Programming Methodology for Biologically-Inspired Self-Assembling Systems

Radhika Nagpal, Attila Kondacs, Catherine Chang
MIT Technology Square, NE43-432
Cambridge MA 02114
contact: radhi@mit.edu

Abstract

We present a programming methodology for self-assembling complex structures from vast numbers of locally-interacting identically-programmed agents, using techniques inspired by developmental biology. We demonstrate this approach through two examples: shape formation on a reconfigurable sheet, and self-assembling two dimensional structures through replication. In each case, the desired global shape is specified using an abstract geometry-based language, and the agent program is *directly compiled* from the global specification. The resulting self-assembly processes are versatile, analyzable, and reliable in the face of random agent distributions, varying numbers of agents and random death. This approach takes advantage of traditional computer science techniques for managing complexity, while relying on biological models for achieving robustness at the local level.

Introduction

Emerging technologies are making possible novel applications that integrate computation into the environment: smart materials, self-reconfiguring robots, self-assembling nanostructures. We are faced with the challenge of achieving coherent and robust behavior from the interactions of multitudes of elements and their interactions with the environment. These new environments fundamentally stress the limits of our current engineering and programming techniques, which depend on precision parts and strongly regulated environments to achieve fault-tolerance. By contrast, the precision and reliability of embryogenesis, in the face of unreliable cells, variations in cell numbers, and changes in the environment, is enough to make any engineer green with envy. We would like to build novel applications from these technologies that achieve the kind of complexity and reliability that biological systems achieve. This requires solving two problems:

- How do we take a *particular* global goal and translate it into local interactions between identically-programmed agents?
- How do we engineer robust and predictable behavior from vast numbers of unreliable parts?

Our approach is to use morphogenesis and developmental biology as inspiration for robust mechanisms and general principles for organizing local behavior. However unlike current approaches to emergent systems, the general principles are formalized as *programming languages* — with explicit primitives, means of combination, and means of abstraction — thus providing a framework for the design and analysis of self-organizing systems.

In this paper we present two examples of applying this methodology to the design of self-assembling systems. The first system provides a language for specifying shape formation on an intelligent reconfigurable sheet composed of thousands of identically-programmed but locally-interacting flexible agents (Nagpal 2001; 2002). The system uses a novel approach: the desired global shape is specified at an "abstract" level as a folding construction on a continuous sheet of paper, which is then *automatically compiled* to produce the program run by the identically-programmed agents. All agent behavior is constructed from a set of five general purpose robust primitives, inspired by epithelial cell morphogenesis and cell differentiation in multicellular organisms such as the *Drosophila*. The global-to-local compilation is achieved by composing these primitive building blocks in a principled way, using a set of geometry axioms taken from paper-folding mathematics. The resulting process is versatile and reliable in the face of random agent distributions, varying numbers of agents and random agent death, without relying on global coordinates, a global clock, or centralized control. The process also exhibits interesting biological traits; for example, it is *scale-independent* — the shape scales to the number of agents and proportions of the initial sheet without any modification to the agent program.

The second system, currently under development, applies the same approach to a different domain: the synthesis of arbitrary 2D shapes, inspired by cell growth. The goal is to compile a predetermined global shape to produce a program for a seed agent that then "grows" the structure through replication. We can compile a graphical description of an arbitrary connected 2D shape into a network of covering-circles, using techniques borrowed from computer graphics. Neighboring circles are linked using local reference points relative to each circle. This circle-representation permits the formation of the entire structure by agents recursively executing *only two simple primitives:* growing a circle, and triangulating the centers of adjacent circles. Locally the agents use replication, gradients and competition to achieve these primitives robustly. This system has the potential for self-repair and regeneration: agents can replicate to replace dead neighbors and the circle-representations allows agents to recreate entire structures that may be lost. We are currently evaluating the robustness of this system and extending it to 3 dimensions. We intend to extend this model to self-assembling robots, replacing cell growth and death with mobile agents that selectively attach and detach.

The power of the programming language approach is that it allows us to take advantage of traditional computer science techniques for managing complexity, while relying on biological models for achieving robustness at the local level. In both systems, a high-level shape representation was chosen such that the shape could be formed using a small set of simple primitives, with simple means of combination. This has many advantages: (1) We can rely on results from other disciplines, such as origami mathematics and computer graphics, to determine the classes of shapes that can be formed (2) The primitives themselves can be made robust by relying on mechanisms inspired by biological systems (3) The analysis of a complex system becomes tractable because it is built in understood ways from smaller parts (4) The compiler makes it possible to easily specify complex behavior.

This programming methodology will impact emerging fields such as reconfigurable robots, smart matter, and self-assembling systems, where we are trying to program specific global structure and function from vast numbers of parts. In the long run we believe these ideas will help achieve coherent behavior from aggregates of genetically-modified biological cells.As we build artificial systems that replicate biological robustness, we will inevitably develop insights into how complex morphology can arise at the global level from the local behavior of biological cells. We believe that this research will improve our understanding of morphogenesis and how development functions at a systems level.

Motivation

The motivation for this work comes from technologies such as MEMS[1] devices, that are making it possible to bulk-manufacture millions of tiny computing elements integrated with sensors and actuators and embed these into materials, structures and the environment. Applications include computing elements that can be “painted” onto a surface (Butera 2001; Williams & Nayak 1996), an airplane wing that changes shape to resist shear (Folk & Ho 2001), a programmable assembly line of air valves (Cheung *et al.* 1997), a reconfigurable robot that morphs into different shapes to achieve different tasks (Hogg, Bojinov, & Casal 2000; Butler, Byrnes, & Rus 2001). Emerging research in biocomputing may even make it possible to harness the many sensors and actuators in agents and create programmable tissues (Knight & Sussman 1998; Weiss, Homsy, & Knight 1999; Ingber 1998).

[1]Micro-electronic Mechanical Devices. Integrates mechanical sensors and actuators with silicon based integrated circuits.

These new computational environments pose many challenges to our current programming techniques. These applications will require the coherent coordination of vast numbers of elements, where the individual will have limited resources and reliability and the interconnects between elements will be irregular, local and possibly time-varying. In addition the physical embodiment of these elements is strongly intertwined with the individual behavior of the elements and the global goal. Not only are we concerned with topology, but also geometry and physics.

Current engineering paradigms rely heavily on precisely functioning parts and strictly managed interactions in order to achieve robustness (Berlin & Gilbert 1999; Lynch 1996). Programming strategies within the MEMs community have for the most part been centralized applications of traditional control theory, or decentralized versions that have access to global knowledge of the system (Berlin 1994; Hall *et al.* 1991; Williams & Nayak 1996; Pottie & Kaiser 1999) which are not scalable. The tendency to depend on centralized information, such as global clocks or external beacons for triangulating position, puts severe limitations on the types of applications and environments, and exposes easily attacked points of failure. In the modular robotics community, motion planning and heuristic searches are most common. These approaches quickly become intractable for large numbers of modules and are not robust to the failure of modules during the formation process. In addition they often require untenable assumptions such as explicitly defining the final position of every element and access to global position (Pamecha, Ebert-Uphoff, & Chirikjian 1997; Yim *et al.* 2000; Fukuda & Nakagawa 1988). In general, our engineering methodologies do not easily adapt to errors or unpredictable changes, and adding such fault-tolerance is very costly. These strategies put pressure on system designers to build complex, precise (and thus expensive) elements and interconnects rather than cheap, unreliable, mass-produced computing elements that one can conceive of just throwing at a problem.

Currently few alternatives exist. Approaches based on cellular automata have been difficult to generalize; local rules for agents are constructed by hand with insight and ingenuity; there is no general framework for constructing local rules to obtain a desired goal (Resnick 1994; Deneubourg *et al.* 1990; Mataric 1995; Butler *et al.* 2002). Evolutionary and genetic approaches produce local rules for prespecified goals, but the correctness and robustness of the evolved system is difficult to verify because of a lack of understanding of the local-to-global relationship (Forrest & Mitchell 1993).

By contrast, biological systems achieve incredible robustness in the face of constantly dying and replacing parts. The plethora of error-tolerant systems in biological systems, in contrast to our non-robust engineering paradigms, suggests that fundamentally different paradigms are used by biological systems to achieve this robustness. In recent decades, there has been significant progress in understanding how agents produce complex pattern and shape during development (Slack 1991; Wolpert 1998). Morphogenesis (creation of form) in developmental biology can provide insights for embedding computation in the environment. Epithelial cells, for example, generate a wide variety of structures: skin, capillaries, and many embryonic structures (gut, neural tube), through the coordinated effect of many agents changing their individual shape (Odell *et al.* 1981). We believe that important organizational lessons can be learned from these biological systems. This research is builds on previous work in the Amorphous Computing project, on exploring programming paradigms for collective behavior(Abelson *et al.* 2000; Coore 1999).

A Programmable Reconfigurable Sheet

The first system provides an example of how this approach can be used to specify shape formation on an intelligent reconfigurable sheet, composed of thousands of identically-programmed but locally-interacting flexible agents (Nagpal 2001; 2002).

Our model for a programmable sheet is inspired by folding in epithelial cell sheets — the programmable sheet consists of a single layer of thousands of randomly and densely distributed agents that can coordinate to fold the sheet along straight lines. All agents have the *identical* program, but execute it autonomously based on local communication and internal state. Communication is strictly local: an agent can communicate only with a small local neighborhood of agents within a fixed distance, or through physical contact as a result of folding. The sheet starts out with a few simple initial conditions, but apart from that the agents have no knowledge of global position or interconnect topology. An example application could be a reconfigurable sheet for deploying in space, that can fold compactly for storage but then unfolds into a chair or a shelter depending on how it is programmed.

In our system, the self-assembly works as follows: The desired global shape is specified as a folding construction on a continuous sheet, using a language called the Origami Shape Language (OSL). The language is based on a set of six paper-folding axioms (Huzita & Scimemi 1989), that can be used to construct a large class of flat folded shapes. The program for an individual agent is automatically compiled from the global shape description. All agents execute the same program and differ only in a small amount of local dynamic state. The agent program is composed from a small set of primitives: gradients, neighborhood query, cell-to-cell contact, polarity inversion and flexible folding. When the agent program is executed by all the agents in the sheet, the sheet is configured into the desired shape.

The language and primitives are presented in detail in (Nagpal 2002; 2001) along with many examples and theoretical and experimental results. Here we illustrate this approach through an example. Figure 1 shows a diagram for constructing a cup from a piece of paper using the paper-folding axioms. This diagram can be represented as an OSL program. The basic elements of the language are points, lines and regions. Initially, the sheet starts out with four corner points (`c1-c4`) and four edge lines (`e12-e41`). The axioms generate new lines from existing points and lines, while new points are created by intersecting lines. For example, the first cup operation constructs the diagonal `d1` from the points `c1` and `c2` by using axiom 2 (which essentially folds the sheet so that `c1` lies on `c2` and then unfolds the sheet). Lines can be used to create regions and regions can be used to restrict folds to certain layers. The fold is always a flat fold, and hence the structures created by OSL are flat, but layered.

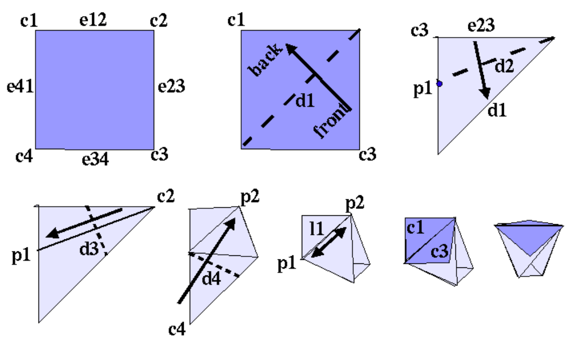
Figure 2 shows a programmable sheet differentiating to fold into a cup. Initially the surface is mostly homogeneous, with only the agents on the border having special local state. This is analogous to the presence of a small number of maternal determinants in an embryo, which determine the original axes of the embryo (Nusslein-Volhard 1996). The overall *global* view of this process is very close to what the diagram of the continous sheet suggest. This is because each global operation is translated into a local operation. But instead of folding, the axioms are implemented using biologically-inspired primitives.

For example, one of the key primitives is a *gradient*. A gradient is analogous to the gradient of a chemical concentration from a source. This primitive can be used for measuring distance as well as ascertaining local direction. It exploits the spatial locality of communication to make crude geometric inferences. Gradients themselves can be quite complex based on the location of sources, however a agent program for creating and propagating gradients is very simple. An agent creates a gradient by sending a message to its local neighborhood with the gradient name and a value of zero. The neighboring agents forward the message to their neighbors with the value incremented by one and so on, until the gradient has propagated over the entire sheet. Each agent stores the minimum value it has heard for a particular gradient name, thus the gradient value increases away from the source. Because agents communicate with only neighboring agents within a small radius, the gradient provides an estimate of distance from the source. The source of a gradient could be a group of agents, in which case the gradient value reflects the shortest distance to any of the sources. If a single agent emits a gradient then the value increases as one moves radially away from the agent but if a line of agents emit a gradient then the gradient value increases as one moves perpendicularly away from the line.

In order to implement the first line creation, the agents belonging to `c1` and `c2` create two distinct gradients. The remaining agents test if the gradient values are equal; if so then they lie on the new line. This is the local rule corresponding to axiom 2. Gradients also serve as a form of barrier synchronization, so that agents can determine when it is safe to move on to the next fold operation.

This work has several interesting features:

1. **Global-to-local Compilation:** The compilation process confers significant advantages: not only can the desired global goal be described at an abstract level, but results from paper-folding mathematics can be used to reason about what kinds of shapes can and cannot be self-assembled by the agents. Huzita has proven that the axioms can construct all plane Euclidean constructions(Huzita & Scimemi 1989). Lang(Lang 1996) has shown that tree-based folded shapes can be automatically



```
;; OSL Cup program
;;----------------------
(define d1 (axiom2 c3 c1))
(define front (create-region c3 d1))
(define back  (create-region c1 d1))
(execute-fold d1 apical c3)

(define d2 (axiom3 e23 d1))
(define p1 (intersect d2 e34))
(define d3 (axiom2 c2 p1))
(execute-fold d3 apical c2)

(define p2 (intersect d3 e23))
(define d4 (axiom2 c4 p2))
(execute-fold d4 apical c4)

(define l1 (axiom1 p1 p2))
(within-region front
  (execute-fold l1 apical c3))
(within-region back
  (execute-fold l1 basal c1))
```

Figure 1:

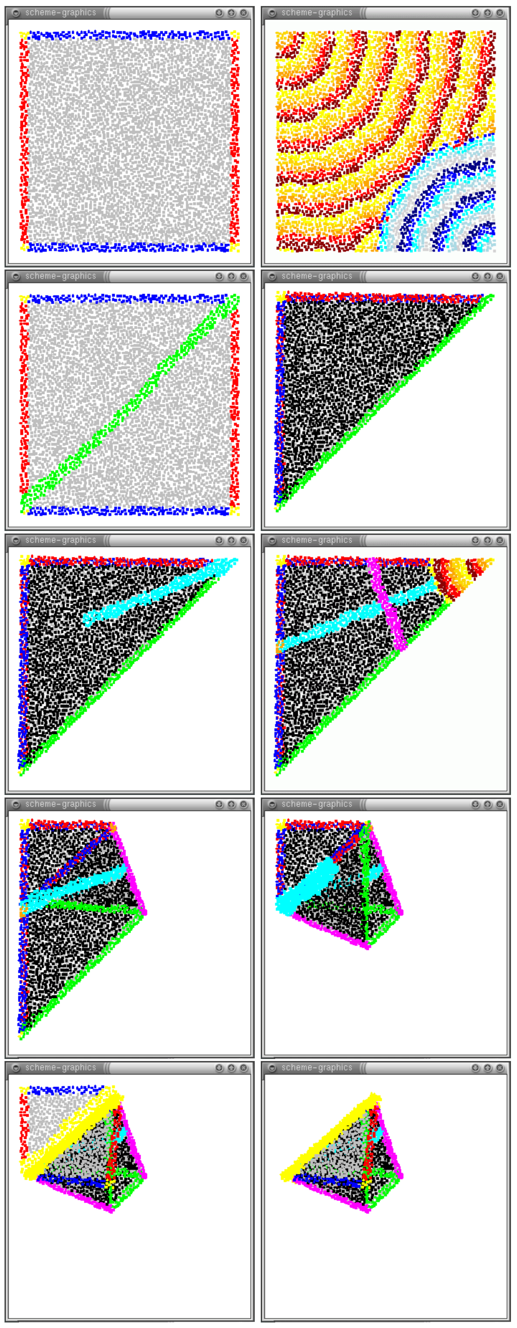


Figure 2: Simulation images from folding a cup

generated by computer and methods exist for constructing scaled polygonal shapes. There is a large practical literature of shapes that can be constructed using these techniques. As the relationship between paper-folding constructions and geometry is explored further, the results will directly impact this work.

2. **General-purpose Local Primitives:** This work uses a small set of powerful but general primitives for local organization. For example, we have used gradients for topological pattern formation language, the design of ad-hoc networks for distributed sensors, and discovery of global coordinates without the use of external beacons (Coore 1999; Butera 2001). Complexity comes from the effect of the interactions between the elements and their interactions with the environment.

3. **Robustness:** The agent programs are robust, without relying on regular agent placement, global coordinates, or synchronous operation, and can tolerate a small amount of random agent death. Robustness is achieved by depending on large and dense agent populations, using average behavior rather than individual behavior, trading off precision for reliability, and avoiding any centralized control. The robustness of the system can be analyzed by separately analyzing the primitives and the means of combination. We have shown that 15–20 agents is sufficient for gradients to produce reliable distance estimates, and the axioms compose to minimize error.

4. **Analogies to Biology:** The system also provides many insights into the relationship between local behavior and global morphology, that have analogies to biological systems. For example, the agent program is *scale independent* which implies it can create the same shape at many different scales without modification, simply by increasing the size of the agent sheet. Scalability is an extremely attractive feature: not only does the agent program work correctly over varying numbers of agents, but the function (goal) itself scales with the number of agents. This system can also generate many different related shapes without any modification to the agent program. This has significant implications for biology: if the agent programs do not differ, then methods such as genetic analysis are not likely to reveal much information. Instead observations of the process of development and comparisons between processes in closely related species may give us better insights into how morphology is created.

Self-Assembly 2D shapes based on growth

Our preliminary work has focused on 2D pattern formation and shapes produced through epithelial agent based folding. However this only scratches the surface of what is possible. Biology suggests many different metaphors for constructing complex structure: growth, mobility, death, material deposition, etc. Growth is fundamental to the creation of structure in embryos, from folding in the chick eye to limb generation. It is important not only for creating structure but also for repairing structure, such as regeneration of limbs (Wolpert 1998).

The second system, currently under development, applies the programming language approach to a different domain: the synthesis of arbitrary 2D shapes from cell growth. The goal is to compile a predetermined global shape to produce a program for a single seed agent that then "grows" the structure through replication.

In this case, we chose a very different shape representation from the sheet. A 2D connected shape is represented as *network of overlapping circles*. Neighboring circles are linked using local reference points relative to each circle; a circle can use its internal reference points to triangulate the location of all of its neighboring circles centers. A rooted spanning tree in this network represents a process for drawing the structure starting from a particular circle.

This work goes a step further than the previous work: we can compile such a shape description directly from a graphical description of an arbitrary connected 2D shape, using techniques borrowed from computer graphics. Figure 3 shows the circle-network for a cross shape, compiled from a drawing program rendition.

This circle-network representation is key because it permits the formation of the entire structure by agents recursively executing *only two* high-level operations: growing a circle via agent replication, and triangulating target agents which will become reference points and centers of adjacent circles.

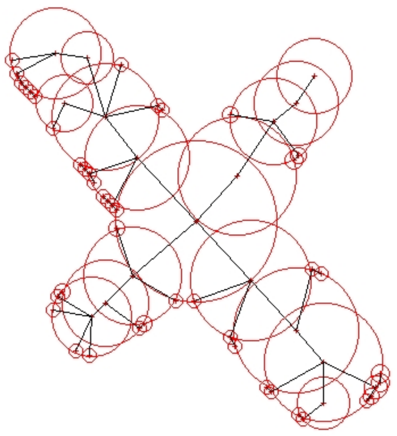
The model for an agent is similar to an agent in the programmable sheet, except instead a fixed number of agents, there is a single agent that can produce more agents through replication. At the lowest level, the agent can:

1. Create gradients
2. Exchange messages with immediate neighbors
3. Reproduce, by placing the child agent in a random position within a fixed small radius. If the location is occupied then the replication is considered failed, and the agent may try again.
4. Die, after death an agent is removed from the substrate.

These primitives are then combined to allow the agent to execute the high-level operations.

Agents can assume different roles, and their role determines what behavior they will have. For example, an agent with the role of circle center induces replication of agents outward to a certain radius, thereby causing a circle to form around itself. Reference points are agents that have been designated as "coordinates" of a local grid; these agents create gradient messages that allow nearby agents to triangulate their relative positions.

As agents begin to form circles, four agents in each circle designate themselves as cardinal reference points— that is, they mark the north, south, east, and west poles of a circle. Agents with reference-point roles create distinct gradient messages, from which a nearby agent may extract information about its relative orientation. This, in effect, establishes a local coordinate system that aids agents in determining where the next circle must be grown. The gradients do not travel outside the circle, but within the circle they are modeled as direct diffusion and therefore provide a better



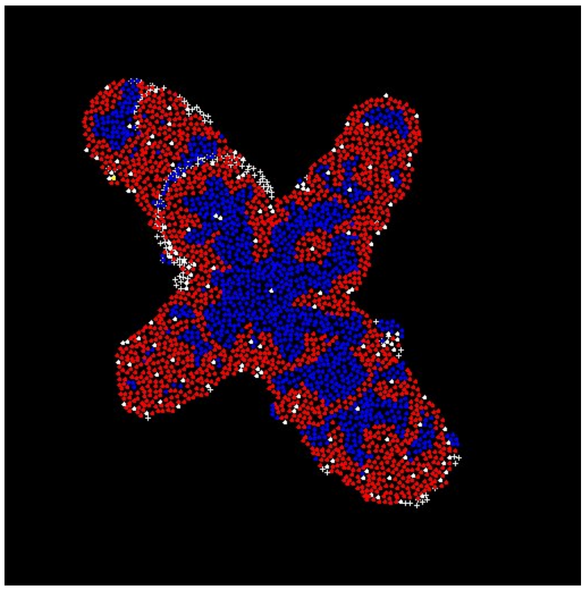


Figure 3:

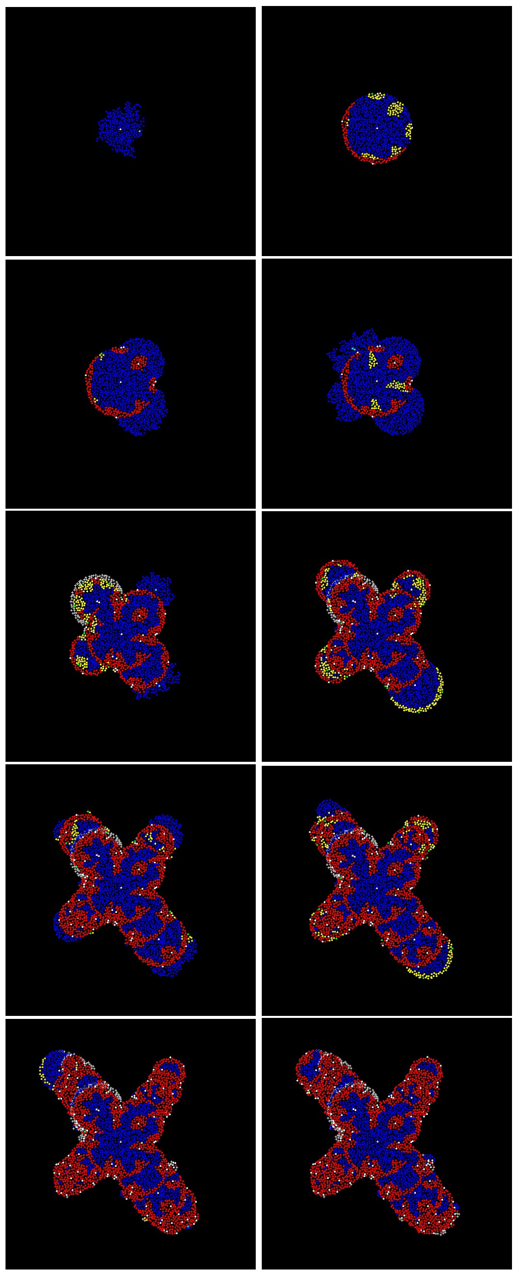


Figure 4:

distance estimate than gradients simulated by message passing.

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

Figure 4 shows the process of a single agent creating the cross shape. The agent program is generated from the circle-network representation.

We have many simulations of 2D shapes. We are currently evaluating the robustness of the system. The preliminary qualitative results suggest that the system can produce good replications of the target shape give a high density of cells within a single circle. The higher the density the more likely there is an agent that is well placed to be a reference point. However qualitative results are needed to place bounds on the error that results from insufficient density. We are also evaluating the robustness of the competition mechanisms, which are likely to be applicable to many different applications. Extending the shape representation to 3D is relatively straightforward however it does add a significant amount of complexity in the number of additional reference points that must be created.

Self-repair

A key feature of this system is that it has the potential for both self-repair and regeneration after the formation process is over.

The first method of self-repair is based on awareness of neighboring agents. In the process of growing a circle, every agent that hears a growth message from the circle center attempts to reproduce and place daughter agents randomly around itself within a ring. This agent only succeeds in reproducing if there is room at the chosen location for another neighboring agent. Agents are aware of their neighbors, and they resume replication when a neighbor disappears. Hence the structure never really stops growing and constantly replaces dying parts.

The role-competition among agents also allows the system to recover when an agent holding a reference point role disappears. The competition for this role among the agents in the local neighborhood resumes. However with very high probability, a neighbor's descendant will fill the space left by the dead agent in time to become the new role-holder.

The system is also capable of *regeneration* after a large part of the shape is destroyed. This is a feature of the shape representation chosen. Each circle contains enough information to generate its neighbors, and this process can occur recursively. As a result agents can rely on their original growing procedure to regenerate. Our most recent compilation specifies multiple ways in which any single reference point can be regenerated and each circle can be regrown from multiple neighbors. Therefore any part of the structure, that has at least a circle center and enough reference points to orient itself, can regenerate the rest.

Conclusion

This research represents a different approach to engineering self-organizing systems. Rather than trying to map a desired goal directly to the behavior of individual agents, the problem is broken up into two pieces: (a) how to decompose the goal globally (b) how to map the construction steps to local rules. In both examples, we take advantage of current understanding in other disciplines of how to decompose a problem. This approach suggests that exploring new global paradigms is at least as important as experimenting with local rules. The choice of shape specification language determines to a large degree the decomposition of the problem. For example, paper folding is a natural process for describing shape formation on a sheet and the axioms provide the decomposition. Choosing the right decomposition will be necessary in order to translate abstract notions of shape to local processes. We will need to discover new models to describe goals in new domains, and this will require cross-disciplinary thinking.

Biology gets tremendous mileage by using vast numbers of cheap and unreliable components to achieve complex goals reliably. Our current research has focussed on self-assembly, however the same approach could be applied to distributed robots or sensor networks. Developmental biology can teach us how to make complex shapes, with well-defined topological and functional structure. Study of social insects can teach us how to achieve robust behavior from ant-like robots. The global-to-local compilation is the key to achieving complexity while still being able to analyze the behavior of the system. By encoding these processes as programming languages we can combine principles for controlling complexity, drawn from computer science, with techniques for robust design, inspired by biology.

Acknowledgements

This research is 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.

References

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

Berlin, A., and Gilbert. 1999. Collaboration and coordination in massively distributed smart matter. *Workshop on Amorphous Computing*.

Berlin, A. 1994. *Towards Intelligent Structures: Active Control of Buckling*. Ph.D. Dissertation, MIT, Dept of Electrical Eng. and Computer Science.

Butera, W. 2001. *Programming a Paintable Computer*. Ph.D. Dissertation, MIT Media Lab.

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

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

Cheung, P.; Berlin, A.; Biegelsen, D.; and Jackson, W. 1997. Batch fabrication of pneumatic valve arrays by combining mems with printed circuit board technology. In *Proc. of the Symposium on Micro-Mechanical Systems, ASME Intl. Mech. Engineering Congress and Exhibition*.

Coore, D. 1999. *Botanical Computing: A Developmental Approach to Generating Interconnect Topologies on an Amorphous Computer*. Ph.D. Dissertation, MIT, Dept of Electrical Eng. and Computer Science.

Deneubourg, J.; Goss, S.; Franks, N.; Sendova-Franks, A.; Detrain, C.; and Chretien, L. 1990. The dynamics of collective sorting: robot-like ants and ant-like robots. *Simulation of Adaptive Behavior: from animals to animats*.

Folk, C., and Ho, C.-M. 2001. Micro-actuators for control of delta wing with sharp leading edge. In *39th AIAA Aerospace Sciences Meeting and Exhibit*.

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

Fukuda, T., and Nakagawa, S. 1988. Approach to the dynamically reconfigurable robot systems. *Journal of Intelligent and Robotic Systems* 1:55–72.

Hall, S.; Crawley, E.; Howe, J.; and Ward, B. 1991. A hierarchic control architecture for intelligent structures. *Journal of Guidance, Control and Dynamics* 14(3):503–512.

Hogg; Bojinov; and Casal. 2000. Multiagent control of self-reconfigurable robots. In *4th International Conference on Multi-Agent Systems*.

Huzita, H., and Scimemi, B. 1989. The algebra of paper-folding. In *First International Meeting of Origami Science and Technology*.

Ingber, D. 1998. The architecture of life. *Scientific American*.

Knight, T., and Sussman, G. 1998. Cellular gate technology. In *Proceedings of the First International Conference on Unconventional Models of Computation (UMC98)*.

Lang, R. J. 1996. A computational algorithm for origami design. In *Annual Symposium on Computational Geometry*.

Lynch, N. 1996. *Distributed Algorithms*. Wonderland: Morgan Kaufmann Publishers.

Mataric, M. 1995. Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems* 16((2-4)):321–331.

Nagpal, R. 2001. *Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics*. Ph.D. Dissertation, MIT, Dept of Electrical Engineering and Computer Science.

Nagpal, R. 2002. Programmable self-assembly using biologically-inspired multiagent control. In *Autonomous Agents and Multiagent Systems (AAMAS)*.

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

Odell, G.; Oster, G.; Alberch, P.; and Burnside, B. 1981. The mechanical basis of morphogenesis: 1. epithelial folding and invagination. *Developmental Biology* 85:446–462.

Pamecha; Ebert-Uphoff; and Chirikjian. 1997. Useful metrics for modular robot planning. *IEEE Trans. on Robotics and Automation* 13(4).

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

Resnick, M. 1994. *Turtles, Termites and Traffic Jams*. Cambridge, MA: MIT Press.

Slack, J. 1991. *From Egg to Embryo, second edition*. U.K.: Cambridge University Press.

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

Williams, B. C., and Nayak, P. P. 1996. Immobile robots: AI in the new millennium. *AI Magazine*.

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

Yim, M.; Lamping, J.; Mao, E.; and Chase, J. G. 2000. Rhombic dodecahedron shape for self-assembling robots. Spl techreport p9710777, Xerox PARC.