The Morphogenetic Path to Programmable Matter*

Bruce J. MacLennan, *Member, IEEE*

***Abstract*—The endpoint toward which reconfigurable systems should develop is *programmable matter*, that is, complex systems whose physical properties and structure can be controlled in a systematic way. This can be accomplished by recognizing that computational processes can be used to assemble and reconfigure complex, hierarchically structured systems. Programmable matter may be programmatically controlled externally or internally, which includes self-assembly. The best approach to the self-assembly of complex, hierarchical systems, such as future robots with capabilities comparable to those of animals, is by *artificial morphogenesis*, which adapts embryological morphogenesis to artificial systems. We review the requirements of self-assembling morphogenetic components.**

***Index Terms*—artificial morphogenesis, assembly systems, microassembly, micromechanical devices, molecular communication, multi-agent systems, multi-robot systems, nanofabrication, programmable matter, reconfigurable architectures, robot control, self-assembly, synthetic biology.**

I. Motivation

In this paper we consider *programmable matter* as the natural extension and goal of reconfigurable systems. By pursuing this ultimate goal, we will achieve more widely reconfigurable systems along the way. What could we accomplish with programmable matter?

One application of programmable matter is the assembly of complex, hierarchical systems, that is, systems with levels of complex structure from the micro (or even nano) scale up to the macro scale. The need to assemble such complex, hierarchical structures is illustrated by future robots with cognitive, perceptual, and physical competence comparable to that of animals. Consider the challenge of building a cat-size robot with the behavioral competence of a cat: able to run, leap, stalk and catch prey, etc. Among other problems, consider the sensors, effectors, and artificial nervous system of such a robot. The human retina has about 100 million sensory cells, intricately wired to reduce the dimension of the input to about one million for transmission on the optic nerve. Therefore, to produce an artificial retina we might want to assemble and interconnect some millions of simple sensors. Sophisticated robots with dexterous manipulators will require a delicate sense of touch and other bodily senses. Therefore, we will want to populate their surfaces ("skins") with dense arrays of sensors and to connect them to their central processors ("brains"). Animals' physical competence arises in part from their complex systems of muscles and tendons, which work synergistically to permit a wide variety of fluent, rapid, and robust physical actions. To accomplish this, we need to be able to assemble complex arrays of actuators and to connect them to facilitate synergistic control.

Such sophisticated sensors and effectors must be controlled in real time to achieve competent behavior, and one promising approach is massively parallel neural-network-style computing. (For example, brain-scale low-power electronic neuromorphic computers is the goal of a large DARPA program called SyNAPSE for "Systems of Neuromorphic Adaptive Plastic Scalable Electronics.") But the scale of parallelism is significant: 86 billion neurons in a human brain, 763 million in a cat brain. Furthermore, these neurons are intricately connected, with many neurons having tens of thousands of connections, and some with several hundred thousand. For achieving brain-scale intelligence, we may need to assemble artificial neural networks of comparable complexity. The point of this example is that we would like to be able to assemble complex systems with millions (or more) components in a hierarchical structure (that is, not simple homogeneous or periodic structures).

Not only would we like to assemble such systems from scratch, we would also like to be able to reconfigure complex systems, rearranging their components to address a new mission, recover from damage, improve their capabilities, etc. Typically reconfiguration entails changing the connections among a fixed set of elementary components. *Radical reconfiguration* goes further, altering the components themselves to create different hardware resources. If programmed assembly is like the development of a fetus, then radical reconfiguration is like the metamorphosis of a caterpillar into a butterfly.

Both assembly and reconfiguration are facilitated by *programmability*, by which we mean the ability to govern a large class of processes in some uniform way. In the first case, assembly, a program is used to control a general-purpose mechanism for assembling some complex, hierarchical system. Hierarchical structure in the program may be used to organize hierarchical structure in the product. In effect, the program "computes" a physically instantiated structure

B. J. MacLennan is with the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996 USA (e-mail: maclennan@utk.edu).

* Accepted version of MacLennan, B. J., "The Morphogenetic Approach to Programmable Matter," *Proc. IEEE*, Vol. 103, No. 7 (July 2015): 1226–1232.
 DOI: 10.1109/JPROC.2015.2425394

(arrangement of primitive elements) for its own sake or for controlling subsequent behavior. Programmability is also valuable for reconfiguration. As a program can create a physically instantiated structure, so it can recompute parts or all of a physically instantiated structure, allowing programmed reconfiguration. Approaches for accomplishing computation and recomputation of physical structures are considered later.

II. Computation is Physical

Rolf Landauer coined the slogan "information is physical," by which he intended to stress that information is always represented in a physical substrate, and therefore that information processing is bound by the laws of physics [1]. Viewing computation as a physical process suggests an approach to programmable matter. Our usual practice in computing is to use physical processes to support information processing, which we think of as an abstract process. In a general-purpose computer, abstract programs are used to govern these physical processes to realize a wide variety of abstract computations. But we can turn these ideas around, using computation for the sake of the physical processes it governs rather than for abstract information processing. Then the running of a program has the correlated physical effect of rearranging matter for some purpose, such as assembly or reconfiguration. In this way we have computation for the sake of physical processes rather than physical processes for the sake of computation. The tradeoffs, however, can be different. In computing our goal has been to move less matter (e.g., electrons), since this is faster and requires less energy, but for the purposes of programmable matter we often want to move more matter.

Programmable systems achieve their generality by controlling the interactions in a generic substrate in order to implement specific functions. For example, an ordinary computer program applies universal operations to a homogeneous array of bits in order to implement a specific application. In a similar way, programmable matter is based on a generic material that can be programmed to exhibit a wide variety of useful physical properties; we call this a *programmable material*.

It might seem unlikely that such a material could exist, but nature provides an example: *proteins* [2]. They are the principal structural elements of living things and fulfill an enormous variety of active and passive functions. These include the skeletons of cells, structural elements such as the collagen and elastin of which connective tissue is made, and the keratin of which horns, nails, and feathers are made. They also include ion channels, enzymes, motor proteins, transporter molecules, signaling molecules, receptors, and sensor molecules (e.g., rhodopsin). Through their very specific binding mechanisms, proteins can perform simple decision making processes by changing their conformation, and thereby altering their behavior, in response to a molecule in their environment [3]. Some proteins perform extremely complex functions, such as guiding the assembly of other proteins, transcribing RNA, and repairing DNA.

Nevertheless, this wide variety of physical properties is produced by just a few different kinds of basic building blocks (amino acids). This is because chains of amino acid residues can fold into an endless variety of physical conformations with a wide range of physical properties. These chains are represented by sequences of nucleotides in DNA, and therefore this wide range of physical properties is programmed, in effect, by strings of A, C, T, and G.

What can we learn from proteins that is applicable to programmable matter? First, that a general-purpose programmable material can be based on a small number of building blocks that can be combined in a large number of ways that lead to different physical properties. Second, that arrangements of the building blocks should have descriptions that are simple, but that need not be strings, like DNA. We may call such a description a *genotype*. The genotypes should be combinatorially rich, that is, they should be capable of expressing a nearly endless variety of arrangements of the building blocks. Finally, as proteins fold into complex shapes, which give them their properties, so an arrangement of the building blocks may reorganize itself into a configuration with specific physical properties, its *phenotype*. The goal, then, for a *programmable material* is for the space of possible phenotypes to encompass a wide variety of useful physical properties. Proteins prove that it can be done.

The space of physical properties can be further expanded by making the material *functionalizable*. That is, the phenotypes can be functionalized for a specific purpose by the incorporation of additional building blocks with specific properties, such as electrical conductivity, hardness, or photosensitivity.

III. Characteristics of the Microscopic Domain

Our physical intuition can mislead us at the microscopic scale, and so in this section we review some characteristics typical of the microscopic domain, which programmable matter must accommodate and — preferably — exploit. First, various sorts of imperfection, including noise, imprecision, defects, and faults, loom larger at the microscopic scale and are ultimately unavoidable. We commonly think of these properties as reflecting an undesirable deviation from an unphysical ideal (a noiseless signal, perfect or faultless operation, etc.). This is a sort of Platonic perspective, in which we imagine perfect forms imperfectly impressed on recalcitrant matter. This has been a useful viewpoint for conventional computing technology and macroscopic engineering more generally. It is less useful at the microscopic scale, where the ideal is often physically unrealistic. Therefore we should look at programming differently.

At very least we must use robust methods that are either insensitive to errors or that correct them after they have occurred. This is the conventional approach, except that we must begin our design with these considerations, rather than adding them on as afterthoughts. A better approach, however, is to take a lesson from nature and learn to exploit these "errors" as sources of *free variability*. For example, algorithms such as simulated annealing exploit randomness to escape from local optima [4], and stochastic resonance uses noise to

enhance signal detection [5]. More generally, free variability can be used for exploring various possibilities.

Other effects, which are negligible at the macroscopic level, are significant and potentially useful at the microscale. For example, relatively weak molecular interactions, such as van der Waal's forces, are significant at the microscale and can be used to coordinate processes. Ambient energy may be available for use in the form of light, temperature gradients, external agitation, chemical energy, etc. Finally, at the very small scale, we cannot ignore quantum effects, such as tunneling, superposition, and entanglement, which might be used to advantage.

IV. Externally Programmed Assembly

As a step toward programmable matter we can consider *externally programmed assembly*, in which a conventional computer controls reconfiguration of a general-purpose material. This approach is based on the observation that information structures are physical; that is, when a program computes a data structure, it results in a physical rearrangement of the matter in the computer's memory. The differing bits are represented by differing distributions of matter (e.g., electrons). If our goal is information processing, then these distributions' sole purpose is to reliably represent the information. But if our goal is programmable matter, then the object of the computation is the change in physical state.

We have simple examples of such systems in field-programmable gate arrays, analog arrays, and transistor arrays, which provide an array of general-purpose devices and an externally programmable means of interconnecting them. These systems are precursors to more general kinds of externally programmed matter, but to extend the scope of externally programmable matter beyond the reconfiguration of electronic circuits, we need to use addressable cells of a few types but with a wider range of physical properties.

First, while some cells will require only a small number of possible physical states into which they can be switched, other types of cells will have a continuum of possible physical states (e.g., resistance, capacitance, opacity, refractive index). Further, the set of cell types should be capable of exhibiting a wide variety of useful physical properties, not just electronic. For example, some cells will have properties that are optical or fluidic. For greater generality, some of the cell types should have active states, such as amplification, energy conversion (e.g., photovoltaic), light emission, fluidic valves, and MEMS. Other states will be passive, implementing connections of various sorts between the other cells or implementing other fixed material structures. The difficult challenge, of course, is to define a minimal set of economically implementable cell types that together span a large space of useful physical properties and behaviors.

V. Programmable self-assembly

The scope of externally programmable matter is limited because all the cells must be accessible, directly or indirectly, by the external programming mechanism. It may be difficult to program them in parallel, and sequential programming may result in long latencies. This limitation is overcome by *internally programmed matter*, in which the individual components contain their own programming and self-organize into the desired structure. In particular, programmed self-assembly has the potential of producing much more complex structures than does externally programmed matter.

One example is algorithmic assembly by DNA, in which DNA molecules are programmed (through their nucleotide sequences) to self-assemble into desired structures [6], [7], [8]. DNA fragments anneal into their most stable structures. More generally, *one-pot molecular computation* refers to processes in which the molecular reactions sequence themselves without detailed external control; each reaction enables the next. When permitted, assembly proceeds with *molar* or *Avogadro-scale* parallelism. The DNA molecules can be functionalized with metal particles or other molecular groups suitable to the application. However, programmable self-assembly does not have to be limited to DNA or RNA, for we can design new molecules specifically for programmable matter, for example, *xeno-nucleic acids* (XNAs): polymers of non-natural nucleotide bases [9].

Programmable self-assembly will be advanced by investigating nonstandard models of computation that are inherently parallel and model molecular reactions. For example, *combinatory logic* is a computationally universal (Turing-complete) model of computation in which computation proceeds by tree substitutions [10]. It satisfies the *Church-Rosser property*, which means that substitutions can be done in any order, and if they terminate, they will always produce the same result (the same tree structure). Therefore, combinatory logic provides a formal model capable of reliably generating any computable tree structure in spite of operations proceeding asynchronously and in parallel [11]. It is an example of a model of computation more suitable to programmable self-assembly than standard models.

VI. Artificial Morphogenesis

How can we assemble more complex structures, such as the future robot described at the beginning of this article? One answer is to learn from nature, for an embryo develops autonomously from a single cell into a complex, hierarchically structured body composed of trillions of cells. Of course, technology is not obliged to ape nature; there may be better ways to assemble such systems. But embryological development provides an existence proof that complex self-assembly can be accomplished, and suggests a way forward that we know can scale to trillions of components.

Morphogenesis refers to the embryological processes that create three-dimensional form in the body. It arises from the behavior of individual cells, which is governed by their genetic regulatory circuitry, which governs their communication and coordination. Therefore, *artificial morphogenesis* presents itself as a possible approach to programmed self-assembly of large-scale systems [2], [12], [14], [15], [16], [17], [18], [19], [20].

To illustrate the process we consider *somitogenesis* in the vertebrate embryo. The number of vertebrae is, of course, characteristic of a species (33 for humans, 55 for chickens, etc.), which raises the question of how masses of individually unintelligent fetal cells are able to count to a specific number. This has been explained by the *clock-and-wavefront model*, recently confirmed [21], [22]. In brief, the embryo lengthens by cell proliferation in the tail region, which contains a pacemaker that generates a pulse of chemical (a *morphogen*) at regular intervals. This pulse stimulates nearby cells also to generate a pulse of the morphogen, and so a wave of morphogen propagates toward the head of the embryo. (After generating a pulse, a cell enters a refractory period, which ensures that the signal propagates in one direction.) When this signal passes through undifferentiated cells in an appropriate context, they differentiate into spinal segment cells. This context is defined by two other morphogens, which diffuse from the tail and anterior differentiated segments, respectively, so that the new segment differentiates in a region of their joint low concentration posterior to already differentiated segments. The number and length of the segments is determined by the duration of growth, the growth rate, the diffusion constants, and the pacemaker frequency. Through simulation, we have shown that the clock-and-wavefront model can be used to assemble an artificial spine for an insect-like robot frame, analogous to the morphogenetic process in embryos, but we have also applied it to the generation of segmented legs [23]. This demonstrates that we can learn the natural mechanisms of embryological morphogenesis and apply them in new contexts in artificial morphogenesis.

In the remainder of this section, we discuss the general requirements for a morphogenetic approach to programmable matter. We focus on techniques inspired by developmental processes in embryos (as opposed to those, for example, in plants) because of their ability to generate a wider variety of complex hierarchical structures.

A morphogenetic approach to programmable matter can be organized around two broad classes of objects: active *assemblers* and passive *components*; together they fulfill roles analogous to cells and their products in embryological morphogenesis. The components include structural elements and physical carriers of signals (e.g., chemical morphogens). These components are passive with respect to the assembly process, but might be active in the assembled system. For example, the components could include objects that will function as sensors, actuators, or amplifiers in a robot, but are manipulated passively during assembly and reconfiguration.

Assemblers are the active agents in artificial morphogenesis. Like cells in a developing embryo, they can signal each other and coordinate their movement and other actions in order to assemble a three-dimensional object. Assemblers are also capable of transporting the passive components. In principle, everything could be done with assemblers, but it is likely that the components can be simpler, less expensive, and easier to produce than assemblers. (In many cases the components can be single molecules or small inert structures.) Therefore it is useful to distinguish those elements that need to behave as active agents (assemblers) and those that do not (components).

Assemblers can be *artificial* (i.e., microrobots) or *biological* (genetically engineered cells). One significant difference between artificial assemblers and embryonic cells, at least at this time, is that artificial assemblers are not self-reproducing. This is a significant difference, since cell proliferation is a fundamental process in embryological morphogenesis [24], [25], and so the morphogenetic functions of proliferation must be fulfilled by other means. One way is by providing an external supply of assemblers; another way is by using transport of passive components to an area that would otherwise be a site of cell proliferation. Either way, natural morphogenesis will have to be adapted to artificial assemblers. This is an ongoing topic of research, but simulations have demonstrated, for example, the formation fine capillary networks by cell migration [26], pillar assembly by mobile agents [12], and the collision-free routing of neuron-like connections [27]. In morphogenesis, diffusion of morphogens can guide masses of mobile agents from waypoint to waypoint toward a destination.

An alternative to artificial assemblers is to exploit the capabilities of living cells for metabolism and reproduction by genetically engineering them to function as assemblers [28], [29], [30], [31]. This has the additional advantage that cells are able to move and can emit and respond to chemical signals, which are especially useful in morphogenesis. Since biological cells naturally perform many of the functions required of assemblers, the required genetic modifications might not be too difficult, but this is a subject of ongoing research [32].

We briefly review the capabilities of assemblers, whether artificial or biological. First we consider the *effectors*, that is, the means by which an assembler can act on its environment, including other assemblers. The functions of effectors include adhesion, mobility, shape change, transport, and signaling.

Controlled adhesion among assemblers and between assemblers and passive components is important for mobility, transport, and assembly. There are a variety of mechanisms for controlled adhesion. For example, adhesion can be implemented mechanically, latching an assembler onto another object permanently or temporarily. Electrically powered assemblers might use electrostatic or magnetic adhesion. Finally, both artificial and biological assemblers might use molecular processes for controlled adhesion. This is the most common mechanism in biological morphogenesis and permits a range of binding strength and permanence [25].

Cell migration is one of the fundamental driving forces in biological morphogenesis [24], [25], and therefore our assemblers must be mobile, but requirements differ depending on whether the assemblers move through a fluid or on solid objects. Moreover, there are both passive and active strategies for moving through fluids. Assemblers move passively by Brownian motion, agitation, convection, etc. In these cases an assembler must recognize when it is in an appropriate location and adhere to other objects. Alternatively, an assembler may actively move itself, for example using flagella as microorganisms do [28].

Although self-assembly in a fluid medium is perhaps easiest to understand, biological morphogenesis takes place in the realm of *soft matter* or *viscoelastic materials* [33], [34], that is, by cells adhering more or less tightly to each other and to non-living materials. Therefore, cells are often crawling across other cells or extracellular substrates created by cells, or they are migrating through other populations of cells or viscoelastic

extracellular matrices. We anticipate the use of similar processes in artificial morphogenesis. Assemblers might crawl, as some cells do, or move by differential adhesion, which is also important in natural morphogenesis.

Biological morphogenesis depends in part on cells changing shape, sometimes inducing stresses and strains that reshape tissues. Likewise, assemblers might use a limited ability to change shape (e.g., extending or contracting) in order to control the developing form and as means of movement.

Assemblers should also be capable of transporting other objects, both passive components and other assemblers. Objects might be transported by attaching to them using any of the adhesive mechanisms previously described. Alternately, objects might be transported by *inclusion*, that is, by taking them temporarily into the transporter.

Finally, assemblers may affect their environment by generating signals, which are intended to be interpreted by other assemblers. This is indeed the primary means for coordinating morphogenesis. In artificial morphogenesis, as in biological morphogenesis, we expect many of these signals to be chemical. Diffusion is a useful mechanism for distributing morphogens, and by controlling diffusion and rate constants regions can be defined in the developing structure (as illustrated by the clock-and-wavefront process). The clock-and-wavefront process also illustrates how *active diffusion* (diffusion facilitated by active agents) can be used to propagate signals in a desired direction. The signal carriers need not be limited to molecules, and other kinds of components can be used as signal carriers, moved either passively by diffusion or actively by transport.

The sensors available on the assemblers should be matched to signals produced by the assemblers, but also to external signals, which might be used to control the assembly process at a global level. Since we expect many morphogens to be chemical, assemblers will require chemical sensors. Biological assemblers are especially suited for this kind of communication and coordination [35]. However, since components can be used as signals, it may be easier to design sensors for them. Light is especially useful for carrying signals from outside the system, and so optical sensors will be useful in some systems.

Assemblers also require some means of behavioral control, using sensor input and internal state to govern the effectors. Fortunately, the algorithms required for morphogenesis are relatively simple, and so assemblers do not require much computing power. Generally analog control is more appropriate than digital; this is the case in embryological morphogenesis and, by extension, in artificial morphogenesis. Analog computation is easier to implement in microrobots, since it requires fewer components and consumes less power, in general [36]. For example, addition can be implemented by directly combining physical quantities (e.g., charges, currents, chemical concentrations) and integration can be implemented by accumulation of such quantities [37]. (Assemblers can, of course, be in distinct states, but this is easily implemented by analog computation.)

Cell differentiation is one of the principal regulatory mechanisms in biological morphogenesis [24]. Internal state and external signals cause a fetal cell to switch off some genetic regulatory circuits and to switch on others, thereby changing the behavior of the cell and utimately its structure. Analogous mechanisms allow assemblers to switch between behaviors (essentially activating or deactivating terms in differential equations) [12]. We do not expect artificial assemblers to have as wide a range of behaviors, since they do not construct themselves as cells do. Therefore, similar versatility can be achieved by assembling a variety of passive components. Progress in programmable materials, however, might allow artificial assemblers to approach the behavioral and structural versatility of fetal cells.

Limiting differentiation might actually facilitate radical reconfiguration in artificial morphogenesis. In biological metamorphosis, cells must *dedifferentiate* into pluripotent cells before *redifferentiating* into their new state. Artificial assemblers, in contrast, retain their full behavioral repertoire, and therefore can disassemble and reassemble structures as required. Similarly, damaged structures can be regenerated without dedifferentiation [38].

Finally, of course, assemblers must be powered, but their small size precludes many conventional power sources. One option is to follow the lead of embryological morphogenesis and use biological metabolism. This is an attractive option for biological assemblers (i.e., genetically engineered microorganisms), but is not currently feasible for artificial assemblers. However, artificial assemblers might be preloaded with living cells and use their metabolic products for energy [30]. Short of full-scale metabolism, assemblers might make use of other forms of chemical energy, for example by catalyzing reactions among reactants provided in their environment. Alternately, chemical fuel, provided in the environment, can drive molecular motors [39], [40]. Light is another vehicle for conveying energy to assemblers, where it can be captured by photovoltaic transducers or used to drive chemical reactions. Other forms of electromagnetic radiation might also be used.

VII. Conclusion

Programmable matter can be achieved by exploiting the fact that computation is physical. In ordinary computation, physical processes are used to realize abstract computation, which is the goal. For programmable matter the tables are turned, and abstract computation is used to govern the rearrangement of matter, which is the goal. The program controls the pattern that is imposed on a neutral substrate, which is universal for a class of applications. Proteins demonstrate that a small set of building blocks can be assembled in myriad different combinations to exhibit a wide variety of physical properties, and this suggests approaches to making programmable materials. Programmable matter can be controlled externally, as an ordinary computer's memory is controlled by its processor, or it can be controlled internally, self-organizing or reorganizing to produce the required structure. For assembling complex, hierarchical structures composed of many millions of components, embryological morphogenesis provides the best example to date. Therefore artificial morphogenesis, in which artificial or biological assemblers coordinate their action, can be used as a basis for implementing programmable matter.

REFERENCES

[1] R. Landauer, "The physical nature of information," *Phys. Lett. A*, vol. 217, p. 188, 1996.

[2] B.J. MacLennan, "Embodied computation: Applying the physics of computation to artificial morphogenesis," *Parallel Processing Letters*, vol. 22, no. 3, p. 1240013, 2012.

[3] D. Bray, *Wetware: A Computer in Every Living Cell*. New Haven: Yale University Press, 2009, pp. 63–5, 78–9.

[4] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–80, 1983.

[5] R. Benzi, G. Parisi, A. Sutera, and A. Vulpiani, "Stochastic resonance in climatic change," *Tellus*, vol. 34, pp. 10–16, 1982.

[6] P.W.K. Rothemund and E. Winfree, "The program-size complexity of self-assembled squares," in *Symposium on Theory of Computing (STOC)*, New York: Assoc. for Computing Machinery, 2000, pp. 459–468.

[7] R.D. Barish, P.W.K. Rothemund, and E. Winfree. (2005). "Two computational primitives for algorithmic self-assembly: Copying and counting," *Nano Letters*, vol. 5, pp. 2586–2592, 2005.

[8] P.W.K. Rothemund, N. Papadakis, and E. Winfree, "Algorithmic self-assembly of DNA Sierpinski triangles," *PLoS Biology*, vol. 2, no. 12, pp. 2041–2053, 2004.

[9] M. Schmidt, "Xenobiology: A new form of life as the ultimate biosafety tool," *Bioessays*, vol. 32, no. 4, pp. 322–331, April 2010.

[10] H.B. Curry, R. Feys, and W. Craig, *Combinatory Logic, Volume I*, Amsterdam: North-Holland, 1958.

[11] B.J. MacLennan, "A formal model of universal algorithmic assembly and molecular computation," *Intl. J. Nanotechnology & Molecular Computation*, vol. 2, no. 3, pp. 55–67, July–Sept. 2010.

[12] B.J. MacLennan, "Morphogenesis as a model for nano communication," *Nano Communication Networks J.*, vol. 1, pp. 199–208, 2010. DOI: 10.1016/j.nancom.2010.09.007.

[13] H. Kitano, "Morphogenesis for evolvable systems," in *Towards Evolvable Hardware: The Evolutionary Engineering Approach*, E. Sanchez and M. Tomassini, Eds. Berlin: Springer, 1996, pp. 99–117.

[14] R. Nagpal, A. Kondacs, and C. Chang," Programming methodology for biologically-inspired self-assembling systems," in *AAAI Spring Symp. on Computational Synthesis: From Basic Building Blocks to High Level Functionality*, March 2003. URL http://www.eecs.harvard.edu/ssr/papers/aaaiSS03-nagpal.pdf.

[15] A. Spicher, O. Michel, and J.-L. Giavitto, "Algorithmic self-assembly by accretion and by carving in MGS," in *Proc. of the 7th International Conference on Artificial Evolution (EA'05)*, volume 3871 of LNCS. Berlin: Springer-Verlag, October 2005, pp. 189– 200.

[16] S. Murata and H. Kurokawa, "Self-reconfigurable robots: Shape-changing cellular robots can exceed conventional robot flexibility," *IEEE Robotics & Automation Mag*., pp. 71–78, March 2007.

[17] R. Doursat, "Organically grown architectures: Creating decentralized, autonomous systems by embryomorphic engineering," in *Organic Computing*, R. P. Würtz, Ed. Berlin: Springer, 2008, pp. 167–200.

[18] P. Bourgine and A. Lesne (Eds.). *Morphogenesis: Origins of Patterns and Shapes*. Berlin: Springer, 2011.

[19] J.-L. Giavitto and A. Spicher, "Computer morphogenesis," in *Morphogenesis: Origins of Patterns and Shapes*, P. Bourgine and A. Lesne, Eds. Berlin: Springer, 2011, pp. 315–340.

[20] R. Doursat, H. Sayama, and O. Michel (Eds.), *Morphogenetic Engineering: Toward Programmable Complex Systems*. Berlin: Springer, 2012.

[21] J. Cooke and E.C. Zeeman, "A clock and wavefront model for control of the number of repeated structures during animal morphogenesis," *J. Theoretical Biology*, vol. 58, pp. 455–476, 1976.

[22] M.-L. Dequéant and O. Pourquié, "Segmental patterning of the vertebrate embryonic axis," *Nature Rev. Genetics*, vol. 9, pp. 370–382, 2008.

[23] B.J. MacLennan, "Molecular coordination of hierarchical self-assembly," *Nano Communication Networks J.*, vol. 3, no. 2, pp. 116–128, June 2012. http://dx.doi.org/10.1016/j.nancom.2012.01.004

[24] G.M. Edelman. *Topobiology: An Introduction to Molecular Embryology*. Basic Books, New York, 1988, p. 17.

[25] I. Salazar-Ciudad, J. Jernvall, S. Newman, "Mechanisms of pattern formation in development and evolution," *Development*, vol. 130, pp. 2027–2037, 2003.

[26] A. Gamba, D. Ambrosi, A. Coniglio, A. de Candia, S. Di Talia, E. Giraudo, G. Serini, L. Preziosi, and F. Bussolino, "Percolation, morphogenesis, and Burgers dynamics in blood vessels formation," *Physical Rev. Letters*, vol. 90, pp. 118101-1–118101-4, 2003.

[27] B.J MacLennan, "Coordinating massive robot swarms," *Intl. J. Robotics Applications & Technologies*, vol. 2, no. 2, pp. 1–19, 2014. doi: 10.4018/IJRAT.2014070101

[28] B. Behkam and M. Sitti, "Bacterial flagella-based propulsion and on/off motion control of microscale objects," *Applied Physics Letters*, vol. 90, p. 023902, 2007. http://dx.doi.org/10.1063/1.2431454

[29] E.B. Steager, M.S. Sakar, D.H. Kim, V. Kumar, G.J. Pappas, and M.J. Kim, "Electrokinetic and optical control of bacterial micro-robots," *J. Micromechanics & Microengineering*, vol. 21, p. 035001, 2011.

[30] M.S. Sakar, E.B. Steager, D.H. Kim, A.A. Julius, M.J. Kim, V. Kumar, and G.J. Pappas, "Modeling, control and experimental characterization of microbiorobots," *Intl. J. Robotics Research*, vol. 30, pp. 647–658, 2011.

[31] E.B Steager, D. Wong, D. Mishra, R. Weiss, and V. Kumar, "Sensors for micro bio robots via synthetic biology," in *Robotics and Automation (ICRA), 2014 IEEE Intl. Conf.*, pp.3783–3788, May 31 2014 – June 7 2014. doi: 10.1109/ICRA.2014.6907407

[32] Ferber, D. "Microbes made to order," *Science*, vol. 303, pp. 158–161, January 2004.

[33] G. Forgacs and S.A. Newman, *Biological Physics of the Developing Embryo*. Cambridge, UK: Cambridge University Press, 2005, p. 98.

[34] Y.C. Fung, *Biomechanics: Mechanical Properties of Living Tissues*. New York: Springer-Verlag, 1993.

[35] P.S.S. Kim, A. Becker, Yan Ou, A.A. Julius, A.A., and Min Jun Kim, "Swarm control of cell-based microrobots using a single global magnetic field," in *Ubiquitous Robots and Ambient Intelligence (URAI), 2013 10th Intl. Conf.*, pp. 21–26 , Oct. 30 2013 – Nov. 2 2013. DOI: 10.1109/URAI.2013.6677461

[36] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989, pp. 70–71, 87–99.

[37] B.J. MacLennan, "The promise of analog computation," *Intl. J. General Systems*, vol. 43, no. 7, pp. 682–696, 2014. DOI: 10.1080/03081079.2014.920997

[38] J.P. Brockes & A. Kumar, "Plasticity and reprogramming of differentiated cells in amphibian regeneration," *Nature Rev. Mol. Cell Biol.*, vol. 3, pp. 566–574, 2002.

[39] N. Koumura, R.W.J. Zijlstra, R.A. van Delden, N. Harada, and B.L. Feringa, "Light-driven monodirectional molecular rotor," *Nature*, vol. 401, pp. 152–155, 1999.

[40] B. Yurke, A.J. Turberfield, A.P. Mills Jr, F.C. Simmel, and J.L. Neumann, "A DNA-fuelled molecular machine made of DNA," *Nature*, vol. 406, pp. 605–608, 2000.

Bruce J. MacLennan (M'87) earned a B.S. in mathematics (with honors) from Florida State University (Tallahassee, Florida, USA) in 1972 and M.S. (1974) and Ph.D. (1975) degrees in computer science from Purdue University (Lafayette, Indiana, USA).

He was a Senior Software Engineer at Intel (1976–9), where is worked on the 8086 and iAPX-432 microprocessors. After returning to academia he was an Assistant Professor (1979–83), Associate Professor (1983–7), and Acting Chair (1984–5) of the Computer Science Department of the Naval Postgraduate School (Monterey, California). Since 1987 he has been an Associate Professor in the Department of Computer Science (1987–2007) and Department of Electrical Engineering and Computer Science (2007–present) of the University of Tennessee, Knoxville. He authored *Principles of Programming Languages: Design, Evaluation, and Implementation* (1st ed., New York, NY: Holt, Rinehart & Winston, 1983; 2nd ed., New York, NY: CBS College Publ., 1987; 3rd ed., New York, NY: Oxford Univ. Pr., 1999) and *Functional Programming: Practice and Theory* (New York,

NY: Addison-Wesley, 1990), and he edited *Theoretical and Technological Advancements in Nanotechnology and Molecular Computation: Interdisciplinary Gains* (Hershey, PA: IGI Global, 2011). He has published about 75 refereed journal articles and book chapters and was founding Editor-in-Chief of the *International Journal of Nanotechnology and Molecular Computation* (2007–14). At the Naval Postgraduate School, his research focused on programming language design, massively parallel computation, and neural networks. His research now focuses on unconventional computation, bio-inspired computation, embodied artificial intelligence, and self-organizing systems.

Prof. MacLennan is a member of the American Association for the Advancement of Science. He is a past Fellow if the Institute for Advanced Studies of the Collegium Budapest (1997).