynamical organization that will exhibit the required interactions with its environment. This is analogous to programming, the principal difference being that embodied computation makes use of different primitive processes and representations, namely those that have comparatively direct physical realizations. As a consequence, the physical environment and the physical realization of the computation will never be far from the designer's mind.

By looking at embodied computation in nature we may begin to isolate computational primitives that are generally useful and realizable in a variety of media. Because of its importance, I will focus here on embodied computation in morphogenesis (the self-organized development and metamorphosis of hierarchical form). Although there is some overlap and ambiguity, we may distinguish those primitives that pertain to the individual elements of the system and those that pertain to masses of them.

An embodied computation system, especially one organizing morphogenesis, will comprise a very large number of elementary units, such as cells or molecules. In the first case we are interested in physical processes involving single elements, which may respond passively or actively. Examples of such *individual primitives* include mobility (translation, rotation), adhesion and release, shape change, differentiation or state change, collision and interaction, and proliferation and apoptosis (programmed cell death, unit disassembly). Other processes pertain more to spatially distributed masses of elementary units, and they may be called *collective primitives*. Examples include elasticity, diffusion, degradation, fluid flow, and gradient ascent.

Finally, biological morphogenesis teaches us that embodied computation can orchestrate and organize complex, multistage processes operating in parallel at both the microscopic and macroscopic levels. For example, Bonabeau, Dorigo, and Theraulaz, in their investigations of swarm intelligence in wasp nest construction, recognized the concept of

a *coordinated algorithm*, which leads to an organized nest structure [Bonabeau, Dorigo, Theraulaz, 1999]. Similarly, we need to discover how to design coordinated algorithms for embodied computation in artificial morphogenesis and similarly complex applications.

9. But Is It Computing?

The reader may allow that embodied computing, as described above, is interesting and potentially useful, but object to considering it a species of computing. After all, we have a precise definition of computation in the Turing machine and its equivalents (according to the notion of equivalence defined in Church-Turing computation theory). On the other hand, the notion of embodied computing may seem imprecise and difficult to discriminate from other physical processes.

If we consider "computation" and related terms, both in historical usage (which includes "analog computation") and in the context of contemporary discussions in philosophy and computer science, we can describe computation as *a physical process, the purpose or function of which is the formal manipulation (processing) of formal objects* [MacLennan, 1994 a; MacLennan, 2004]. As we have seen, a physical process may be considered computation (or information processing) if its purpose could be fulfilled as well by another physical system with the same abstract (e.g., mathematical) structure. In short, its purpose is *formal* rather than *material*.

This definition might seem to exclude embodied computation, or make it an oxymoron, but I do not think this is so, for there is nothing contradictory about embodied computation's greater reliance on physical processes for information processing. However, embodied computation may be directed also at the production of specific material effects; that is, its purpose may be physical rather than formal.

There are two answers to this. First, embodied computation's physical effects can often be understood abstractly (i.e., mathematically). For example, an activator-inhibitor system will produce characteristic Turing patterns, which can be characterized mathematically, independently of specific substances involved [Turing, 1952]. Second, we cannot expect all physical systems to fit neatly into categories such as *computational* and *non-computational*, but we should expect there will

be degrees of essential embodiment and of independence from specific physical realizations.

Indeed, we must recognize that while artificial systems often have clearly specified purposes, and thus may be definitely computational or not, things are not so clear cut in nature, which often combines multiple functions into a single system. For example, ant foraging may simultaneously bring food to the nest and accomplish computational tasks such as adaptive path finding, path minimization, and exploration. Also, the circulatory system transports oxygen and nutrients, but also transmits hormonal signals.

Indeed, even well-engineered artificial systems obey the *Shanley Principle*, which says that multiple functions should be combined into single parts; orthogonal design is important for prototyping, but it should be followed by integration of function [Knuth, 1974, p. 295]. Thus, as we push the limits of computing technology and embed it more deeply into our world, we will have to combine functions, which will result in systems that are less purely computational and more essentially embodied.

10. Non-Turing Computation

It is important to remember that Church-Turing (CT) computation is a *model* of computation and that, like all models, it has an associated *frame of relevance* [MacLennan, 2003 b; MacLennan, 2009 b]. As previously remarked, a model's frame of relevance is determined by its simplifying assumptions — by the aspects and degrees to which the model is similar to the modeled system or differs from it — since these (often unstated) assumptions determine the sort of questions the model is suited to answer. It is important to understand a model's frame of relevance, since if we use a model to address issues outside its frame of relevance, we are apt to learn more about the model and its simplifying assumptions than about the modeled system. For example, from a highway map we may infer the travel distance between cities from the length of a line on the map, but we cannot infer the width of the road from the width of the line, nor conclude that many cities have circular boundaries and are colored either black or red!

Recall that the theory of CT computation was developed to address issues in effective calculability and formalist approaches to mathematics;

the simplifying assumptions that it makes are well-suited to these issues and define its frame of relevance. Within this frame it makes sense to consider something computable if it can be computed in a finite number of steps (of finite but indeterminate duration) using a finite (but unbounded) amount of memory. It also makes sense to treat computation as a matter of function evaluation and to define computability in terms of sets of functions. (See [MacLennan, 1994 a, 2003 b, 2004, 2009 b] for more on the frame of relevance of CT computation.)

Unfortunately, the CT model is not well-suited to address issues in embodied computation or, more generally, natural computation, which lie outside its frame of relevance; its simplifications and approximations are bad ones for embodied computation systems. For example, the CT model ignores the real-time rates of the operations, but they are highly relevant in embodied computation. Similarly, the CT notions of equivalence and universality do not address the efficiency (in real-time, not asymptotic, terms) with which one system may simulate another.

Although it is premature to define a model of embodied computation, since we do not yet understand which issues are relevant and which are not, and premature formalization can impede the progress of a field, nevertheless we can produce a preliminary list of relevant issues. They include robustness (in the presence of noise, errors, faults, defects, and uncertainty), generality, flexibility, adaptability, morphological and steric constraints, physical size, consumption of matter and energy, reversible reactions, and real-time response [MacLennan, 2003 b; MacLennan, 2004; MacLennan, 2009 b].

11. Conclusions

In conclusion, we can see that embodied computation will play an increasingly important role in post-Moore's Law computing, but that we will need new models of computation, orthogonal to the Church-Turing model, that address the relevant issues of embodied computation and information processing. As a consequence we also expect there to be an ongoing fruitful interaction between investigations of embodiment in computation, psychology, and philosophy.

References

Abbreviations:
LSJ = Liddell, Scott, & Jones, *Greek-English Lexicon*, 9th ed., 1940.
OED = *Oxford English Dictionary*, 2nd ed., 1989.
OLD = *Oxford Latin Dictionary*, 1982.

Anderson, M. L. (2003). Embodied cognition: A field guide. *Artificial Intelligence*, 149, pp. 91–130.

Barish, R. D., Rothemund, P. W. K., and Winfree, E. (2005). Two computational primitives for algorithmic self-assembly: Copying and counting. *Nano Letters*, 5, pp. 2586–92.

Benzi, R., Parisi, G., Sutera, A., and Vulpiani, A. (1982). Stochastic resonance in climatic change. *Tellus*, 34, pp. 10–16.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm intelligence: From natural to artificial systems*. New York: Oxford Univ. Press.

Brooks, R. (1991). Intelligence without representation. *Artificial Intelligence*, 47, pp. 139–59.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, G., Theraulaz, J., and Bonabeau, E. (2001). *Self-organization in biological systems*. New York: Princeton Univ. Pr.

Clark, A. (1997). *Being there: Putting brain, body, and world together again*. Cambridge: MIT Press.

Dreyfus, H. (1979). *What computers can't do: The limits of artificial intelligence*, rev. ed. New York: Harper & Row.

Hamann, H., and Wörn, H. (2007). Embodied computation. *Parallel Processing Letters*, 17 (3), pp. 287–98.

Haykin, S. (1999). *Neural networks: A comprehensive foundation*, 2nd ed. Upper Saddle River: Prentice-Hall.

Iida, F., Pfeifer, R., Steels, L., and Kuniyoshi, Y. (Eds.) (2004). *Embodied artificial intelligence*. Berlin: Springer.

Johnson, M., and Rohrer, T. (2007). We are live creatures: Embodiment, American pragmatism, and the cognitive organism. In J. Zlatev, T. Ziemke, R. Frank and R. Dirven (Eds.), *Body, Language, and Mind*, Berlin: Mouton de Gruyter, vol. 1, pp. 17–54.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5, pp. 90–9.

Kirkpatrick, S., Gelatt, Jr., C. D., and Vecchi, M. P (1983). Optimization by simulated annealing. *Science*, 220, pp. 671–80.

Knuth, D. E. (1974). Structured programming with **go to** statements. *Computing Surveys*, 6 (4), pp. 261–301.

MacLennan, B. J. (1994 a). Continuous computation and the emergence of the discrete. In K.H. Pribram (Ed.), *Rethinking neural nets: Quantum fields and biological data*, Hillsdale: Lawrence-Erlbaum, pp. 199–232.

MacLennan, B. J. (1994 b). Continuous symbol systems: The logic of connectionism. In D.S. Levine and M. Aparicio IV (Eds.), *Neural networks for knowledge representation and inference*, Hillsdale: Lawrence-Erlbaum, pp. 83–120.

MacLennan, B. J. (2003 a). Molecular combinatory computing for nanostructure synthesis and control. In *IEEE Nano 2003 (Third IEEE Conference on Nanotechnology)*, IEEE Press.

MacLennan, B. J. (2003 b). Transcending Turing computability. *Minds & Machines*, 13 (1), pp. 3–22.

MacLennan, B. J. (2004). Natural computation and non-Turing models of computation. *Theoretical Computer Science*, 317, pp. 115–145.

MacLennan, B. J. (2009 a). Computation and nanotechnology (editorial preface). *International Journal of Nanotechnology and Molecular Computation*, 1 (1), pp. i–ix.

MacLennan, B. J. (2009 b). Super-Turing or non-Turing? Extending the concept of computation. *International Journal of Unconventional Computing*, 5 (3–4), pp. 369–387.

MacLennan, B. J. (in press). Models and Mechanisms for Artificial Morphogenesis. In Hiroshi Umeo (Ed.), *International Workshop on Natural Computing*. Springer series, Proceedings in Information and Communications Technology (PICT). Berlin: Springer.

Miller, M. I., Roysam, B., Smith, K. R., and O'Sullivan, J. A. (1991). Representing and computing regular languages on massively parallel networks. *IEEE Transactions on Neural Networks*, 2, pp. 56–72.

Pfeifer, R., and Bongard, J. C. (2007). *How the body shapes the way we think — A new view of intelligence*. Cambridge: MIT.

Pfeifer, R., Lungarella, M., and Iida, F. (2007). Self-organization, embodiment, and biologically inspired robotics. *Science*, 318, pp. 1088–93.

Pfeifer, R., and Scheier, C. (1999). *Understanding intelligence*. Cambridge: MIT.

Rimon, E. and Koditschek, D. E. (1989). The construction of analytic diffeomorphisms for exact robot navigation on star worlds. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, Scottsdale AZ. New York: IEEE Press, pp. 21–6.

Rothemund, P. W. K., Papadakis, N., and Winfree, E. (2004). Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology*, 2 (12), pp. 2041–53.

Rothemund, P. W. K, and Winfree, E. (2000). The program-size complexity of self-assembled squares. In *Symposium on Theory of Computing (STOC)*, New York: Association for Computing Machinery, pp. 459–68.

Steinbeck, O., Tóth, A., and Showalter, K. (1995). Navigating complex labyrinths: Optimal paths from chemical waves. *Science*, 267, pp. 868–71.

Stepney, S. (2004). Journeys in non-classical computation. In T. Hoare and R. Milner (Eds.), *Grand Challenges in Computing Research*, Swindon: BCS, pp. 29–32.

Stepney, S. (2008). The neglected pillar of material computation. *Physica D*, 237 (9), pp. 1157–64.

Ting, P.-Y., and Iltis, R. A. (1994). Diffusion network architecture for implementation of Gibbs samplers with applications to assignment problems. *IEEE Transactions on Neural Networks*, 5, pp. 622–38.

Tokuriki, N., and Tawfik, D. S. (2009). Protein dynamics and evolvability. *Science*, 324, pp. 203–7.

Turing, A. M. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society*, B 237, pp. 37–72.

Winfree, E. (1998). *Algorithmic self-assembly of DNA*. Unpublished doctoral dissertation, California Institute of Technology, Pasadena.

Wolfram, S. (2002). *A new kind of science*. Champaign, IL: Wolfram Media.