

# Bridging Biology and Engineering Together with Spatial Computing

Jacob Beal<sup>1</sup>

Raytheon BBN Technologies, Cambridge MA 02138, USA  
jakebeal@bbn.com

**Abstract.** Biological systems can often be viewed as spatial computers: space-filling collections of computational devices with strongly localized communication. Applying a continuous-space abstraction allows the behavior of such systems to be modeled or specified in terms of aggregate geometry and information flow. This can simplify both the engineering of biological systems and the application of biological models to the engineering of non-biological systems, as illustrated by examples from synthetic biology and morphogenetic engineering.

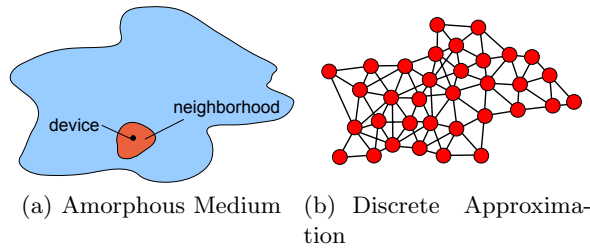
**Keywords:** spatial computing, synthetic biology, morphogenesis

## 1 Motivation

Forging links between biological systems and engineering practice is often challenging. The complexities of the biological world and the architectural thinking of current engineering practice are often simply at odds with one another. In many cases, however, we can find a useful intermediate point between these two worlds in the notion of a *spatial computer*: any potentially large collection of computational devices, distributed through space, in which the ability of devices to communicate is strongly correlated with physical distance. There are many examples of such systems in the engineering world, such as sensor networks, ad-hoc mobile networks, pervasive systems, robotic swarms, and reconfigurable computing architectures. There are also many examples in the natural world, such as biofilms, multicellular organisms, flocking birds, and insect colonies.

Although the details of such systems are often radically different, the constraints imposed by spatial locality frequently dictate similar solutions. For example, gradients are a common organizing pattern in biological systems, guiding such phenomena as embryo development[8] and the movement of foraging ants[9]. Gradients are also used in the engineering world in such diverse areas as distance vector routing (e.g., [11]), collective construction[12], and crowd management[10]. There are a number of similar such “building block” patterns, such as symmetry breaking and approximate consensus, that have either emerged independently in both worlds or where models of natural phenomena have been adapted for engineering use, and many of these patterns are spatial in nature.

We may thus hypothesize that further linkages between engineering and biology will be facilitated by adoption of spatial computing models, such as the



**Fig. 1.** The amorphous medium is an approximable continuous computing abstraction.

amorphous medium abstraction[1]. Such models should allow engineers to abstract away many domain-specific implementation details, laying bare the essential spatio-temporal structure of a phenomenon, and making it easier to transport from domain to domain.

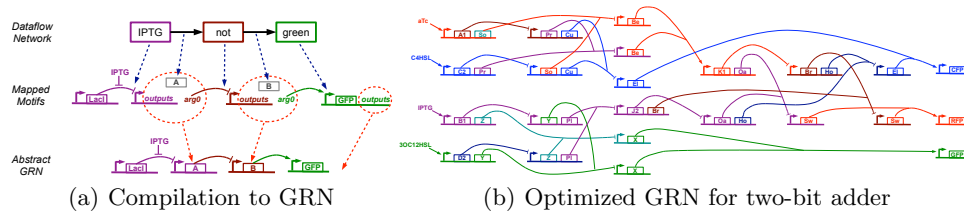
The remainder of this paper expands on this hypothesis, first by more closely examining the amorphous medium abstraction, and its use in the Proto spatial computing language, then by discussing two examples where these have been used in projects that link engineering and biology. In the case of synthetic biology, the spatial computing view has allowed organism- or colony-level descriptions to be automatically transformed into genetic regulatory networks for controlling the behavior of cells. In the case of morphogenetic engineering, the manifold geometry of the amorphous medium abstraction has allowed ideas from embryogenesis to be applied to the adaptation of robot designs. Finally, we will conclude with suggestions of future research directions.

## 2 Space as a Unifying Abstraction

An *amorphous medium*[1] is a compact Riemannian manifold with a computational device at every point (Figure 1). Information flows through this manifold with a fixed maximum velocity of  $c$ . Finally, every device has access to the recent past state of all other devices in some local neighborhood around it: specifically, all state in the intersection of the device’s past light cone with its neighborhood.

Obviously, we cannot actually build a system that contains an uncountably infinite number of devices. We can, however, view any system of locally communicating computing devices as a discrete approximation of an amorphous medium. Each device then represents a small region of nearby space, and messages sent from device to device (whether they be radio packets, chemical markers, or other) implement the information flow through neighborhoods.

The Proto spatial computing language[3] uses this abstraction to simplify the construction of distributed algorithms by making many of the details implicit. With Proto, the programmer writes a program that operates over regions of space-time, manipulating *fields* that map each point in space-time to a value. For example, a field representing a chemical morphogen gradient in a developing embryo would assign a chemical concentration to each location in the embryo



**Fig. 2.** Proto BioCompiler transforms programs to genetic regulatory network designs.

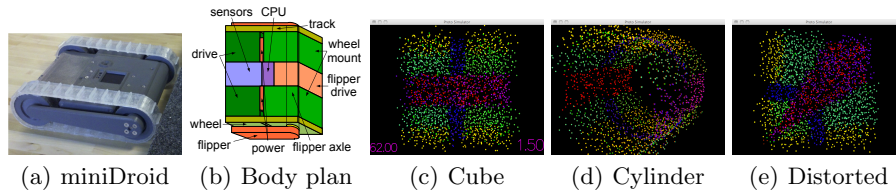
and specify how that concentration evolves over time. A *hox* gene might then be abstracted as a comparison function that is applied to this field in order to produce a Boolean field that indicates choice of cell fate.

Proto has four families of operators for producing and manipulating fields: *pointwise* operators do “normal” computing like logic and mathematics, *restriction* operators change where a program is run, *feedback* operators create and evolve state over time, and *neighborhood* operators gather information from the neighborhood (neighbor state and spatial metrics) and compute over it. Proto is a functional language, so programs are formed by combining these operators together using the rules of mathematical function composition. This functional composition, combined with a careful choice of operators, makes it possible for the Proto compiler to automatically transform these aggregate level spatial programs into programs for the individual devices of an amorphous medium. The amorphous medium is then approximated by physical devices. The details of communication and state management can thus be derived from a simpler description of aggregate behavior, and the manifold geometry and discrete approximation provide implicit scalability, robustness, and adaptability[7].

All then that must be determined for a given realization on physical devices, whether biological or otherwise, is how those devices can be used to approximate an amorphous medium. A biological model, written for an amorphous medium, can thus be executed equivalently on a non-biological system, and vice versa.

### 3 Example Applications

Synthetic biology is the application of engineering design methods, particularly from electrical engineering and computer science, to genetic engineering. A major challenge in this field has been effective design of systems containing more than a few engineered elements, partly due to the large variety of different factors that a design must take into account. A colony of cells, however, may be viewed as a spatial computer, and the continuous-time dataflow model that lets Proto cope with parallel and asynchronous execution in networks maps just as well onto the a parallel and asynchronous dynamics of a genetic regulatory network (GRN)[4]. We have taken advantage of this to construct the Proto Bio-Compiler[6], which uses a motif-based method to transform Proto programs into GRN designs by associating each primitive with a motif (Figure 2). The resulting



**Fig. 3.** MADV facilitates variation of designs like the iRobot miniDroid (a) with an embryonic body plan (b, c) that automatically adapts to distortion (d, e).

designs are then optimized, allowing automated creation of complex genetic regulatory networks. The compiler guarantees that, if appropriate DNA parts can be obtained, the design GRN will correctly implement the high-level program. This also allows execution to be simulated on simple message-passing network models, so programs can be debugged there before proceeding to verification in computationally expensive chemical simulations.

Moving in the opposite direction, Morphogenetically-Assisted Design Variation (MADV) applies models of morphogenesis to adapt robot designs. Here the challenge is to maintain the viability of a tightly integrated design as portions of it are modified in response to the desires of a designer or the changing demands of the external environment. The relationships between key parameters of the design are set by functional blueprints[2], which specify a design in terms of how it can be changed to maintain desired properties. Most parameters of the design, however, are implicit in the relationships between key parameters, and these are established instead by simulating development of a robot “embryo” [5]. Figure 3 shows an example robot, the iRobot miniDroid, the early embryo body plan used to establish design relations, and examples of a partial body plan, simulated in Proto and adapted in response to distortion of the embryo. This adaptation is not explicitly designed, but is implied by the continuous space manifold model of Proto, which also allows the embryo to be simulated with a much smaller number of cells than in many natural embryos, albeit more coarsely.

## 4 Future Directions

We have thus seen that continuous space-time abstractions may be fruitful in connecting biological systems and engineering models, whether to improve the engineering of biology or to improve other fields of engineering by importing concepts from biology. There are many possible avenues in which these connections might be broadened. For example, biological organisms use many morphological operations that have not been well explored for use in engineered systems, and the potential of swarming remains underutilized in the engineering of infrastructure and logistics. Likewise, spatial computing may focus engineers better on the key issues in areas like ecology, helping to avoid some of the pitfalls frequently encountered. Ultimately, all of these domains and many more are deeply entangled with space, and that fact must be recognized and used.

## References

1. Beal, J.: Programming an amorphous computational medium. In: Unconventional Programming Paradigms International Workshop. Lecture Notes in Computer Science, vol. 3566, pp. 121–136. Springer Berlin (September 2004)
2. Beal, J.: Functional blueprints: an approach to modularity in grown systems. *Swarm Intelligence* pp. 1–25 (2011)
3. Beal, J., Bachrach, J.: Infrastructure for engineered emergence in sensor/actuator networks. *IEEE Intelligent Systems* 21, 10–19 (March/April 2006)
4. Beal, J., Bachrach, J.: Cells are plausible targets for high-level spatial languages. In: *Spatial Computing Workshop* (2008)
5. Beal, J., Lowell, J., Mozeika, A., Usbeck, K.: Using morphogenetic models to develop spatial structures. In: *Spatial Computing Workshop* (2011)
6. Beal, J., Lu, T., Weiss, R.: Automatic compilation from high-level biologically-oriented programming language to genetic regulatory networks. *PLoS ONE* 6(8), e22490 (08 2011)
7. Beal, J., Schantz, R.: A spatial computing approach to distributed algorithms. In: 45th Asilomar Conference on Signals, Systems, and Computers (November 2010)
8. Carroll, S.B.: *Endless Forms Most Beautiful: The New Science of Evo Devo and the Making of the Animal Kingdom*. W. W. Norton & Company, New York, NY, USA (2005)
9. Deneubourg, J., Goss, S.: Collective patterns and decision-making. *Ethology, Ecology & Evolution* 1, 295–311 (1989)
10. Mamei, M., Zambonelli, F., Leonardi, L.: Co-fields: an adaptive approach for motion coordination. Tech. Rep. 5-2002, University of Modena and Reggio Emilia (2002)
11. Perkins, C., Royer, E.: Ad-hoc on-demand distance vector routing. In: *Second IEEE Workshop on Mobile Computing Systems and Applications*. pp. 90–100 (February 1999)
12. Werfel, J.: *Anthills built to order: Automating construction with artificial swarms*. Ph.D. thesis, MIT, Cambridge, MA, USA (2006)