Chapter 1

Programmable Pattern-Formation and Scale-Independence

Radhika Nagpal
PostDoctoral Lecturer, MIT Artificial Intelligence Lab
radhi@ai.mit.edu

This paper presents a programming language for pattern-formation on a surface of locally-interacting, identically-programmed agents, by combining local organization primitives from developmental biology with combination rules from geometry. The approach is significantly different from current approaches to the design of self-organizing systems: the desired global shape is specified using an abstract geometry-based language, and the agent program is *directly compiled* from the global specification. Using this approach any 2D Euclidean construction can be formed from local-interactions of the agents. The resulting process is extremely reliable in the face of random agent distributions and varying agent numbers. In addition the process is *scale-independent*, which implies that the pattern scales as the number of agents increases, with no modification of the agent program. The pattern also scales asymmetrically to produce related patterns, such as D'Arcy Thompson's famous transformations.

1.1 Introduction

Cells cooperate to form complex structures, such as ourselves, with incredible reliability and precision in the face of constantly dying and replacing parts. Emerging technologies, such as MEMs, are making it possible to embed millions of tiny computing and sensing devices into materials and the environment. We would like to be able to build novel applications from these technologies that achieve the kind of complexity and reliability that cells achieve. These new environments pose significant challenges: a) How does one achieve a particular global goal from the purely local interactions of vast numbers of parts? b) What are the appropriate local and global paradigms for engineering such systems?

This paper presents a programming language approach to self-assembling complex structures, using techniques inspired by developmental biology. We present a programming language for instructing a surface of locally-interacting, identically-programmed agents to differentiate into a particular pattern. The language specifies the desired global pattern as a construction on a continuous sheet, using a set of axioms from paper-folding (origami) mathematics [3]. In contrast to approaches based on cellular automata or evolution, the program executed by an agent is *automatically compiled* from the global shape description. With this language, *any plane Euclidean construction* pattern can be be specified at an abstract level, compiled into agent programs, and then synthesized using purely local interactions between identically-programmed agents. The process relies on the composition of a small set of general and robust biologically-inspired primitives. The resulting process is not only reliable in the face of random agent distributions and random agent death but is also theoretically analyzable[4].

The process is also *scale-independent*. Scale-independence implies that the pattern, however complex, scales as the number of agents increases, with no modification of the agents program. The pattern also scales asymmetrically, allowing a single program to generate many related patterns, such as D'Arcy Thompson's famous coordinate transformations which he used to explain shape differences in related species[5]. Scale-independence is common in biology. The pattern-formation process provides insights into how complex morphology emerging from local behavior can exhibit global properties such as scale-independence.

This research is motivated by emerging 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 to build novel applications: smart materials, self-reconfiguring robots, self-assembling nanostructures. Approaches within the applications community have been dominated by a centralized, hierarchical mind-set. Centralized control hierarchies are not easily made scalable and fault-tolerant, and centralized/heuristic searches quickly become intractable for large numbers of agents. The tendency to depend on centralized information, such as global clocks or external beacons for triangulating position, puts severe restrictions on the applications while exposing easily attackable points of failure. Currently, however, few alternatives exist. Decentralized approaches based on cellular automata models of natural phenomena and artificial life research have been difficult to extend to engineering systems; local rules are constructed empirically without providing a framework for constructing local rules to obtain any desired goal. Evolutionary and genetic approaches are more general but the local rules are evolved without any understanding of how or why they work. This makes the correctness and robustness

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

of the evolved system difficult to analyze.

Biological systems achieve incredible robustness in the face of constantly dying and replacing parts. 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 propose to use morphogenesis and developmental biology as a source of mechanisms for organizing complex behavior. Our approach is to formalize these general principles 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. This work is part of larger vision called Amorphous Computing to explore new programming models for collective behavior[1, 2].

1.2 A Programmable Surface

Our model for a programmable surface consists of randomly distributed agents on a 2D surface. 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 distance r. The surface starts out with a few simple initial conditions, but there are no external beacons for triangulating position or global clocks. Individual agents have limited resources and no unique identifiers, instead they have random number generators to break symmetry. The motivation comes from the applications — we would like to cheaply bulk manufacture billions of smart sensors and embed them in the environment. Assumptions such as globally unique identifiers, global clocks or coordinates, regular grids and perfectly reliable elements are unrealistic in this setting. Furthermore developmental biology suggests that it should be possible to construct complex structures without such assumptions.

1.3 Global Specification Language

The key to this process is the language for specifying the desired global pattern. Paper-folding (origami) provides a natural, although somewhat unusual, way for describing how to create patterns starting from a blank continuous sheet. Huzita presented a set of axioms for origami, a subset of which are equivalent to the plane Euclidean axioms (ruler and compass axioms)[3]. Each axiom generates new lines starting from an existing set of lines and points.

1. `crease-lbp`: Fold a line between two points $p1$ and $p2$.
2. `crease-p2p`: Fold $p1$ onto $p2$ to create a line (perpendicular bisector).
3. `crease-l2l`: Fold line $L1$ onto $L2$ to create a line (bisector of the angle between $L1$ and $L2$).
4. `crease-l2self`: Fold $L1$ onto itself through $p1$.

```
;; OSL PROGRAM for a CMOS INVERTER
(define v1 (crease-l2l e23 e41))
(define v2 (crease-l2l v1 e23))
(define v3 (crease-l2l v1 e41))
(define IN  (create-region e41 v3))
(define OUT (create-region e23 v1))
(define MID (create-region v1 (or v2 v3))
;; Similarly create horizontal regions...

;; Lay down Material (differentiate)
(within-region IN (color h1 "poly-red"))
(within-region MID (color (or h2 h3) "poly-red"))
(within-region OUT (color h1 "poly-red"))
(within-region CNTR (color v3 "poly-red"))
(within-region UP (color v1 "n-diff-yellow"))
(within-region DOWN (color v1 "p-diff-green"))
(define contacts (intersect v1 (or e12 h1 e34))
(color contacts "contacts-black")
```

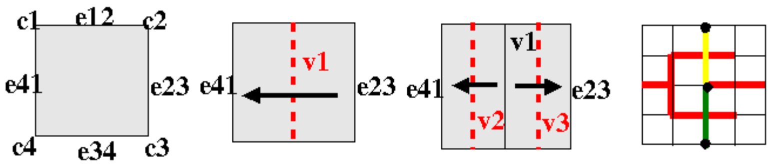

Figure 1.1: OSL code and diagram for a CMOS Inverter

In addition, there are two more operations - intersecting two lines to find a new point and defining a region. Given a line that divides the sheet into two parts, one side can named as a region using a landmark point on that side.

We have developed a programming language based on these axioms and paper-folding practice, called the Origami Shape Language (OSL). The language is described in detail in [4]; here we have illustrated it with a simple example, a caricature pattern of a CMOS inverter. Figure X shows a diagram of how the pattern is created and the corresponding OSL code. The sheet always starts out blank but with four sides and corners. Using the axioms the sheet is recursively subdivided and a grid pattern of lines is created. The lines are used to define different regions and the regions are used to set the colors (state) of the agents. In an actual inverter, the different colors refer to different materials.

Rather than specify the desired pattern directly, this language specifies a “generative program” or a process for creating the pattern. However this specification is abstract — the process is on a continuous sheet with no notion of agents or self-assembly. By specifying the pattern this way, we can take advantage of known and new results in geometry to analyze the power of our system.

1.4 Global to Local Compilation

The agent program is directly compiled from the global shape specification. This is done by using a small set of primitives for organization at the local level and combining these primitives in well-understood ways to produce robust and predictable behavior.

1.4.1 Biologically-inspired Primitives

Gradients: Gradients are analogous to chemical gradients secreted by biological cells; the concentration provides an estimate of distance from the source of the chemical. Gradients are believed to play an important role in providing position information in morphogenesis[6]. 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. Thus, the shape and positions of the sources affects the spatial pattern of gradient values. For example 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 emits a gradient then the gradient value increases as one moves perpendicularly away from the line.

Neighborhood Query: This primitive allows an agent to query its local neighborhood and collect information about their state. For example an agent may collect neighboring values of a gradient for comparison. This primitive is from cellular automata. An agent can also broadcast a message to its local neighborhood.

1.4.2 Composition into Local Rules

Each of the global OSL operations can be implemented as a simple agent program (also called a local rule) using the above primitives. OSL points and lines are represented by groups of agents; all agents the group are equal. Each agent has a boolean variable in its internal state for each distinct point/line. Initially the sheet starts out with four distinct lines and points (edges and corners). The initial conditions are very simple; agents do not know where they are within an edge and the remainder of the sheet is homogeneous, like a blank sheet of paper.

The axioms use gradients to determine which agents belong to the crease. The axioms make use of the fact that gradients provide a distance estimate as well as reflect the shape of the source. For example, axiom 1 creates a line from one point to another by having one point generate a gradient and the other point “grow” a line towards increasing values of the gradient, similar to Coore[2].

Axiom 2 creates a line such that any point on the crease is equidistant from points $p1$ and $p2$. Therefore if $p1$ and $p2$ generate two different gradients, each agent can compare if the gradient levels are approximately equal to determine if it is in the crease line. The other two axioms are similar, but take advantage of the fact that lines produces gradient values that increase away from the line.

Regions allow the user to *restrict the context* in which a local rule applies. A region is created by using bounded gradients, which implies that certain types of agents will not forward the gradient message; the intuition being that certain agents can act as barriers to particular gradients. A gradient is created from point that cannot pass a particular line, marks the region on one side of the line.

The final compilation of an OSL program involves creating local boolean state variables for each distinct point and line and then translating each OSL operation into a call to the corresponding agent procedure with the appropriate arguments. The compiler assigns different gradient names for each call. For example, `(define d1 (crease-p2p c1 c3))` becomes `(define d1 #f) (set! d1 (axiom2-rule c1 c3 gradient1 gradient2))`. Thus, the agent program mirrors the original OSL program. However note that at the OSL level there is no notion of gradients, or even agents. The compilation process from global to local is easy to understand. However the agent programs generated are no different from any other emergent systems — the eventual shape "emerges" as a result of local interactions between the agents and the initial conditions.

1.5 Examples

The examples presented were generated by specifying the pattern using OSL, compiling the OSL program to generate the agent program and then executing the agent program on the simulated programmable sheet. The initial conditions are always the same, and the number of agents is between 1000-8000 with local neighborhoods of 15 agents on average. Figure 1.2 shows the result of simulating the inverter program. Complex patterns can be created in a modular fashion: for example a chain if inverters can be created by segmenting the sheet into regions and invoking the inverter program within each region. Many examples of patterns are presented in [4].

1.6 Scale-Independence and Related Shapes

The formation of the same structure at many different scales is common throughout biology. Many species develop normally over large variations in egg sizes and large morphological differences can occur between species with little genetic difference. Genetic analysis is unlikely to reveal much information in such cases. However artificial systems can give us insight into how complex patterns may achieve scale-independence and what limitations exist on the scaling.

The Origami Shape Language is a scale-independent description of shape, the program specifies the shape but not the size of the sheet to use. At the cell level

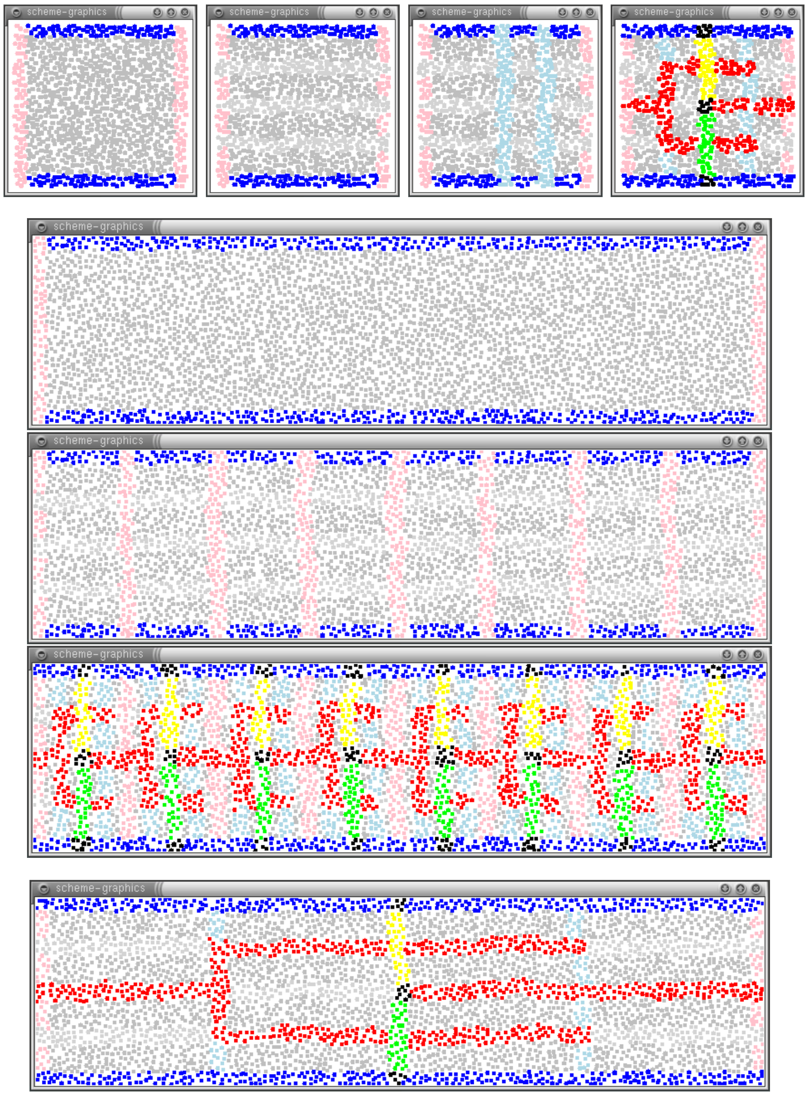

Figure 1.2: Simulation images of pattern-formation: single inverter, chain of inverters, the single inverter run on a long sheet.

scale-independence is achieved by using primitives that depend on comparisons of gradients values, never absolute values. Axiom 2 is similar to Wolpert's model of balancing gradients, and axiom 1 grows a line until the end is reached - so both primitives scale. At the global level scale-independence is achieved by recursively creating nesting structures, starting from the original boundary. Thus one can create highly complex structures without reference to size.

This leads to an interesting observation, which is that by changing the *shape* of the boundary we can also change the shape generated. The inverter program executed on a long sheet produces a stretched inverter, a kite shaped surface produces a distorted inverter. D'Arcy Thompson observed that the forms of many related animals (crabs, fish, skulls) could be transformed into one another by stretching along different axes, and that the relationships extended even to internal structures. He envisioned differential growth as the mechanism responsible. The origami shape language suggests another mechanism — changing the shape of initial boundary.

1.7 Conclusions

This work 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 achieve the goal globally b) how to map the construction steps to local rules. The compilation process confers many advantages: we can use theoretical results from paper-folding to reason about the kinds of shapes can and cannot be self-assembled and we can use the decomposition into primitives and means of combination to analyze the robustness of the system. We believe that many of these mechanisms will be applicable to programming smart matter and reconfigurable substrate applications, as well as provide a basis for directing experiments towards understanding biological morphogenesis.

Bibliography

[1] ABELSON, ALLEN, COORE, HANSON, HOMSY, KNIGHT, NAGPAL, RAUCH, SUSSMAN, WEISS "Amorphous Coomputing" *Communications of the ACM* **volume 43, number 5** (2000).

[2] COORE, Daniel, "Botanical Computing: A Developmental Approach to Generating Interconnect Topologies on an Amorphous Computer", *PhD Thesis*, MIT, Department of Electrical Engineering and Computer Science, Feb 1999.

[3] HUZITA, Humiaki, "The Algebra of Paper-folding", *1st International Meeting of Origami Science and Technology* (1989).

[4] NAGPAL, Radhika, "Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics" *PhD Thesis*, MIT, Department of Electrical Engineering and Computer Science, June 2001.

[5] THOMPSON, D'Arcy, *On Growth and Form*, Cambridge University Press, U.K., 1961

[6] WOLPERT, Lewis, *Principles of Development*, Oxford University Press, 1998.