Aspects of Embodied Computing

Toward a Reunification of the Physical and the Formal

Technical Report UT-CS-08-610

Bruce J. MacLennan[*]

Department of Electrical Engineering & Computer Science
University of Tennessee, Knoxville
`www.cs.utk.edu/~mclennan/`

March 6, 2008
Revised August 6, 2008

Abstract

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 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.

[*] This report was the basis for a presentation at *The 2008 North American Conference on Computing and Philosophy: The Limits of Computation* (The International Association for Computing and Philosophy, Bloomington, IN, July 10–12, 2008). It may be used for any non-profit purpose provided that the source is credited.

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 the 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 (2004, p. 29) 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.

Traditionally, a sort of Cartesian dualitism 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.[1] 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.

[1]When I was in graduate school, I recall one of my fellow students announcing that he had heard of a program so perfect that it would run without benefit of a computer! This jest betrays a common underlying attitude in computer science.

Embodied Cognition

Johnson and Rohrer (2007) 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. 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 and Merleau-Ponty, have made similar points.

Hubert Dreyfus and others have stressed the importance and benefits of embodiment in cognition. As Dreyfus (1979) observed, there are many things that we (implicitly) know simply by virtue of having a body. 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) and Pfeifer and 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.

Embodied Computation

Pfeifer, Lungarella, and Iida (2007, p. 1088) provide a concise definition of *embodiment*: "the interplay of information and physical processes." This suggests that we can define *embodied computation* as information processing in which the physical realization and the physical environment play an unavoidable and essential role.

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, in which (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 its 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 gasses, and 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 (Miller, Roysam, Smith & O'Sullivan, 1991; 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 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 eventually 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 (e.g., simulated annealing, stochastic resonance) 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 may "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. The goal in embodied computation is to exploit the physics, not to circumvent it (which is costly).[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, 2003b). 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 a small a quantity of energy or matter (e.g., electrical charge) as possible — consistent with reliable operation — so that state changes will require as small a change 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 physi-

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

cal 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, electrical quantities, etc. For example, Winfree (1998) investigates *algorithmic assembly*, in which DNA computation is used to assemble nanostructures, and MacLennan (2003a) explores the use molecular computation based on combinator reduction for nanostructure synthesis and control.

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*.

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, Deneubourg, Franks, Sneyd, Theraulaz & Bonabeau, 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 this case we are interested in physical processes involving single elements, which may respond passively or actively. Examples of such *discrete 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 *continuous 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 (1999) in their investigations of swarm intelligence in wasp nest construction, recognized the concept of a *coordinated algorithm*, which leads to an organized nest structure. Similarly, we need to discover how to design coordinated algorithms for embodied computation in artificial morphogenesis and similarly complex applications.

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.

What distinguishes physically realized information processing from other physical processes is, I think, an interesting and important question, but outside the scope of this report. A few remarks must suffice here; towards the end this report I will address the relation between embodied computing and the familiar theory of Turing computation.

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 conclude that computation is *a physical process, the purpose or function of which is the abstract manipulation (processing) of abstract objects* (MacLennan, 1994a, 2004). That is, 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.

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. 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.

Related Work

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

According to Hamann and Wörn (2007) 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. 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 (in press) discusses the ideas of *material computation* and *in materio* computers, that is, systems in which the physical substrate "naturally" computes. 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.) 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 to which I now turn.

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, 2003b, in press). 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 define computability in terms of sets of functions. (See MacLennan, 1994a, 2003b, 2004, in press, 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, morphology and steric constraints, physical size, consumption of matter and energy, reversible reactions, and real-time response (MacLennan, 2003b, 2004, in press).

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. As a consequence we also expect there to be an ongoing fruitful interaction between investigations of embodiment in computation and philosophy.

References

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

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

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

Camazine, S., Deneubourg, J.-L., Franks, N.R., Sneyd, G., Theraulaz, J., & 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: A critique of artificial reason*. New York: Harper & Row.

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

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

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

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

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

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

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

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

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

MacLennan, B.J. (in press). Super-Turing or non-Turing? Extending the concept of computation. *International Journal of Unconventional Computing*.

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

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

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

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

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

Stepney, S. (in press). The neglected pillar of material computation. *Physica D*.

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

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