Compilation and Biologically-Inspired Self-Assembly of Two-Dimensional Shapes

Rebecca Bloom, Catherine Chang, Attila Kondacs *

Abstract

In this paper, we present a programming language approach for assembling an arbitrary two-dimensional shape from decentralized, identically-programmed agents. Our system compiles a predetermined global shape into a program that allows the agents to grow the shape via replication, using only location-based control mechanisms. In the global-to-local compilation phase, an input shape is decomposed into a network of efficient covering-spheres. The sphere network parameterizes the agent program, a biologically-inspired framework allowing cells to produce the shape using replication and local interactions. 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 sensors on a chip [3, 19, 14], or program biological cells [21, 20, 22] to serve as logic gates, marks a shift in technology to a reliance on cheaper, decentralized parts [6, 5]. Traditional programming techniques are no longer sufficient for engineering systems, such as self-assembling nanostructures, to display 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 [17, 10]. We use morphogenesis and developmental biology [24, 4] 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

*contact: Attila Kondacs, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02114, kondacs@mit.edu

– 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. These computing units, or *cells*, are identically-programmed and decentralized, with the ability to engage in only limited, local communication. The goal is to compile a predetermined global shape into this program for individual cells that allows them to produce ("grow") the shape via replication. In other words, the cells replicate such that the resulting mass of cells approximates the input shape.

In the global-to-local compilation phase, we represent input shapes as a network of covering spheres. This choice of representation permits the structure to be produced amorphously [2, 1] by cells whose internal program relies on the recursive execution of only two key primitives: growing into a sphere, and locating the centers of adjacent spheres. The compilation proceeds in two phases: first, the input shape is decomposed into a semi-efficient packing of covering spheres; second, adjacent spheres are linked into a bidirectional network using a local, relative coordinate system ("reference points"). Locally, the cells use replication, messaging, and competition – mechanisms inspired by cell differentiation and morphogenesis [12, 23] – to achieve these primitives robustly.

Attributes of this system include scalability, robustness, and the ability for self-repair. Just as a starfish can regenerate its entire body from part of a limb, our system can self-repair in the event of agent death; the sphere-network representation allows the structure to be grown starting from any sphere, and every cell contains all necessary information for reproducing the missing structure.

3 The Model

The growing shape is comprised of many locally-interacting, asynchronous computing units, here referred to as *cells*. The cells replicate, placing themselves in a configuration that approximates the target global shape. Cells are identically-programmed, differing only in a small amount of local state. They have limited computing power and are vulnerable to random malfunction or death.

Our model for a cell substrate is analogous to living tissue; the cells cannot overlap or move around in the substrate, and are closely packed (within two cell radii from one another). The space between cells may be likened to a liquid that is capable of conducting signals between cells.

3.1 Communication between Cells

There are three different ways that cells may communicate information:

- Cells can send out gradient messages.

- Cells can hand messages directly to their immediate neighbors (cells within a fixed, shortradius)
- Cells can inherit their initial, internal state from parent cells.

Just as chemicals diffuse in fluid, a gradient message's strength decays as distance from the source cell increases. Therefore, the strength of a received message encodes information about the receiver's distance from the source [18]. In this way, cells can accomplish differentiation through *triangulation*: after receiving a gradient and observing relative distances from at least three nearby non-colinear cells, it can determine its approximate position relative to the source cells.

3.2 Cell Primitives

At the lowest level, a cell can execute five basic actions. It may:

1. Change its internal state
2. Exude gradient messages
3. Hand messages to immediate neighbors
4. Reproduce (divide), placing the child cell in a random position within a fixed radius around the mother cell.
5. Die. After death, a cell is removed from the substrate.

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

1. Grow a sphere
2. Compete for a role
3. Activate or deactive a cell as a role-holder
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. In addition, a cell may hold multiple roles, as each sphere determined by the compilation phase possesses its own set of reference points. The purpose of a reference point will be explained more thoroughly in the next section.

A cell may also assume the role of sphere center, a special type of reference point. A sphere center induces the replication of cells outward to a certain radius, thereby causing a sphere to form around itself.

4 Cell Program

Throughout the shape formation, each cell executes the same program (the "cell program"). The shape compilation procedure parameterizes this program from any simply-connected two-dimensional shape. The cell program is a function; in each time-step, it maps a cell's internal state and received messages to a sequence of high-level actions that must be performed in this time-step. The high-level actions are composed of low-level actions which determine (a) which messages a cell must prepare to send, and (b) the internal state it must assume during the next time-step. The cell time-steps are not synchronized, and the expected duration of a step varies among cells; however, the expected number of steps executed by all cells over a long time period is the same. The cell program does not rely on sequential execution of steps among other cells.

Because the global input shape is represented as a network of spheres, the cell program for growing the shape is based on two major operations: (a) growing a sphere via cell replication, and (b) triangulating target cells which will become reference points and centers of adjacent spheres. Locating reference points is a means for establishing the correct relative placement of the spheres comprising the global shape. As cells begin to form spheres, four cells in each sphere designate themselves as cardinal reference points— that is, they mark the north, south, east, and west poles of a sphere. Cells with reference-point roles exude distinct gradient messages, from which a nearby cell may extract information about its relative orientation. This, in effect, establishes a local coordinate system that aids cells in determining where the next sphere must be grown.

How do cells determine who should assume a reference point role, or who should become the center of a new sphere? The triangulation question is settled by a competition mechanism. Compilation has specified an ideal ratio of gradient strenghths that a cell should receive in order to be the optimal holder of a role. Cells that perceive that they are receiving a set of sufficiently strong gradients enter a competition for this role. Cells communicate their fitness to their immediate competing neighbors; eventually, the cell with the greatest fitness is selected as a leader. If this cell remains the leader for a certain number of time-steps, it stabilizes as the holder of the reference point role. Once a cell stabilizes, the other competing cells enter a passive state for the duration of the leader's existence. Failure of the stable leader signals the passive cells to resume active competition until a new leader stabilizes.

With the cardinal reference points in place, the center of a new sphere may be located in one of two ways. If this new sphere's (Sphere B's) center should be within the first sphere (Sphere A), then B's center may be triangulated from reference points. If B's center is to be placed outside of A, then a cell in sphere A — that which is closest to the desired position of B's center — is activated as a temporary sphere center, thereby growing a sphere around itself. As the temporary sphere grows, new cells inevitably appear that are even closer to the target location of B's center. The temporary center triggers this best-positioned cell to activate itself as a sphere center, while the second-best cell is deactivated. As the process continues, the role of the sphere center is pushed closer and closer to its optimal location.

graphicx

After the optimal center of sphere B stabilizes, it grows sphere B around it. The

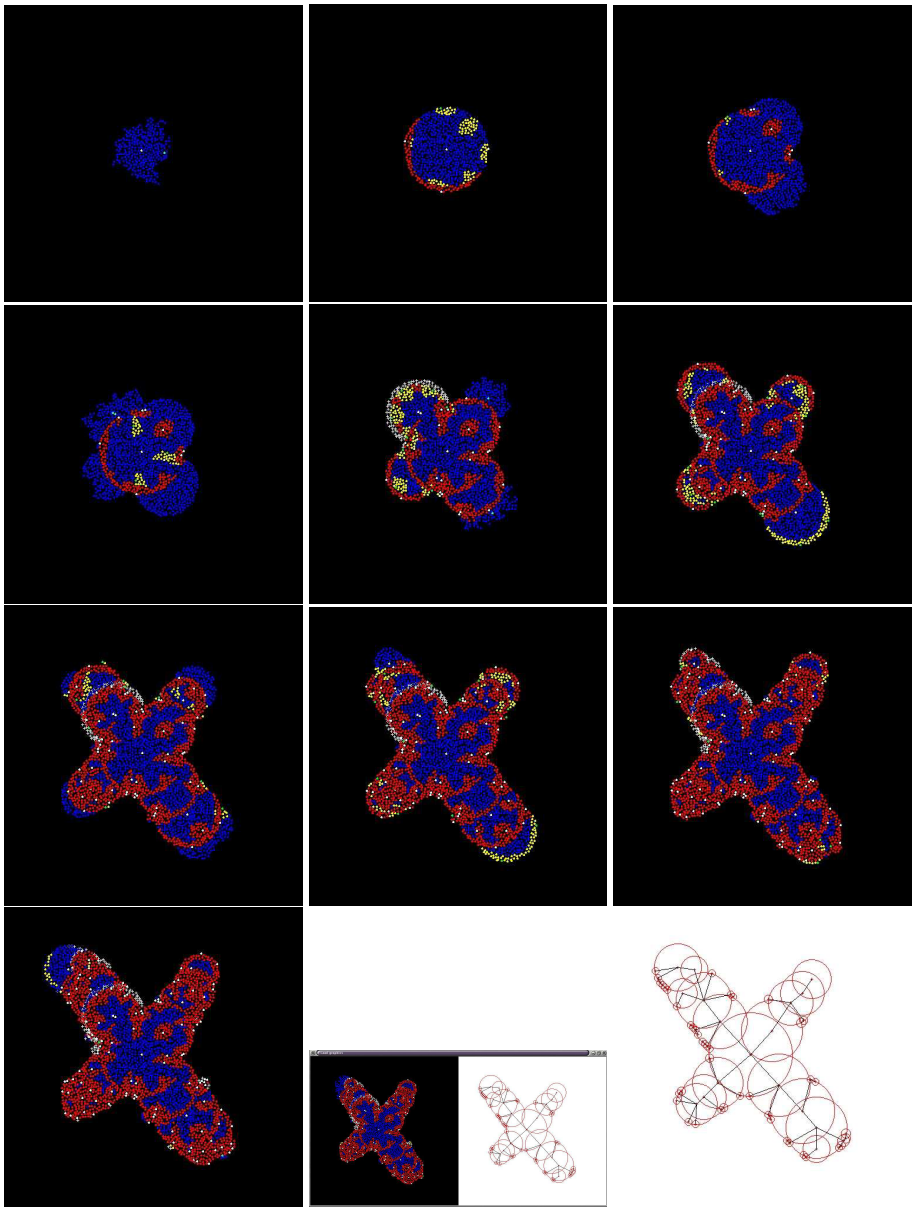

Figure 1: Stages of growing a cross shape. Yellow signifies competing cells, green cells are leading competitors, red cells are those who have settled and lost the competition, white cells are reference points (winners of the competition), and a white cross marks dead cells. The last figure shows the target input shape and its sphere-covering.

cells of B proceed to triangulate B's cardinal reference points. The location of the first cardinal reference point is specified in terms of gradient ratios from the sphere center of B, and the intersection points between spheres A and B (which also exude gradients). If B's center or any of its cardinals are located inside of sphere A, the cardinal reference points of sphere A are also used. After this first cardinal reference point of B is established, it is used – along with the previously-mentioned points – to triangulate the remainder of B's cardinal reference points. At this point, cells which are members of temporary spheres residing outside the intersection of spheres A and B are directed to decay and, eventually, die.

The remainder of the target spheres are produced by recursively applying the above procedure to neighbor spheres of sphere B. It is the compilation phase, described in more detail below, that designates which adjacent spheres are to become neighbors in the network. A sphere grows all of its neighboring spheres simultaneously.

5 Compilation

In our system, the cell program is compiled directly from the two-dimensional shape. This differs significantly from approaches based on both cellular automata [13], in which local rules are constructed empirically, and evolutionary approaches [11, 16, 15], where the relationship between local and global behaviors is not well understood.

The compilation operates in five simple stages. In the first stage, we select a relatively efficient sphere-covering for the input shape. Second, we create a graph in which the covering spheres are the nodes and intersecting spheres form edges. A spanning tree is then selected from the graph; pairs of spheres that are connected in the spanning tree will be considered *neighboring spheres* throughout the remainder of the compilation and amorphous growth. Third, we locate the positions for reference points within the spheres, and designate *activating sets* and *deactivating sets* for each reference point. An activating or deactivating set is defined as a set of reference point messages whose combined presence or absence can activate or deactivate another reference point role. Next the (de)activating sets are converted to boolean-like statements that specify exactly when a cell should consider activating or deactivating a reference point role. The booleans statements would, for example, compactly describe the following scenario:

A cell should activate as the "north cardinal point" in a sphere if: (a) it does not hear other north cardinal point messages, and (b) it does hear a message from the center and three other cardinal points of a sphere, or if (a) it is inside sphere B, which is a neighboring sphere of sphere A, and (b) it fails to hear another north cardinal point message from sphere B, and (c) it hears all four cardinal points and the center of sphere A.

The boolean-like language is the heart of the compilation. 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: $At\_least(k, set)$ 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 relations existing among reference points.

In the last stage of the compilation, we calculate the ideal message strength ranges associated with messages in the boolean statements. The latter distances, together with the compound boolean statements, make up the cell program. The cell, depending on its internal state and received messages, chooses a subset of its messages and makes the final decision of whether to activate a reference point role based on the proximity of the received message strengths to the ideal message strength ranges.

Each reference point is assigned several activating sets. Several sets make it possible to arrange reference points in the form of a multidirectional grid in which each reference point role can be brought to life by various methods. This increases the system's ability to regenerate and the robustness considerably. The growth process can start from any sphere center in any part of the shape, and similarly, the growing shape can always regenerate the whole structure as long as at least one sphere stays relatively unharmed.

6 Robustness and Simulation Results

One of the main goals of this research is to investigate how the robustness of our system can be improved using spatial control mechanisms. More specifically, we would like to observe the robustness of the shape growth in the events of random cell death, the death of large regions of cells, and unreliable messaging.

6.1 Random Cell Death

We have implemented several mechanisms for helping the system to recover from random cell death. The first method of recovery is based on awareness of neighboring cells (cells within a fixed, short distance from one another - the "touching cells"). In the process of growing a sphere, every cell that hears a growth message from the sphere center attempts to reproduce and place daughter cells randomly around itself within a ring. This cell only succeeds in reproducing if there is room at the chosen location for another neighboring cell. Cells are aware of their neighbors, and they resume replication when a neighbor disappears.

The second method of recovery employs role-competition among cells. If a cell holding a reference point role disappears, then the competition for this role among the cells in the local neighborhood resumes. With very high probability, a neighbor's descendant will fill the space left by the dead cell in time to become the new role-holder.

6.2 Regional Cell Death

The above mechanims cause the structure to recover in the event that all members of the local ccompetition for a role perish. In this case, surrounding cells will simply replicate until the gap is filled. These descendants, because they occupy a region close to the optimal position of the role-holder, naturally become the new competitors for the role.

The system is robust enough to recover from widespread cell death. If the cells in large parts of the structure are wiped out, the system may rely on its original growing

procedure to regenerate. This is possible because the reference points form a multidirectional grid in which each reference point can be regenerated from multiple activating sets, and each sphere can be regrown from multiple neighbors. Therefore, any part of the structure, that has at least a spherecenter and enough reference points to orient itself, can regenerate the rest.

6.3 Unreliable Messaging

Our system is robust in certain cases of unreliable messaging — for instance, in failure of message delivery during the local competition, or failure to hand on a role to a neighboring cell. This is because messages in the local competition can reach cells via multiple paths, and because handing a role to the wrong cell every once in a while does not alter the final destination of a role.

The system can tolerate some erroneous messages — slightly distorted message strength, for example, causes distortions in the resulting shape. If gradient messages deviate largely from their ideal strenghts, or are not sent at all, then the system will either (a) fail to grow the affected region, or (b) attempt to find an alternate, unaffected path of activating sets in the compile reference-point grid and therefore still succeed in growing. In addition, the system is unaffected by the timing of message delivery.

7 Future Work

There are numerous areas open to future exploration. The most obvious direction for future work is to translate the algorithm for amorphously growing two-dimensional shapes into one which grows three-dimensional shapes.With modifications to the underlying geometry, the current program should translate into a 3D implementation rather straightforwardly.

In addition, we could make the present 2D shape growing more robust by replacing the present one-by-one activation of reference point roles (mozaic-creature style [24]) with increasing specialization and splitting of roles. This can be implemented by handing down a set of roles to a certain region but activating only the ones *within* the region that matter at the particular stage of growth. Reconfigurable robots and autonomous agents are also an area of interest [7, 8, 9]. Finally, a significant step would be to create reversible amorphous shape compilation and growing — that is, decentralized shape learning and recognition.

References

[1] 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/.

[2] 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.

[3] 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.

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

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

[6] W. Butera and V. Bove. Literally embedded processors. In *Proc. of SPIE Media Processors*, 2001.

[7] Z. Butler, S. Byrnes, and D. Rus. Distributed motion planning for modular robots with unit-compressible modules. *Proceedings of the Intl Conf. on Intelligent Robots and Systems*, 2001.

[8] Z. Butler, K. Kotay, D. Rus, and K. Tomita. Generic decentralized control for a class of self-reconfigurable robots. *Proceedings of the IEEE Intl Conf on Robotics and Automation (ICRA)*, 2002.

[9] Z. Butler, S. Murata, and D. Rus. Distributed replication algorithms for self-reconfigurable modular robots. *Distributed Autonomous Robotics Systems*, 5, 2002.

[10] 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, Feb. 1999.

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

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

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

[14] J. McLurkin. Algorithms for distributed sensor networks. Master's thesis, University of California, Berkeley, Dec. 1999.

[15] M. Mitchell, J. Crutchfield, and P. Hraber. Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D*, 75:361–391, 1994.

[16] 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.

[17] 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.

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

[19] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5), May 1999.

[20] R. Weiss. Programming colonies of biological cells. *PhD Thesis Proposal, MIT, Department of Electrical Engineering and Computer Science*, Jan. 1999.

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

[22] 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*, Oct. 1998.

[23] L. Wolpert. Positional information and the spatial pattern of cellular differentiation. *Journal of Theoretical Biology*, 25:1–47, 1969.

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