Biologically-Inspired Self-Assembly of Two-Dimensional Shapes Using Global-to-Local Compilation

Attila Kondacs, MIT Artificial Intelligence Laboratory, kondacs@mit.edu

In this paper, we present a programming language approach for the assembly of arbitrary two-dimensional shapes by decentralized, identically-programmed agents. Our system compiles a predetermined global shape into a program that instructs these agents to grow the shape via replication and location-based control mechanisms. In the global-to-local compilation phase, an input shape is decomposed into a network of covering-discs. The disc network parameterizes the agent program, a biologically-inspired framework allowing agents to amorphously produce the shape using replication and local interaction. Our system is robust to random agent failure, and regenerates in the event of region death.

1 Introduction

Biological cells assemble into complex structures with impressive robustness. They exhibit advanced global behaviors without centralized control or strict sequentiality of execution, despite random cell death or malfunction. In contrast, modern artificial systems are highly centralized and sequential, rendering them vulnerable to failure and encouraging the production of more complex, precise components. Our ability to embed millions of tiny chips with sensors [Abelson *et ai,* 1995; McLurkin, 1999], or program biological cells to serve as logic gates [Weiss *et al,* 1999; 1998J, marks a shift in technology to a reliance on cheaper, decentralized parts [Butera, 2001]. Traditional programming techniques are no longer sufficient for engineering systems, such as self-assembling nanostructures, to exhibit a robustness comparable to biological cells. How will we program large numbers of unreliable, locally-interacting parts to engage in coherent behaviors?

In this paper, we present a programming language approach to designing self-assembling systems [Nagpal, 2001; Coore, 1999]. We use morphogenesis and developmental biology [Wolpert, 1998; Bard, 1990] as motivation for organizing robust local behavior. However, unlike current approaches to designing emergent systems, the general principles are formalized as a programming language - with explicit primitives, means of combination, and means of abstraction - thus providing a framework for the design and analysis of self-organizing, spatially-controlled systems.

2 The Problem

Here, we apply the above approach to the synthesis of arbitrary two-dimensional shapes from tiny distributed computing units *amorphously.* An amorphous computing medium [Abelson *et ai,* 2000; 1999] is a system of irregularly placed, locally interacting, identically-programmed, asynchronous computing elements. We can model these computing units as *cells* with identical DNA. These cells have modest computing power, with limited ability to retain local state, and they may die. The communication radius is large relative to the radius of a cell, yet small compared to the size of the whole structure. Despite all these limitations, we would like the cells to exhibit highly complex collective behaviour: our goal is to compile a predetermined global shape into an amorphous cell program that instructs the cells to grow into the shape. In other words, starting from one cell, the cells continuously replicate into a configuration that geometrically approximates the input shape.

The global-to-local compilation proceeds in two phases: first, the input shape is decomposed into a semi-efficient packing of covering discs; second, adjacent discs are linked into a bidirectional network using a local, relative coordinate system ("reference points"). This choice of representation permits the structure to be produced in the growing phase amorphously by cells whose internal program relies on the recursive execution of only two key primitives: locating the reference points of discs, and growing into a disc. The cells use replication, messaging, and competition - mechanisms inspired by cell differentiation and morphogenesis [Lawrence, 1992; Wolpert, 1969] - to achieve these primitives robustly.

The system compiles and grows any connected two dimensional shape. The size of the compiled cell program is linear in the number of covering discs of the shape. The growth process is robust to random cell-death.

3 The Model

Our objects of discourse are locally-interacting, asynchronous computing units, here referred to as *cells.* Cells have a center location and a fixed radius r. A cell can replicate, and will place a given child between 2r and AT away

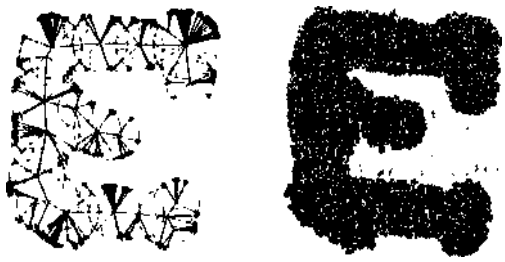

Figure 1: Compiled disc network with the spanning tree and the fully grown versions of a letter E shape.

from its center. Cells are identically-programmed, differing only by a limited amount of local state. They have limited computing power and are vulnerable to random death. They have no a priori knowledge of their positions or orientations. Our model for a cell substrate is analogous to living tissue: cells are tightly packed, cannot move around nor can they overlap.

3.1 Cell Primitives and Communication

A cell can execute five basic *low-level actions* actions. It may:

1. Change its internal state
2. Emit gradient messages
3. Hand messages to immediate neighbors
4. Reproduce
5. Die.

A *gradient message* has a specified amplitude and carries a numeric id. Amplitudes have an upper bound of 30 times the cell radius. Just as chemicals diffuse in fluid, a gradient message's strength decays with distance from the source [Nusslein-Volhard, 1996]. In our model, the decay is linear, and without noise. The amplitude of a given message is known by all cells, thus the strength of a received message will determine the distance from the source. In this way, cells can accomplish differentiation through *triangulation:* after receiving gradients and observing distances from at least three nearby non-collinear cells, a cell can determine its position relative to the source cells.

A *local message* carries a fitness value, a reference point id, and a state bit. Local messages are delivered only to *immediate neighbors,* or "touching" cells: cells within a fixed, short radius. There is always some local message exchange between immediate neighbors. Note that the lack of noise in messaging is a simplifying and unrealistic assumption in amorphous computing.

The low-level actions are combined to form the following *high-level actions:*

1. Grow a disc of cells
2. Activate or deactivate a cell as a competitor for a role
3. Turn reference point role on or off
4. Hand over a role to a neighbor
5. Triangulate another set of potential role holders

High-level actions are always associated with a *reference point role.* Reference points are cells that have been designated as "coordinates" of a local grid; these cells must exude gradient messages that allow nearby cells to triangulate their relative positions. Reference point roles can be *activated* and *deactivated* in any cell. Reference points will be discussed in great detail in sections 4 and 5. When such a role is activated, the cell enters a *local competition* for that role with other activated cells in its local neighborhood in an attempt to determine a unique role-holder for the given reference point. The details of the local competition are discussed in section 5.1.

The execution model of the cell is analogous to a Turing machine. The cell *transfer function* takes the internal state and the messages heard, maps these into a sequence of high-level cell actions, and translates these into a sequence of low-level cell actions. In each *cell-step,* the cell determines the next in the sequence of low level cell-actions, using the output of the transfer function, and then executes this action.

Asynchronous execution is achieved by using 5 threads, and each thread cycles through 70% of the cells in random order. A scheduler ensures that any cell that is more than 8 cell-steps ahead of the average number of cell-steps will not be chosen for execution in the next cycle, and that any cell that is more than 8 steps behind the average will certainly be included. The rest of the 70% of the cells are chosen randomly. The difference in the number of cell executions between any two cells cannot exceed 2 * (5 + 8) = 26 cell-steps.

4 Compilation

In our system the two-dimensional shape is compiled directly into the cell program. The complexity of the program grows linearly with the complexity of the input shape - that is the number of discs the compiler covers the shape with. This approach differs significantly from models based on cellular automata [Margolus, 1996], in which local rules are constructed empirically and creation of complex shapes would be intractable; and from those based on evolution [Forrest and Mitchell, 1993; Mitchell *et al,* 1994b; 1994a], in which the relationship between local and global behaviors is not well understood.

The compilation is a regular sequential (non-amorphous) procedure whose input is the pixel map of the 2 dimensional shape, and whose output is the amorphous cell program or transfer function. It operates in six simple stages. In the first stage, we select a *disc-covering* for the input shape. Second, we form a graph using our discs for our vertex set, in which two vertices form an edge if they intersect. A spanning tree is then selected from this graph; pairs of discs will be called *neighboring discs* if they are connected in the spanning tree.

Third, we locate the positions for reference points within each disc. In order to do this, we pick an arbitrary global orientation. Then we find, along with the *disc center,* the north, south, east and west *cardinal reference points* (i.e. points on the circumference of the discs that lie north, south, east and west of the the disc centers) in each disc. We proceed to locate the *intersection points* of neighboring discs in the spanning tree. Two problems can arise at this stage of the compi-

lation. (1) Consider a neighboring pair of spheres *(A) B).* If the intersection points of *A* and *B* are too close to each other the reference points in *A* and *B* used for triangulating reference points of *B* may be almost collinear, which makes the outcome of triangulation procedure ambiguous. To remedy this, if this problem arises we locate an extra reference point called the *temporary disc center.* (2) If the center of disc *B* is not inside the fully grown disc *A,* there may be no cells where B's center is supposed to be. In order to place cells at J3's center location in the growing phase, a sequence of temporary discs are grown, whose centers hand over the temporary disc center role called *line reference point* to a better positioned neighboring cell as soon as there is one. During the compilation phase the first line reference point is located. Sections 5.3 and 5.4 give a more detailed description of how the growth phase proceeds in these two cases.

Fourth, we designate *activating sets* and *deactivating sets* for each reference point. An activating or deactivating set is a set of reference point messages whose combined presence and/or absence can activate or deactivate another reference point role.

Next the (de)activating sets are converted to *boolean statements* that specify exactly when a cell should consider activating or deactivating a reference point role. The boolean language is the heart of the cell transfer function. It contains only one primitive: a message from a reference point can be heard. The language has the usual *And, Or* and *Not* constructors, as well as two additional constructors: *AtJeast(k, set)* and *At-most(k, set).* These express that at least (or at most) *k* of the statements in *set* must (may) be true. The primitives are combined, using the constructors, to form compound statements, thus forming a concise description of the complex geometric and logic relations existing among reference points. The boolean statements would, for example, compactly describe the following simplified scenario: A cell should consider competing for the north cardinal point role in disc *B* if: (it does not hear other north cardinal point messages from *B) And* (it hears a message from B's center) *And* (it hears at least three other cardinal points of disc *B).* A more complete example boolean statement is listed in section 5.2.

In the last, sixth stage of the compilation, we calculate the ideal message strengths associated with messages in the boolean statements. The latter distances, together with the compound boolean statements, make up the transfer function.

5 Growing Phase

Each cell executes the same cell program.The compilation procedure outputs this program from a given connected two-dimensional shape. The program consists of the transfer function and an interpreter. The cell maps the combination of its internal state and received messages to a sequence of high-level actions.The cell proceeds to translate the high-level actions into a sequence of low-level actions that determine (a) which messages a cell must prepare to send, (b) whether the cell should produce a child, (c) the internal state it must assume during the next cell-step. While over a long period of time all of the cells' average running speeds are the same, the cell-steps are not synchronized, and the expected duration of a step varies among cells. Collectively, cells execute two major operations recursively: (a) growing a disc via cell replication, and (b) triangulating target cells that will assume reference point roles.

Let's call line messages, and messages from disc centers and temporary centers, *grow messages.* In every cell-step of its existence, the cell listens for grow messages. If it detects one, it will attempt to reproduce, placing a cell in the surrounding empty space. It will suspend replication after 20 unsuccessful trials until there is a new opening in the surrounding area, i.e. until a nearby ceil dies. Thus the gap in the shape left by a dead cell is filled by the offspring of the expired cell's neighbors. In case the cell does not hear any grow messages for 50 steps, the cell dies and is removed from the space.

In each cell-step the cell processes the gradient messages it hears, let us call this its *in-message set.* It selects the statements from the set of boolean statements stored in the transfer function that evaluate to true. To evaluate a boolean statement the cell should match its in-messages against the list of messages that are required to be present or missing by the boolean. Each of the true boolean statements asserts that a reference point role should activate or deactivate in this cell. The cell makes the final decision of whether to (de)activate as a competitor for the reference point role based on the proximity of the perceived strengths of the messages (that are required to be heard by the boolean) to the ideal message strengths stored in the transfer function. In the next cell-step the activated cell enters (and the deactivated cell exits) the competition for holding this role with other activated cells in its neighborhood.

5.1 Local Competition for a Reference Point Role

Activated (or competing) cells keep track of 4 things: whether they have settled into a steady *winner* or *loser* state from the *competing* state; their own *fitness value;* the best received fitness value, A competing cell will send local messages to its neighbors with (a) the reference point id for which it is competing, (b) a state bit showing whether it has stabilized, and (c) the best fitness value it observes (possibly its own). The cell's active neighbors will keep doing the same. Thus if its fitness value is better than anyone else's, this value will spread accross its competing neighbors and its neighbors' neigbors, and so forth. If a cell has had a better fitness score than any of the fitness values it sees in the local messages for 40 steps, it stabilizes as a winner. This triggers all its competing neighbors to settle (end the competition) as losers, which will induce all their competing neighbors (and in turn their competing neighbors neighbors, and so on) to stabilize as losers. Eventually, all the cells competing for this particular reference point role that can be reached via a sequence of immediate active neighbor relationships from the winner cell stabilize as losers. Once a cell has stabilized as a winner, it becomes the holder of the particular reference point role for which the cells have competed. It instantly sends out the corresponding reference point gradient message. In the event that there is a topologically disconnected pocket of cells still competing for the same role, the presence of the reference point gradient will supress any further competition

with lower leading score. If the gradient message or the winner disappears the passive loser cells resume competition for the reference point role until a new leader stabilizes. Thus the competition never ends, but reaches equilibrium, and resumes competition as soon as the equilibrium is disturbed. The 40 step wait before a cell stabilizes ensures that the local messages with the maximum score have time to spread through the competing cells in spite of the asynchronous execution of cell-steps.

If various competing cells die, local competition will, with high probability, finish succesfully. We may consider the following cases. If a non-leading cell dies its loss does not disturb the outcome of the competition. If a non-stabilized leading cell dies, the neighboring cells will sense the absence of the local message with the winner state bit and will start broadcasting a special local message containing the reference point role. This will spread through the competing cells and restart the local competition for this particular role. It is unlikely that all the cells surrounding the now missing leading cell die before they could trigger a new competition. However, if this does occur, the rest of the competing cells that have stabilized in loser states will realize that the end of the competition for this role has not been followed by a corresponding gradient message from the winner in 50 cell-steps, and will automatically resume competition for this role.

5.2 Triangulating the Reference Points of a Disc

Assume that the fully grown disc *A* has a neighbor, disc *B,* that has not been grown yet. Let's denote .4's center reference point *A.C,* cardinal reference points *A.N, A.S, A.E, A.W,* (north, south, east and west respectively) and the intersection points with disc *B, ABJI* and *AB.I2.* I will describe a possible sequence of reference point activation path that will lead to a fully grown *B* disc with its reference points in place.

An outline is as follows: first triangulate J5's center, *B.C,* then grow disc *B,* proceed to triangulate the cardinal points of *B.* Once all the cardinal points of *B* have stabilized, triangulate the intersection points of *B* with its neighboring discs simultanously, and recursively do the same with *B's* yet ungrown neighbors.

In this section, we will consider the ideal case: when *A* contains *B.C.* Using *A's* cardinal point gradient messages *B.C* can be located easily. In this case all cardinal points of *B* that lie inside *A* are triangulated using reference points of *A.* We need to consider three cases. (1) Assume that there are at least two such cardinal points of *B* that lie in *A.* After these have stabilized, they, along with *B.C* are used to triangulate the remaining cardinals of *B.* (2) If there is only one cardinal, say *B.E,* in *A* then after this one has stabilized, first the opposite direction cardinal, *B.W* is located: *B.E* and *B.C* are used to find the cell in *B* that is furthest away from *B.E,* but is still in *B.* Next the cardinal point that is closer to the intersection points *ABJI* and *AB.I2* is triangulated, using *ABJI, AB.I2, B.E, B.W* and *B.C.* Finally, the last remaining cardinal point of *B* is triangulated using the 3 other cardinals and the center of *B.* (3) If there is no cardinal in *A* then *ABJI, AB.I2,* and *B.C* are used to locate the cardinals of *B* that are closest to *A.C,* then the opposite cardinals using the same procedure as described above.

The scheme described so far is the most preferred method of growth, i.e. the one followed when nothing is damaged. However, in the event that cells die unexpectedly, or if some of the cardinal points are covered by other complete discs, for example, the order may be different. This complex growth order is determined by a sequence of boolean statements in the transfer function of the cell program. Consider the following simplified statement for activating *B.N:*

```
And[Not[B.N]\
    B.C]
    Or[At.least 3 (cardinalsAn.disc B);
       (opposite-cardinal-point-in-disc North B);
       Or[map
```

$$(function\ disc \rightarrow And[intersection\_points\_of\_discs\ B\ disc] (neighboring\_discs\_of\_disc\ B))]]$$

The cell will check if the statement evaluates to true on the presently observed in-message set. It finds the first boolean in the normal form that evaluates to true, which we will call the *satisfied boolean.* Thus, in this example, it will most prefer to use, along with *B.C,* the 3 other cardinals of *B* to locate *B.N,* then *B.S,* then the two intersection points with *A,* or some other neighbors. If the first, most preferred boolean is not satisfied (that is, the messages that would satisfy it are not heard), perhaps because some cells died, it will try to use the second most preferred boolean, then the third, and so on. If there is no satisfied boolean, the cell will wait, thus the triangulation is unaffected by asynchronous execution of cell cell-steps.

5.3 Handing down a Reference Point Role

If *B.C* is outside disc *A,* the cells in *A* proceed to triangulate the line reference point *B.LI* midway between *AB.II* and *AB.I2* on the circumference of *A.* As soon as *B.LI* has stabilized, it starts exuding a temporary disc center message with range equal to the distance between *B.LI* and *AB.II.* This induces cells in the range to reproduce. Cells around *B.LI* constantly communicate their fitness score for the role of B.C via local messages to the cell holding *B.LI.* Once a cell with a better fitness than that of *B.LI* has showed up, *B.LI* role is replaced by a temporary disc center role, *B.T1.* At the same time the cell holding this role sends a local message to the better scoring neighbor that turns on the role *B.L2* in it. The latter cycle continues until a cell can find no immediate neighbor that scores better than itself for 50 cell-steps. At this stage the *B.L* role is turned off and the *B.C* role stabilizes in its place, the appropriate center gradient message is sent out, which in turn turns off all temporary centers in *B.* Note that during this passing on of reference point roles the local competition procedure plays no part other than stabilizing 5.L1. The procedure is still robust because if a cell with a *B.L* role dies, and as a result, there are no more *B.L* (or *B.C)* messages observed by the cell, then the temporary center *B.Ti* that hears no larger id *B.Tj* temporary center message will turn into a *B.Li,* and the handing down procedure restarts from that point.

The first disc is special as it cannot be grown by any other disc. The latter reference point role handing over procedure

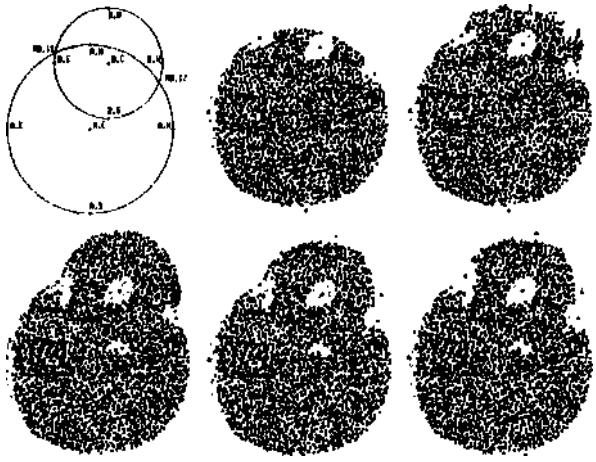

Figure 2 : 1 : Center, cardinal and intersection reference points of discs A and B. 2-6: Growing phase. Grey cells are stabilized loser competitors, white cells are still competing for a role. Black cells surrounded by white or grey cells are leading competitors or stabilized reference points. Local competition for the intersection points (AB.I1, AB.I2) and B's center (B.C) (2); growing disc B (3); local competition for south (B.S) and east (B.E) (4) and north (B.N) and west (B.W) (5) cardinal reference point roles in disc B; and the grown shape (6).

is used to pick the first, north, and the second, east cardinal points in the first disc. The orientation of the first north cardinal is chosen randomly and the orientation of the first east cardinal point will randomly choose between the east and west directions relative to the already picked north. On the figures these are externally forced to be up and left.

5.4 Temporary Discs

If the distance between *AB.I1* and *AB.I2* is too small relative to the radius of *A* or the cell size, or either of the radii are too small relative to the other one or to the cell size, then triangulation of the cardinals may be ambiguous. In these cases, if simple geometric checks confirm the ambiguity, a temporary helper disc, T, is grown. T's center, *T.C* is at the midpoint of the arc between *AB.II* and *AB.I2* on *A's* circumference. After *T.C* has stabilized, if *T* does not cover *B* completely, T's intersections with JB, *TBJI* and *TB.I2* are triangulated. The diameter of *T* is chosen such that the distance between *TBJI* and *TB.I2* is large enough to avoid the original ambiguity. If T contains *B, TBJI* and *TBJ2* are picked on the circumference of *T.* In either case *TBJI* and *TB.I2* are used in place of *ABJI* and *ABJ2* in all the above discussed procedures. Once at least 2 cardinal points and the center of *B* have stabilized, *T.C* is turned off, and any cells that fall outside the ranges of all the presently existing grow messages die in 50 cell-steps.

6 Overall Robustness and Simulation Results

One of the main goals of this research is to explore how the robustness of computing can be improved using spatial control mechanisms. We ask ourselves, to what types of failures, and to what degree of failure, is our system robust? Furthermore, what imperfections can we introduce that will expose vulnerabilitcs in our system?

Let us consider the robustness of our parallel system in the face of random cell death, that is, if we continuously exterminate cells at random with a certain frequency. Whether the shape can effectively continue to grow depends on the rate at which we cause cells to die. Certainly, the rate of cell death can not be faster than the rate of regeneration.

All high level actions of cells, including local competition, role handing, triangulation, disc creation, can function in spite of a high death rate. The robustness of the competition procedure against cell death was discussed in section 5.1. Let us discuss some of the other various aspects of how our system can accomodate cell death.

If a cell with a cardinal, intersection point, disc center or temporary center reference point role dies, its missing gradient message will trigger a new local competition among all the cells competing for the role, and this will eventually lead to the selection of a new reference point holder cell. If a line-reference point dies, the role-handing down procedure will backtrack to the last still available line-reference point. If the first disc's center dies before the first disc's cardinal reference points stabilize a neighboring cell will resume the first disc center role.

If multiple reference points vanish simultanously, it is possible that an entire set of cells competing for a reference point role will stop competing for the role. This is very unlikely for two reasons: first, deactivation will take place only after the boolean statement of the satisfying set in the deactivating set has evaluated true for 100 consecutive cell-steps. Therefore, some reference point role holders and thus their gradient messages will be missing for 100 steps. In this many steps, however, they will be replaced by new role holder cells, which will turn the booleans in the deactivating set false before they can take effect. Second, each reference point role has multiple activating sets, each of which must be missing for the role to be deactivated.

In case such a deactivation occurs to a disc center, it will cause the cells only in this disc to die, after a 50 cell-step wait period. Let's call the discs that can regenerate all their neighbors *regenerating discs.* The compilation is designed so that any disc with radius larger than 8 times the cell radius is a regenerating disc. Assume a whole disc of cells has been destroyed because all the cells competing for the center reference point have died. As long as at least one of the regenerating disc neighbors of the missing disc is relatively unharmed, the disc will regrow under normal growth procedures. On the other hand, to make regrowth impossible, all disc centers and at least two of the cardinals of all regenerating disc would have to die, all in a very short amount of time.

Overall, our system is quite robust to random cell death. Consider the following experiment. After there exist at least 50 cells, in each cell-step in which a new cell is created, a

cell is randomly chosen and with 50% probability it dies. For every *2k* cells that are created, on average *k* have died, so about *k* cells are left. That is, the number of live and dead cells will be approximately the same. Several runs of this experiment have shown that, under these circumstances, the cells will still correctly approximate the compiled shape.

The system has been designed so that the shape can recover from two other types of failure. The first type is sudden massive cell death, that is, death to many cells at random locations accross the shape at once. The second is regional cell death where a contiguous patch of cells die at once. We are still investigating how well the system can recover from these failures.

7 Future Work

As stated before, lack of noise in messaging is unrealistic; a natural next step would be to deal with this type of failure. Furthermore, to match the robustness of live tissue, we would have to get rid of the assumptions of uniform cell size, immobile cells, linear, reliable and non-changing gradient decay function (the latter for real chemical gradients is exponential with exponent multiplier varying with temperature!). The present approach could prove interesting for the areas of reconfigurable robots and autonomous agents [Butler *et al,* 2002]. Finally, it would be worthwile to translate the current 2 dimensional program into a 3D shape growing implementation.

8 Acknowledgements

This research was supported by a National Science Foundation grant on Quantum and Biologically Inspired Computing (QuBIC) from the Division of Experimental and Integrative Activities, contract EIA-0130391. The author would like to thank Jacob Abernethy, Catherine Chang, Hal Abelson, Radhika Nagpal and Gerald Sussman for their help.

References

[Abelson *et al,* 1995] H. Abelson, A. Berlin, N. Cohen, L. Fogel, C. Ho, M. Horowitz, J. How, T. Knight, R. Newton, and K. Pister. Distributed information systems for MEMS. *ISAT (Information Science and Technology Study Group) Study,* 1995.

[Abelson *et al,* 1999] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman, and R. Weiss. Amorphous Computing. *White paper,* 1999. http://www.swiss.ai.mit.edu/projects/amorphous/.

[Abelson^*al,* 2000] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman, and R. Weiss. Amorphous computing. *Communications of the ACM,* 43(5), May 2000.

[Bard, 1990] J. Bard. *Morphogenesis.* Cambridge University Press, U.K., 1990.

[Butera, 2001] W. Butera. *Programming a Paintable Computer.* PhD thesis, MIT Media Lab, 2001.

[Butler e/a/., 2002] Zack Butler, Satoshi Murata, and Daniela Rus. Distributed replication algorithms for self-reconfigurable modular robots. *Distributed Autonomous Robotics Systems,* 5, 2002.

[Coore, 1999] D. Coore. *Botanical Computing: A Developmental Approach to Generating Interconnect Topologies on an Amorphous Computer.* PhD thesis, MIT, Dept of Electrical Eng. and Computer Science, February 1999.

[Forrest and Mitchell, 1993] S. Forrest and M. Mitchell. What makes a problem hard for a genetic algorithm? *Machine Learning,* 13:285-319, 1993.

[Lawrence, 1992] P. A. Lawrence. *The Making of a Fly: the Genetics of Animal Design.* Blackwell Science, Oxford, U.K., 1992.

[Margolus, 1996] N. Margolus. Cam-8: A computer architecture based on cellular automata. *Pattern Formation and Lattice-Gas Automata, American Mathematical Society,* pages 167-187, 1996.

[McLurkin, 1999] James McLurkin. Algorithms for distributed sensor networks. Master's thesis, University of California, Berkeley, December 1999.

[Mitchell *etal,* 1994a] M. Mitchell, J. Crutchficld, and P. Hraber. Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D,* 75:361—391, 1994.

[Mitchell *etal,* 1994b] M. Mitchell, J. Holland, and S. Forrest. When will a genetic algorithm outperform hill climbing? *Advances in Neural Information Processing Systems,* pages 285-319, 1994.

[Nagpal, 2001] R. Nagpal. *Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics.* PhD thesis, MIT, Dept of Electrical Engineering and Computer Science, June 2001.

[Nusslein-Volhard, 1996] C. Nusslein-Volhard. Gradients that organize embryo development. *Scientific American,* August 1996.

[Weiss *etal,* 1998] R. Weiss, G. Homsy, and R. Nagpal. Programming biological cells. In *8th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '98), Wild and Crazy Ideas Session,* October 1998.

[Weiss *etal,* 1999] R. Weiss, G. Homsy, and T. Knight. Toward in vivo digital circuits. In *Dimacs Workshop on Evolution as Computation,* January 1999.

[Wolpert, 1969] L. Wolpert. Positional information and the spatial pattern of cellular differentiation. *Journal of Theoretical Biology, 25:1-41,* 1969.

[Wolpert, 1998] L. Wolpert. *Principles of Development.* Oxford University Press, U.K., 1998.