Synthetic Morphogenesis: Space, time, and deformation

by

Micah Z. Brodsky

B.S., University of Washington (2005)
S.M., Massachusetts Institute of Technology (2009)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2014

Author .....
Department of Electrical Engineering and Computer Science
August 29, 2014

Certified by .....
Gerald Jay Sussman
Panasonic (Matsushita) Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
Professor Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Synthetic Morphogenesis: Space, time, and deformation

by

Micah Z. Brodsky

Submitted to the Department of Electrical Engineering and Computer Science
on August 29, 2014, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

Synthetic biology has presented engineers with a fascinating opportunity: can we understand the principles of our origins – animal embryonic development – by re-engineering it in the laboratory? I investigate, from an engineer's perspective, some of problems that arise in developing geometric form in a deformable substrate. More abstractly, I attack the problem of establishing spatial patterns, when rearranging and deforming parts of the system is inherent to the process.

Deformable, foam-like cellular surfaces are developed as a model for embryonic epithelia (polarized cellular sheets), one of the principal tissue types in early animal development. I explore ways in which simple agent programs running within the individual cells can collectively craft large-scale structures. The mechanical properties of the substrate prove crucial to the patterning process.

In such a distributed, heterogeneous substrate, little can be assumed about the progress of time. In one branch of my work, I develop patterning techniques where convergence is transparently and locally detectable, drawing insights from clockless digital circuits and casting the problem as distributed constraint propagation. In another branch of work, I avoid the problem of timing by making all patterns self-correcting.

In self-correcting patterning, I attempt to understand "canalization" – how development is naturally robust to perturbations. I formulate a model for regional patterning, inspired by regeneration experiments in developmental biology, and using mathematical principles from classical models of magnetic domains and phase separation. The problem again becomes a form of distributed constraint propagation, now using soft constraints. I explore some of the resulting phenomena and then apply the mechanism to crafting surface geometries, where self-correction makes the process robust to both damage and self-deformation. I conclude with a look at how this naturally leads to an example of partial redundancy – multiple systems that partly but not completely overlap in function – yielding confusing responses to the effects of virtual knock-out experiments, reminiscent of the confusing behavior of knock-out experiments in biology.

Thesis Supervisor: Gerald Jay Sussman
Title: Panasonic (Matsushita) Professor of Electrical Engineering

Acknowledgments

With a journey this long, it becomes impossible to enumerate all the individuals and fellow travelers whose hospitality made it possible to reach this point.

I'd like to acknowledge the brief but eye-opening contribution made when I was an undergrad by the crew of organizers and speakers at Caltech's 2004 Computing Beyond Silicon Summer School, especially André DeHon, Erik Winfree, Michael P. Frank, and Tom Knight. Tom, in particular, made me realize that biology was really about systems, and not just glorified stamp collecting. This marked the turning point where I began to explore the patterns in living systems, instead of avoiding biological studies like the plague.

Mike Swift, also during my undergrad, led me through my first serious research project while he was finishing his doctorate. I ultimately followed a different path, but the early experience was invaluable (and surely crucial in getting accepted at MIT).

Around the lab, discussions with Robert Berwick (reader), Arnab Bhattacharyya, Alexey Radul, Pavel Panchekha, Norman Margolus, Mitchell Charity, Mark Tobenkin, and Ian Jacobi proved crucial in finding my way and overcoming obstacles. In my previous life with the systems and networking crowd, Max Krohn, Hariharan Rahul, Nate Kushman, Austin Clements, and Russ Cox filled a similar role. It's a blessing to be accompanied by people so much more accomplished than I, who seem to scatter food for thought wherever they go.

The role of Jake Beal was especially important, both for his inspiring work on spatial computing and for his time and effort showing me through the academic game. Radhika Nagpal's work was also profoundly inspiring, and I only wish she could've found more time for me ;). Her work, along with the book by Marc Kirschner and John Gerhart [36], were the germ that gave rise to this thesis.

Of course, none of this would've amounted to anything without my advisor, Gerry Sussman. Gerry was the reason I came to MIT, even though it took four years before I had the guts to drop everything and set up shop with him without any money. I have never worked well in sink-or-swim environments, and Gerry has never believed in education by attrition. Gerry taught me to be judiciously over-optimistic in planning my goals, and he helped nurture my ideas along until they amounted to something (and then helped nurture me along until I *realized* they amounted to something). Gerry

has a special talent for freeing up friction-locked thinking, and I hope I have learned something from him about how to think beyond "allowed" bounds. It's not going to be easy walking away from his office.

Outside of academics, I owe a great debt to Jamie Spangler, Brett Shapiro, Vivian Hecht, Ethan Sokol, Isaac Oderberg, Eliad Shmuel, Rabbi Michelle Fisher, and the rest of the crew around MIT Hillel, whose tireless community-building efforts made the Institute a place I could call home.

I must acknowledge my wife, Andrea Gillespie, who in addition to putting up with me all this time, also contributed several key technical insights to this work.

I'm not even going to try to enumerate all the teachers who had a lasting impact on me; there are too many of you. I hope you know who you are. This thesis is dedicated, above all, to the teachers, mentors, and family who accommodated my quirks and refused to give up on me, even when times seemed dark. I hope I've proved your efforts worthwhile.

This thesis is based on work supported in part by the National Science Foundation under Grant No. CNS-1116294, in part by a grant from Google, and in part by MIT's EECS department via a heck of a lot of TAing.

Contents

Chapter 1

Introduction

In recent years, computer scientists have begun seeking insight among natural phenomena for the design of robust systems and spatially-embedded computations ("spatial computing") [1, 7, 67, 2, 8]. Animal and plant development, in particular, have provided a trove of novel ideas [15, 71, 45, 17, 20]. Development, however, entails deformation and reinvention of the spatial substrate itself, into which the computation is embedded; deformation, in a sense, is the heart of development. Thus far, little attention has been paid to this crucial but complicated aspect, even restricted to the case of idealized surfaces, which capture the most interesting elements.

Nature is a master craftsman of surfaces. From the leaves of carnivorous pitcher plants to the imaginal disks that give rise to limbs and antennae in flies, nature demonstrates the extraordinary repertoire that can be algorithmically sculpted from 2-dimensional surfaces. Indeed, given the long and intricate story of the primordial germ layers in animals, much of embryonic development can be cast as the elaboration and interaction of 2-dimensional surfaces with their environments. Although considerable attention has been focused on processes involved in laying down patterns on existing surfaces, under such banners as pattern formation [50], amorphous computing (e.g. [1, 45, 15]), and cellular automata, little seems to be understood about the development of dynamic substrates and surface geometries themselves. In most natural examples, this involves the reshaping and elaboration of preexisting surfaces by locally mediated changes. Complicating this process, substrate deformation seems to cross-couple any algorithmic elements in unexpected ways, with large-magnitude strains distorting the results of prior patterning steps and leaking beyond the boundaries of regions.

Nature's resiliency in producing viable structures in spite of evolutionary and environmental perturbations, and in spite of this complexity, put humankind's rigid engineering techniques to shame. What sorts of algorithmic tools does nature use to ensure robust, reliable, and evolvable development of the desired structures? Can we exploit these techniques ourselves, either for engineering applications or to better understand the methods of nature?

1.1 Approaches to Patterning Geometry

I set out to explore the question of how a deformable sheet of cells might fashion itself into the kinds of complex geometries naturally produced by the processes of embryonic development. One might naively imagine that the problem could be divided into two steps: patterning the sheet in its initial configuration with a pre-pattern describing the geometric features to be crafted and then sculpting the features according to the pre-pattern. If such a strategy were adequate, then the existing body of research studying the patterning of fixed substrates, e.g. Amorphous Computing, Turing patterns, and so forth, would provide solutions to the first half of the problem, and my research could focus merely on the actuation of pre-specified geometric features.

However, it turns out that the pre-patterning strategy alone is neither necessary nor sufficient. It is not necessary because there exist reliable mechanisms for producing feature patterns directly as geometry, without any prior reflection in the logical states of the cells. The most notable example is compressive buckling. In theory, on a mathematically ideal manifold, pre-patterning is nonetheless sufficient, but in practice it encounters a host of complications. Because the ability to separate where features go from how to construct them offers such conceptual advantages, and because ample evidence demonstrates biology indeed using such strategies [36], the bulk of my research (aside from developing my numerical surface model) became devoted to avoiding and overcoming the complications inherent to pre-patterning.

The simplest, least abstract approach to pre-patterning would be to paint a full mathematical description of the geometry. In other words, a patterning cascade would construct a set of pattern fields that described all the local geometric properties of the surface. The mechanical properties of the substrate would then be modified locally to match the pre-pattern stencil's prescription.

This is immediately nontrivial, because deforming the substrate may entail stretching and rotation, deforming and reorienting the pre-pattern in the process. If these distortions are predictable, the pre-pattern may be corrected in advance. However, if pre-patterning computations are not finalized and suspended prior to actuation, the pre-patterning process itself will be perturbed through geometric feedback, greatly increasing overall complexity. This can be avoided by globally checkpointing the pattern at completion of pre-patterning, before any actuation, or by using pre-patterning mechanisms that are insensitive to the kinds of deformations that will appear.

As a first attempt at pre-patterning, one might construct pattern fields that described the direction(s) in which the surface was to be curved and the associated radii of curvature. Such "extrinsic" curvatures could then be created by driving the surface to bend accordingly, for example, via apicobasal constriction. This proposal, however, suffers from two serious flaws: it both incomplete and inconsistent. It is incomplete because distinct shapes may be described with indistinguishable curvature fields, for example, a 2x2 square curved along a cylindrical arc and a 1x4 rectangle with the same radius of curvature. Given a particular algorithm and initial conditions, some shape will of course be produced, but it may not be be possible to fully specify which one. Worse, however, this scheme is inconsistent, for the same reason one cannot flatten an orange peel without tearing it: extrinsic curvatures require, in general, non-Euclidean geometries within the surface. Distances between points within the surface must change in order to accommodate the extrinsic curvature. As the surface deforms extrinsically, such "intrinsic curvature" will necessarily be generated, by elastic deformation, plastic flow, and other passive material responses, at the cost of high stresses, stress-driven distortions to the final curvature, and possible material failure.

The complementary strategy, patterning only intrinsic curvature, e.g. by adjusting cell sizes and in-plane eccentricities, is similarly limited, both incomplete and inconsistent. Purely cylindrical curvatures cannot be specified, and the inverted forms of curves are indistinguishable. Stresses and bending torques again accumulate, this time in order to force the material to bend extrinsically, fighting against its flexural rigidity. Like extrinsic pre-patterning, intrinsic pre-patterning is functional but severely limited.

The flaws of inconsistency and incompleteness can be avoided by simultaneously patterning the surface both intrinsically and extrinsically. Mathematically, this entails patterning the first and second fundamental forms of classical surface geometry, that is, the metric tensor and the shape tensor.

Such a description is complete, but it is also highly redundant – most features need to be specified twice, in different ways. The relationship between the two representations is complicated and nonlocal (the Gauss-Codazzi equations). Any inconsistency between the two manifests as before, in stresses and distortions. Such redundant description bodes ill for both engineering and evolvability: adding and changing features is no longer simple, because changes must be made two different ways simultaneously. One might term this "embrittling redundancy".

Even with consistent, simultaneous patterning, and more so without, a subtle flaw remains. While the final state of a consistently and simultaneously patterned surface may be stress-free, the intermediate states during convergence need not be. Some deformations, such as dimpling a sphere (i.e. invagination), naturally entail elastic snap-through [59], passing through a high-stress, unstable state. Even under less challenging transformations, high stresses can arise due to timing skew across the substrate, where some changes are actuated before others. While such stresses will eventually recede, they may nonetheless cause permanent deformations to the substrate by exceeding the yield limit, triggering plastic flow. Plastic flow is problematic because it is discontinuous, irreversible, and somewhat unpredictable, and hence is very difficult to accommodate in a fixed pre-pattern. Even fully consistent deformations may be subject to transiently induced plastic distortion.

For sufficiently large deformations, plastic flow can also be a necessary tool, as a result of the discrete, cellular nature of the substrate. Without plastic flow, large deformations such as evaginating a tentacle out of a sheet would entail cells stretching into strands of spaghetti. However, cells can only deform so far. Further deformation requires that cells be rearranged or destroyed and created anew.[1] Without new cell growth, actuation must purposefully trigger plastic flow in order to achieve large deformations.

Stress-driven plastic flow and other forms of cell rearrangement are thus all but inevitable for large deformations, and patterning algorithms must be prepared for them. Small deformations can be achieved approximately with naive pre-patterning, and accuracy can be improved by incorporating local feedback in actuation. However, large deformations entail poorly predictable rearrangement of the substrate during the patterning process, and such rear-

[1] In the special case of reducing a single large cell into many small cells, it is also possible to generate large deformations through directed division, although this often produces plastic rearrangement as well.

rangement rearranges the pre-pattern itself. This fundamentally changes the nature of the pre-patterning problem. Rather than pre-patterning once and checkpointing static stencils, patterning must be a continuous, self-correcting process that operates concurrently with actuation.

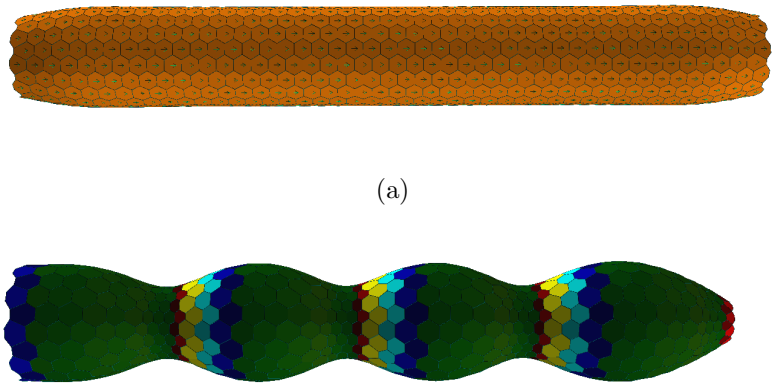
I explored two general strategies for large deformation patterning, one based on direct geometric control without pre-patterning and one based on self-correcting pre-patterns. In the first strategy, organized proliferation patterning, cells are grown or rearranged according to some control law, generally based on outside feedback. Geometric features result largely from buckling, with the mechanical properties of the substrate playing an important role. In the second strategy, self-stabilized pre-patterning, a pre-pattern is established by a symmetry-breaking, feedback-based patterning process, which continues operating throughout actuation. As actuation distorts the surface, the pre-pattern readjusts itself to fit. Actuation applies feedback control to local geometric properties, such that the desired final geometry becomes a stable attractor of the system. These two general strategies, in addition to naive pre-patterning for small deformations, characterize most of my successful experimental results.

Naive Pre-patterning

Although limited in scope, naive pre-patterning is still a useful technique, and its key concepts are relevant to more sophisticated patterning strategies. Naive pre-patterning can also be used to assist in bootstrapping and combining more sophisticated algorithms, by arranging patterning modules and pre-breaking symmetries.

The most immediate problem in implementing naive pre-patterning is determining when pre-patterning has converged so that actuation may begin. In a centralized algorithm this would be trivial, but in distributed setting it is less so. If the size of the domain and the speed of computation are known (or at least known to be consistent, and hence measurable), time to completion can be estimated in advance for some algorithms (e.g. [45]), although such dead reckoning estimates can be a form of embrittling redundancy. Other algorithms are more problematic, such as Turing patterning, which is inherently metastable. If size and speed are variable, e.g. due to growth, nutrient, and temperature variation, even well-behaved algorithms defy timing by dead reckoning. Without knowing when the pattern has converged, naive pre-patterning is limited to weak, non-disruptive actuation

(e.g. Figure 1.1), unable to safely modify cellular topology. In Chapter 3, I demonstrate how to solve this problem for a broad class of patterning algorithms, using a method of computation such that convergence is detectable locally.



(a)

(b)

Figure 1.1: Example of small-deformation naive pre-patterning. (a) Initial conditions, consisting of a uniform cylinder of polarized cells (taper is a passive effect of surface tension at cut ends). (b) Final structure, a corrugated cylinder. The pre-pattern is series of segments, each partitioned in half, such that cells in the left half are instructed to enlarge and cells in the right half are instructed to shrink. The segments themselves are produced by a somitogenesis-inspired clock-and-wavefront mechanism [47] (effectively, an implementation of the modulo function on a gradient that does not require multiple thresholds). The magnitude of deformation shown is approaching the disruption-free limit.

Organized Proliferation Patterning

In organized proliferation patterning, cells are directed to grow and divide according to their proximity to organizer regions and according to the values of gradients both confined within the surface and emanating through the ambient medium. Proliferation leads to both buckling and yielding, along with displacement of the organizers and motion relative to external gradients.

The combination of all these effects produces the final shape; no pre-pattern is ever constructed.

Organized proliferation algorithms are among the easiest large-deformation programs to implement, often requiring only a few lines to specify. They are also among the hardest to predict in detail, relying heavily on complicated mechanical effects and feedbacks. I lack a general theory of the mechanism, although mechanical intuition can serve as a useful guide. I explore this patterning mechanism through a series of examples in Section 2.4, showing how to produce a family of assorted axisymmetric and branching structures (as in Figure 1.2).

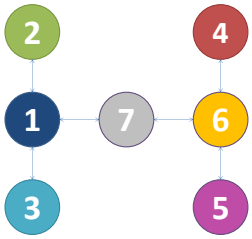
Figure 1.2: Example of an allotropically-scaled, randomly branching tree, produced by organized proliferation.

Self-Stabilizing Geometry

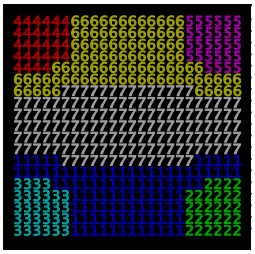
In naive pre-patterning, a structure, once produced, is static. In organized proliferation patterning, a structure may continue to grow, and a branched structure may, like a plant, grow additional structures to replace ones that are lost or enlarge existing structures that face increased load, but the particulars of a structure cannot be regenerated. Most animals, however, develop fairly particular structures and have at least some ability to regenerate when damaged. Some, such as freshwater hydra, can reconstruct themselves even after complete disaggregation [27]. In self-stabilizing geometry, I finally attack the problem of morphological homeostasis – how to produce and maintain a

consistent, particular structure in spite of developmental uncertainty and a lifetime of insults and changing circumstances.

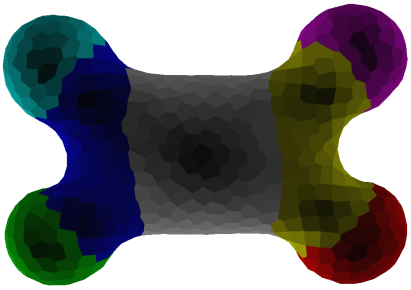
The general strategy is a combination of decomposition and feedback. Based on a high-level, topological specification (e.g. Figure 1.3a), a pre-pattern is constructed and maintained by a novel feedback patterning algorithm developed in Chapter 4. With such a pre-pattern, stable to both external insults and disruptive actuation, aggressive feedback algorithms for controlling geometry can be layered on top. These algorithms, along with the final results of the cascade, are discussed in Chatper 5. The results, as in Figure 1.3c, are a first, elementary demonstration of programmable morphological homeostasis.



(a)



(b)



(c)

Figure 1.3: Example developing a four-lobed, self-stabilizing structure, shaped via normal neighbors patterning and closed-loop curvature control. (a) Normal neighbors graph, a topological specification for the pre-pattern, or "body plan", of the structure. (b) Example realization of the body plan pattern on a static, regular square lattice. (c) Self-stabilizing 3D geometry produced by combining the body plan with mechanical actuation.

1.2 Philosophy: An Engineer's Perspective

Why should engineers care about biology and development?

Traditionally engineered systems are brittle. Despite modern know-how and materials, it has proven hard for engineers to design and construct robust, flexible systems: systems that have acceptable behavior over a wide range of situations, anticipated or otherwise. Instead, our artifacts are specified with narrow tolerances and precise interfaces, pushed beyond which they fail, often spectacularly. Furthermore, efforts to rebuild and redesign them to accommodate a widened scope tend to be costly and time-consuming.

By contrast, the natural world is filled with examples of robustness and flexibility: wounds that heal, bones that strengthen (anisotropically) from stress, behaviors that survive rich times and lean times, microbes that evolve to resist antibiotics, clades of species of all kinds that survive changing environments over and over again. From single-celled organisms through entire ecosystems, nature seems to know something we don't about flexibility.

This research is an investigation into design principles for flexible physical systems, both natural and engineered. I use the medium of abstract programmable deformable surfaces, as an approximation of both natural developmental epithelia and synthetic smart materials. Through mathematical modeling, simulation, and theoretical analysis, I develop and characterize algorithmic techniques capable of producing and maintaining three-dimensional structures by local surface deformation. The most sophisticated of the techniques show flexibility to mechanical and environmental variability and successful regeneration even in the face of massive damage. These techniques and the experience of developing them can provide insight into how engineered systems can mimic natural flexibility.

1.2.1 Flexibility, Robustness, and Evolvability

Pondering flexibility more closely, we may articulate two distinct forms. *Robustness* is the tolerance exhibited by an artifact for varying parameters, circumstances, and insults, internal and external. Critical to achieving this is *adaptivity*, the ability of an artifact to self-adjust its structure and/or behavior to variability. *Evolvability*, by contrast, is the ease with which an artifact's specification can be altered by minimal increments to specify a new artifact capable of performing in new and different circumstances. In engineering,

this corresponds to the ability to efficiently re-design an updated version. In biology, this corresponds to the ability of a population of organisms produce new and different mutant lineages when circumstances demand. Where robustness involves only the behavior of an artifact, evolvability concerns also the representation used for its specification and what constitutes a minimal, "easy" change.[2]

While human engineering uses different sorts of specifications and different kinds of incremental changes from biological species, the parallels run deep. In biology, major leaps and re-designs are limited by the low probability of hitting mutations that make large changes and yet preserve viability. In engineering, major leaps and re-designs are limited by their cost and the difficulty of ensuring continued functionality, or in other words, preserving viability. Just as in biology, strings of small, incremental changes that can be validated along the way are a far easier path to follow. In biology, potential changes are also restricted by the compatibility requirements of the pool of prospective mates and of symbiotic and ecological relationships. In engineering, potential changes are restricted by the compatibility requirements of physical and logical interfaces with existing systems. Compatibility constraints extend within systems as well, inasmuch as the systems can be decomposed into subsystems and modules that must cooperate. Co-evolution is as much a fundamental problem in engineered ecosystems as it is in natural ones.

Another notable parallel between biological and engineered systems is the significance of *neutral space*, the freedom of variation in specification and behavior lacking any observable impact on the system's objectives. In software engineering, the exploitation of neutral space is known as refactoring, the functionality-neutral restructuring of code that nonetheless facilitates subsequent functional changes. In biology, the phenomenon of neutral drift allows a population to explore a range of genotypic and phenotypic variations, all while carefully skirting non-viable regions. Some of these variants will prove useful under altered circumstances or as starting points for functional mutations unreachable from the original starting state. In both cases, the effective result of neutral exploration is to increase the feasible range for incremen-

[2]The distinction is slightly blurred when speaking of self-replicating artifacts that are ultimately responsible for (mis)duplicating their genomes, but this only becomes a concern at the population level: numerous individual organisms will die off, their robustness failing them, in the quest of the larger population to adapt and produce an improved individual specification.

tal changes to the system: larger viable changes are possible than with a specification where every possible adjustment is meaningful.

Neutral space, however, is not merely a function of specification encoding. The encoding must ensure that a small, local change does not cause catastrophic, long-distance effects having no chance of being neutral, but it need not ensure a complete absence of side effects. If the encoded behaviors of a system are sufficiently robust, the system can adapt around any deleterious effects of small modifications, rendering an ostensibly non-neutral specification change phenotypically neutral or neutralizing the unintended side effects of an intentional change. Thus, the behavioral robustness of an artifact facilitates its evolvability.

Robustness facilitates evolvability in other ways as well. Robustness facilitates compatibility, by tolerating variations seen along the interfaces. Where compatibility constrains evolvability, robustness helps to release it.[3] Also, robustness through adaptivity allows a system to accumulate behaviors corresponding to a wider range of environments. As the environment shifts, these alternative behaviors can be cemented as the new default via simple changes to the specification. Adaptivity thus facilitates evolvability towards new parameter regimes. The concept that robustness facilitates evolvability is not new [36], but in this line of research, we have the opportunity to investigate it constructively.

1.2.2 Pursuing Flexible Systems

A key goal of the work presented here is the pursuit of a deeper understanding of flexible systems, both for scientific insight and engineering applications. Nature furnishes us a tremendous number of examples and existence proofs of flexibility, but interpreting them remains a challenge. This work investigates such natural examples by re-engineering solutions under simplified,

[3]It is worth noting, however, that a contrary phenomenon is sometimes also observed in long-lived engineered systems: Broad, forgiving interfaces on one artifact can allow other artifacts to establish relationships via otherwise fragile means. As these relationships proliferate, they haphazardly constrain the interface from multiple points, depriving it of neutral space. A variety of engineering techniques exist to combat this accumulation of unnecessary constraints, such as the publishing of written specifications and standards, linguistic and mechanical mechanisms that enforce information hiding, ex-post-facto compatibility and virtualization layers, and so on, and yet it remains a source of considerable consternation in industry. It would be interesting to investigate whether biology too suffers from the compatibility entrenchment problem and if it employs any protective weaponry.

simulation-friendly models of the underlying problems and comparing the results against what is known about the biological solutions. Using the same models, I also investigate related engineering problems, particularly those that arise in the pursuit of robust and adaptive physical devices.

In the pursuit of flexible physical devices, we rapidly run up against the limitations of today's engineering materials: fashioned top-down, all at once, they're built for a specific, macroscopic purpose and last a limited time. Fortunately, a suite of novel, adaptable substrates are on the horizon, including shape-memory actuated foldable surfaces [32], smart textiles [31, 60] and artificial muscle fibers [60], modular robotics [73] and catoms [28], programmable self-assembly [41], and synthetic cells and engineered epithelia [65, 4]. These incipient technologies offer new opportunities for flexible devices and programmable manufacturing, but they also force us to face the challenge of how to control them. In this research, I focus primarily on programmable deformable surfaces, as embodied in engineered epithelia, because they are a powerful, versatile medium that nicely match nature's own media and because the control problems they present are novel and substantial.

How do we approach the sculpting of a deformable surface for adaptive physical artifacts? Some applications might be able to rely on external tooling and bracing for shaping, but in general this may be inflexible, heavy, or otherwise infeasible and cannot be depended upon. Similarly, some applications may be able to rely out outside computation and control, but in general it can be impossible to gather all sensory information centrally and distribute detailed actuation commands globally. Such is the case with synthetic biology, and as electronic active components miniaturize and proliferate in number, electronic smart materials too will be subject to such restrictions. This leaves us with the *amorphous* approach to computing and control [1], which assumes unreliable components furnished with local sensing, local computation, local actuation, and local communication. This model is, not coincidentally, quite reminiscent of biology.

The central focus of this research is the control of such amorphous deformable surfaces, from the standpoints of basic algorithmic techniques for basic goals and robust and evolvable control for robust and evolvable development and behavior. In fact, these different perspectives on the problem are intertwined, robustness with evolvability for the reasons explored above in 1.2.1 and robustness with basic techniques because of the inherent risk of local component failure and the likelihood of structural irregularity when managing countless cooperating active elements. The latter issues have long

been a defining challenge in the field of spatial computing [6], on which this research builds. Dynamic development and deformation of the underlying substrate increase the need for robust approaches even more.

What gives rise to robust control? No simple, universal answer is known. In spatial computing, this question has been met with such techniques as stable geometric constructions (e.g. gradients and geodesics [15, 45]), high level specification languages (e.g. Growing Points Language [15], Origami Shape Language [45], and circle networks [37]), spatial redundancy (e.g. [5, 37]), neighborhood feedback and relaxation methods (e.g. [3]), and self-stabilizing algorithms (demonstrated in, e.g., [3]). In natural patterning systems, notable claims for robustness motifs include cooperative degradation of morphogens and stable attractors but also fairly complicated multi-component schemes whose rationale is not immediately obvious [21]. A common theme in many though not all of these techniques is a deep and general strategy: the use of feedback control. The recent *functional blueprints* concept [7] formalizes this idea in a manner particularly apt for spatial computing and structural development. Less explored, though easily integrated into the functional blueprints framework, is multimodal control, the use of partially redundant actuation methods. This has been observed many times in biological processes (e.g. multiple sperm exclusion, blood coagulation) and in homeostasis control, and I suspect it also lurks within developmental patterning. These and other robustness methodologies are explored in my research on deformation control.

Chapter 2

Model: Foam-Inspired Surface Mechanics

A variety of well-established models are used to assist in exploring spatial computing and pattern formation on a static substrate, e.g. amorphous computing, cellular automata, agent swarms, and dynamical systems. The same is not true for the more complicated case of deformable substrates. Yet, computational models are all the more necessary here, because the more complex dynamics are less tractable both analytically and intuitively, and surprises abound.

Is it possible to construct a simple model with the kind of universal character exhibited by cellular automata and amorphous substrates? Since the nature of the kinds of active manipulations and passive mechanical dynamics present within a substrate vary significantly, possibly not; the same model may not work equally well for animal embryos, adult plant tissues, and smart fabrics. However, the essential elements should be specifiable in terms of how the substrate is divided into agent domains, how the domains deform, and under what circumstances they rearrange.

I set out to develop a model that would be most appropriate for embryonic and synthetic epithelia, with side-applications to other domains. To this end, I idealized the substrate as a two-dimensional surface deformed in 3D and divided it up into polygonal, approximately planar agents ("cells"). Each cell has an energy function that describes the favorability of different mechanical conformations, and the dynamics are computed by energy minimization, simulating the ultra-slow, quasi-static regime. A set of topological rules inspired by the physics of 2D foams [64, 46] allows cells to rearrange

their neighbor contacts, simulating the processes of cell migration and plastic flow.

2.1 Model Overview

How should a virtual tissue be decomposed into agents? For many adult tissues, with mechanically important components in the extracellular matrix, the answer isn't immediately clear, but for embryonic tissues, especially epithelia, cells are the obvious choice. In epithelial tissue, cells are packed tightly together and represent a discontinuous grain in both material and computational properties. Each cell constitutes a single, coherent nuclear and cytoplasmic environment, with limited neighbor communication, and thus naturally corresponds to a computational agent. Cells may also slide past one another with reasonable ease, particularly in embryonic epithelia, but a single cell can be deformed only so much. With sophisticated material models, one can construct efficient, approximate mechanical models with a coarser grain than cells [11, 13], but cells as agents remain ideal from a control perspective, particularly when cell migration and the associated transportation of computational state is concerned.

What about the deformability of individual cell domains? A cell is neither quite solid nor liquid. Like a liquid, it is incompressible but easily deformed, yet like a solid, it resists shear forces. Unlike volume elements of both liquids and solids, cells have boundaries, and cell-cell contacts support normal forces far better than shear forces. As a result, a cell will deform nonuniformly under shear, and with sufficient strain, may slide past its neighbors. A cell is, in many ways, better modeled as bubble or liquid droplet, fluid but confined by a surface tension. Lacking the data or justification to construct a rigorous model of intra-cellular mechanics, I chose to model cells with a very simple two-term planar model, combining a cortical surface tension and an area restoring force, plus an additional term for bending stiffness. Cells are represented only by their 2-dimensional outlines within the surface, their thickness merely implicit.

Developmental process take place on long timescales and short length scales, immersed in a fluid at small Reynolds number. As a result, inertia is insignificant, and the dynamics of deformation are dominated by viscosity. Even such viscous dynamics are not necessarily significant, however. Gene regulation and hence most developmental feedbacks take place on the

timescale of hours, meaning that faster mechanical behaviors happen all but instantaneously from a cell's perspective. For this reason, I keep the model's dynamics even simpler, simulating in the quasi-static or "adiabatic" regime. In this regime, agents are only allowed to execute once the last round of deformation has largely stabilized. Mechanical conformations can be computed efficiently by energy minimization, and there is no need to supply viscosity parameters, only an energy function. From an agent's perspective, the system is always at mechanical equilibrium.

Unlike volume elements in a homogeneous solid or liquid, cells can exchange neighbors when deformed, producing a discontinuous rearrangement of the substrate. Like bubbles in a foam, contact areas between neighbors will shrink under appropriate strains. When their contact areas shrink away entirely, cells pull apart and their neighbors slide into their place (Figure 2.4). This interchange mechanism is known in the foam physics literature as the T1 process [64] and is a key point of departure from models appropriate to textiles, plant tissues, and other substrates with largely fixed neighbor relationships. Individually, the T1 process allows cells to migrate by exerting tractions on their neighbors. Collectively, it causes bulk plastic flow under shear stress of sufficient magnitude, imparting a yield limit to the substrate. Aside from explicit cell division and death, the T1 process is the sole mechanism by which cells rearrange in the model.

Whenever the model is sufficiently close to mechanical equilibrium, cell agents are given the opportunity to intervene. Agents can manipulate the mechanical properties of their cells, such as size, shape, and elasticity, or cause them to divide or die, and thereby perturb the surface such that it deforms into a new conformation. In order to direct such actuation, agents can sense local geometric and mechanical properties, such as curvature and stress. They can execute arbitrary computations and keep small amounts of local state. They can share information with their nearest neighbors by exchanging morphogen signals, modeled as floating point values. They can also sense and emit diffusive gradients that propagate through the ambient medium in 3D.

Collectively, these modeling design choices strive to capture the essentials of epithelial developmental mechanics, without unnecessary levels of detail and complexity. The following sections explain the mechanisms of the model more thoroughly, including the parameters and their physical meanings. I also discuss the physical situations for which the model is and is not appropriate as well as design decisions that were important for the model's efficiency

and stability.

2.2 Foam Cell Mechanics

The model represents a deformable surface as a complex of simply-connected, polygonal, approximately planar cells. Cells are affixed to their neighbors through shared vertices and edges, and the surface deforms by virtue of the relative motion of the vertices . Vertex motion is governed by a mechanical energy function spanning the entire surface; vertex locations are adjusted in order to relax the energy function toward a local minimum. The energy function is responsible for determining the essential mechanical properties of the substrate as well as for ensuring the well-behavedness of the cellular mesh.

The global energy function is constructed as the sum of local contributions from each cell area. A cell's energy contribution is a function of the positions of the vertices belonging to both the cell and its immediate neighbors, along with a variety of per-cell mechanical parameters.

Within a 2D surface, the per-cell energy functions favor properties common to embryonic epithelial cells, soap bubbles, and other natural cellular materials – maintenance of a natural volume and minimal boundary surface area consistent with volume – and hence produce an emergent geometric order similar to planar versions of such materials. Key features include convex cells with vertices of degree three and vertex angles near 120° [26, 46] – a close approximation to Plateau's laws of soap films. Under mild in-plane stresses, the edge lengths distort, providing an elastic response. The bulk elastic properties are statistically isotropic for random cell meshes but also exactly isotropic to first-order in the case of a regular (and necessarily hexagonal) lattice [39]. When cell distortion reduces an edge to zero length, the vertex topology can then invert, a phenomenon characteristic of all these systems [46, 64], yielding a natural mechanism of cell rearrangement and plastic deformation.

The energy functions additionally include terms specific to flexible surfaces in 3D, such as elastic thin shells and epithelia, to encourage smoothness and provide flexural rigidity. These terms enforce minimal bending at the junction between cells (apart from a "wedge angle" setpoint) and minimal distortion from planarity within a cell. These help to stabilize the mesh against pathologies, as well as providing elastic bending properties.

Term	*Algebraic form*	*Parameter*	*Description*	*Default*	*Dimensions*
Area restoration	$k_A(\mathsf{A} - A_0)^2$	A_0 k_A	Area setpoint Area elastic K	12 0.5	L^2 E/L^4
Surface tension	$k_P\mathsf{P}$	k_P	Perimeter surface tension	1.0	E/L
Junction bending	$k_B\mathsf{L}_{\mathsf{edge}}(\Theta - \theta_0)^2$	θ_{0u} θ_{0q} k_B	Angle setpoint Angle setpoint quadrupole Bending elastic K	0 0 0.8	rad rad $E/Lrad^2$
Distortion	$k_D\frac{A_0}{2}{\epsilon_{\mathsf{distortion}}}^2$	k_D	Distortion K	100	E/L^2
Contact force	$k_C \min(\mathsf{A_i}, \mathsf{A_j})($ $D_r/\mathsf{D_{ij}} - 1)^2$	k_C	Contact force K	0.01	E/L^2
Area guard	k_i/A	k_i	Collapse guard	0.1	EL^2

Table 2.1: Principal mechanical energy terms and their parameters, which can be custom-specified when initializing a simulation as well as manipulated online by cell agents. Default values are used in all examples unless otherwise specified.

Although naturally inspired, the details of the energy functions are synthetic, chosen for simplicity and transparent behavior, rather than attempting to mimic any specific natural system. The complete set of terms are enumerated in Table 2.1 and discussed below.

2.2.1 Two-Dimensional Mechanics

To model the first-order elastic behavior of an approximately isotropic substance in 2D, only two independent parameters are needed, e.g. shear and bulk moduli. Because we are interested in large deformations, higher-order terms would ordinarily also be important. However, for a soft, thin surface, inelastic and 3-dimensional behaviors such as buckling typically take over at large deformations, and so higher order 2D modeling is only of secondary concern. Thus, we have a fair amount of freedom in defining a 2D energy function and can select the simplest mechanism that provides reasonable cell shapes and interchange behaviors.

In 3D, the simplest plausible energy function might use surface tension to minimize surface area, in conjunction with a volume constraint. In formulating a 2D surface idealization, we can often assume that, in addition to a hard constraint on volume, the thickness of cells is roughly consistent, and so this

3D behavior is equivalent to the minimization of 2D perimeter subject to the constraint of constant 2D area. Of course, thickness is not exactly constant and can vary about its equilibrium value. Thus, a soft constraint on area, implying a soft constraint on thickness, is more useful. Taken together, this suggests the following 2D energy function:

$$E_{2d} = k_P \mathsf{P} + k_A(\mathsf{A} - A_0)^2 \tag{2.1}$$

where P is the cell perimeter, A is its area, and A_0 is a parameter specifying the equilibrium area. The soft constraint on area is chosen to be quadratic as this provides a first-order term, higher terms being less important.

Such an energy function naturally provides the foam-like properties outlined above. Vertex angles tend toward 120°, and junctions of degree greater than three are generally unstable, in close accord with Plateau's laws. In the case of a regular mesh, equilibrium vertex angles are exactly 120°, even under stress; only edge lengths distort, not angles. Under sufficient stress, selected edges will deform to zero length, effectively producing four-fold vertices. Energy then can be further reduced by splitting each four-fold vertex into two new three-fold vertices – a T1 topological interchange – and allowing the newly created perpendicular edges to grow (see Section 2.2.3). Such rearrangement is permanent, remaining even after the applied stress is removed.

Continuum limit properties

At a scale larger than cells, what kinds of mechanical properties can we expect from such a cellular material? By analyzing the energy function of a single cell under small deformations, we can derive the linear elastic properties of a regular lattice of cells, giving a reasonable first approximation to the elastic properties of an arbitrary lattice. This allows us to relate model parameters to measurable bulk properties.

For convenience, all derivations here employ 2D elastic moduli rather than the more common 3D versions; these can be related to 3D elastic moduli given a known substrate thickness (see Table 2.2a).

In linear elasticity, the shear modulus μ and Lame's first parameter λ are defined by

$$E = \mu \epsilon_{ik}^2 + \frac{\lambda}{2} \epsilon_{ll}^2 \tag{2.2}$$

using tensor notation, where ϵ is the linear strain tensor and E is the strain energy. In two dimensions, with coordinate axes along the principle axes of strain, this reduces to

$$E = \mu(\epsilon_1^2 + \epsilon_2^2) + \frac{\lambda}{2}(\epsilon_1 + \epsilon_2)^2 \tag{2.3}$$

where ϵ_1 and ϵ_2 represent the principal strains – the fractional increments by which the material is distended along perpendicular axes. Knowing that a hexagonal lattice is linearly isotropic [39], we can pick any arbitrary axes for our test deformation. For convenience, we will use axes aligned to the cell's hexagon.

We substitute ϵ_1 and ϵ_2 into our 2D per-cell energy function above and allow the cell to assume whatever shape minimizes the energy consistent with the strains. This turns out to be a hexagon with 120° angles but edges of two different lengths. Then, by computing derivatives with respect to ϵ, we can determine μ and λ, our first order elastic coefficients. Defining $0 <= Q << 1$ as

$$Q = 1 - A_r/A_0 \tag{2.4}$$

where A_r is the equilibrium area of the undistorted cell (slightly less than A_0 due to surface tension), we can conclude

$$\lambda = k_A A_0(2 - 4Q) \tag{2.5}$$

$$\mu = k_A A_0 Q \tag{2.6}$$

The Poisson ratio, 2D Young's modulus, and 2D bulk modulus are then (see Table 2.2b)

$$\nu = \frac{1 - 2Q}{1 - Q} \tag{2.7}$$

$$Y = k_A A_0 \frac{2(2 - 3Q)Q}{1 - Q} \tag{2.8}$$

$$K = k_A A_0(2 - 3Q) \tag{2.9}$$

By setting to zero the partial derivatives of energy for a free cell, A_r (and Q) are found to be determined by

$$(A_0 - A_r)\sqrt{2A_r/\sqrt{3}} = k_P/k_A \tag{2.10}$$

3D modulus	*Name*	*2D expression*
Y_{3d}	Young's modulus	Y/t
ν	Poisson ratio	ν
μ_{3d}	Shear modulus	μ/t
K_{3d}	Bulk modulus	$\frac{3K\mu}{3(3\mu-K)t}$

(a)

2D modulus	*Equivalent*
Y	$\frac{4K\mu}{K+\mu}$
ν	$\frac{K-\mu}{K+\mu}$
λ	$K-\mu$
K	$\frac{Y}{2(1-\nu)}$
μ	$\frac{Y}{2(1+\nu)}$

(b)

Table 2.2: (a) 3D elastic moduli expressed in terms of 2D moduli, such that an isotropic slab of thickness t (under plane stress conditions) gives the specified 2D moduli by the conventions used in this analysis. (b) Relations among 2D elastic moduli.

which is analytically solvable but messy. In the (realistic) limit of small surface tensions, Q is approximately

$$Q \approx k_P/(k_A A_0^{(3/2)}\sqrt{2/\sqrt{3}}) \tag{2.11}$$

Qualitatively, we can observe that bulk stiffness scales mainly with $k_A A_0$, while shear stiffness scales with $k_P/\sqrt{A_0}$: k_A determines compressibility, and surface tension determines shearability, while both are scaled relative to the cell's dimensions.

Extensions

Optionally, the surface tension term may be varied on a per-edge basis, differing from neighbor to neighbor. This can be used to model differential adhesion: cells that have mutual affinity have a lower surface tension along their contact edges, and vice versa. This captures the surface energy due to the binding of adhesion domains.[1] In the model, differential adhesion is represented by assigning cells adhesivity type identifiers and then looking up identifier pairs in a table to determine surface tension adjustments. Cells with mutual affinity naturally tend to cluster, with sharp boundaries separating distinct clusters; the effect is much like the segregation of oil and water in a thin fluid layer (e.g. as in Figure 2.1).

[1]The same mechanism is also used to model the lack of adhesion at free edges; surface tension is increased relative to internal cell-cell contacts

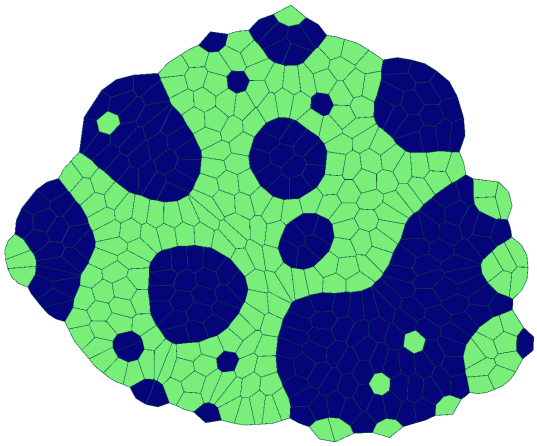
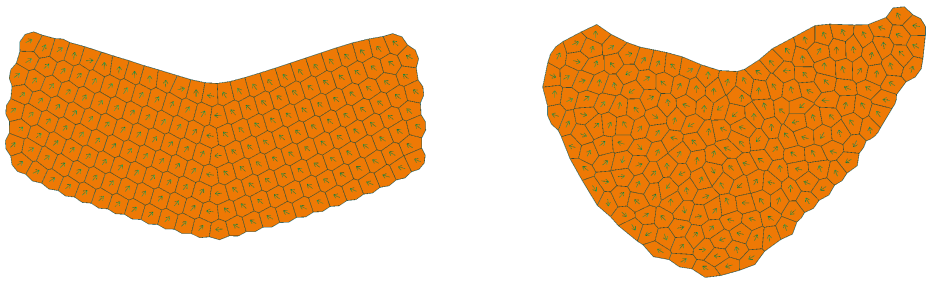


Figure 2.1: Adhesive phase separation in an initially regular rectangular array whose cells are randomly chosen among two adhesion classes (each with self-affinity reducing k_P by 80% for internal contacts).

The same mechanism can also be used to model cell-cell traction via lamellipodia [10]. In this case, rather than modifying a cell's own edges, a radial edge between two adjacent neighbors is affected. The surface tension in that edge is increased by the magnitude of the traction forces exerted on its terminal vertices (see example in Figure 2.2). In the model, a cell's traction forces are thusly specified on a per-vertex basis. When topological changes occur, e.g. due to traction forcing, per-vertex traction forces must be modified to apply to the new set of vertices, at least until the cell agent can re-specify the forces. Tractions from two vertices that merge are summed, while tractions on a vertex that is split are divided equally between the child vertices.

In the original energy function, note that, for sufficiently large $k_P/(k_A A_0^{3/2})$, cell equilibrium area collapses (bulk modulus, Equation 2.9, goes to zero at $Q = 2/3$); such parameter regimes lead to unrealistic behavior and are out of the model's scope. In practice, it is useful to add an additional small term to the energy function to guard against complete cell collapse, not to improve accuracy, but so that when such parameters are inadvertently encountered the inaccurate modeling remains localized to the affected areas rather than causing geometric singularities. I include a term of the form k_i/A for this purpose, using a small value for k_i.

Cells thus far are naturally isotropic. Sometimes, however, is it important



(a) (b)

Figure 2.2: (a) Regular rectangular array elastically distorted by cell-cell traction forces exerted along the radial edges approximately pointed to by the arrows (planar polarization vectors). (b) Increased traction leads to plastic intercalation, permanently rearranging the lattice, as well as causing substantial churn from cells continuing to grapple in the direction of the arrow even after they rearrange.

to introduce explicitly anisotropic properties. For example, cells may wish to differentiate between neighbors for communication and control purposes on the basis of their relative position, e.g. employing a planar cell polarization. One can also introduce anisotropic mechanical terms to the energy function, so that, for example, cells tend to be elongated. For both purposes, it is useful to maintain a planar cell polarization vector attached to each cell, to provide a local reference coordinate system.

In the model, a cell's polarization is maintained as a linear combination of vertex radial directions, so that the polarization rotates and transforms with the cell without any reference to global coordinates. Similar to traction forces, this representation needs to be updated whenever topological changes occur. I employ a crude interpolation method to produce a new linear combination whose orientation is as close as possible to the old. However, this has proven somewhat unsatisfactory, as these linear combinations respond unpredictably to non-uniform (i.e. nonlinear) deformations to the cell shape. It may be wiser to instead use the same update rule as traction forces, splitting and combining weights as vertices split and combine. However, how best to interpolate the weights initially, in order to align the polarization with some

external directional stimulus, remains an open question.

2.2.2 Three-Dimensional Mechanics

Now that we have an essentially complete 2D energy function, we can generalize to 3D behavior. Only one modification is required to the existing 2D terms: in order to apply the 2D energy function to a 3D polygonal hoop, we must define what "area" means for a non-planar hoop. I use the magnitude of the vector area. The vector area also furnishes a convenient normal vector, distinguishing apical and basal surfaces. Combined with the planar polarization vector and their cross product, we have a complete, local 3D coordinate system.

Next, we must add new energy terms to handle bending, contact forces, and undesirable, intracellular out-of-plane distortions. The most important is bending.

Bending

In the model, the bending of a cellular sheet involves both intracellular bending and bending at the junctions between cells. In a lattice where cells formed smooth, straight rows, such as a regular square lattice, it would be possible for lattice-aligned bends to be entirely intracellular or entirely extracellular. However, for a lattice that follows Plateau's laws (and is at least two cells wide), boundaries are inherently jagged, and bending requires changes in both cell shapes and junction angles, each bearing part of the load.

The model's cells already possess some intracellular bending resistance due to surface tension – the perimeter of a bent cell is greater for a given vector area than a planar cell. However, such resistance is weak, in addition to being inflexible: it is hard to specify flexural rigidity independent of shear stiffness, and it is hard to specify a natural setpoint angle, as might be produced, for example, by apical constriction.

Thus, I introduce an energy term for junctional bending resistance, which will generally carry the bulk of bending torques. Lacking detailed data, I model this very simply, as a linear torsion spring. Such a term takes the general form

$$E_B = k_B \mathsf{L}_{\mathsf{edge}} (\Theta - \theta_0)^2 \tag{2.12}$$

where Θ is the bending angle at the junction, θ_0 is the setpoint angle for the junction, and $\mathsf{L}_{\mathsf{edge}}$ is the length of the associated edge. One such term is

introduced for each edge in a cell. The scaling by $\mathsf{L}_{\mathsf{edge}}$ is critical, particularly so that bending energy is continuous across a topological interchange, where an edge shrinks to zero and then disappears completely.

To implement this term, we still need a means to calculate Θ, the angle between two cells, which is uniquely defined only when cells are completely flat. A reasonable first attempt for non-flat cells might be to use the angle between the cells' normal vectors. However, this measures not only the bending angle across the junction but also the skew angle parallel to the junction. Including skew angle causes serious difficulties when attempting to impose a nonzero setpoint curvature. I cancel skew approximately by projecting the cross product of the two normal vectors onto the edge vector. Additionally, it is critical to convert this to a four-quadrant angle, to prevent transient sharp bends from latching past the discontinuity at 90°. The resulting, somewhat Byzantine formula I thus employ is

$$\Theta = \frac{\hat{\mathsf{e}}_{\mathsf{ij}} \cdot \hat{\mathsf{n}}_{\mathsf{i}} \times \hat{\mathsf{n}}_{\mathsf{j}}}{|\hat{\mathsf{n}}_{\mathsf{i}} \times \hat{\mathsf{n}}_{\mathsf{j}}|} \arctan(|\hat{\mathsf{n}}_{\mathsf{i}} \times \hat{\mathsf{n}}_{\mathsf{j}}|, \hat{\mathsf{n}}_{\mathsf{i}} \cdot \hat{\mathsf{n}}_{\mathsf{j}}) \tag{2.13}$$

where $\hat{\mathsf{n}}_{\mathsf{i}}$, $\hat{\mathsf{n}}_{\mathsf{j}}$ are the two cells' normal vectors and $\hat{\mathsf{e}}_{\mathsf{ij}}$ is the unit vector along the shared edge. I do not claim that this is the simplest or best possible formulation of a junction angle, only that it works adequately, while simpler variants did not.

Finally, to create surfaces that are not naturally flat but have some intrinsic curvature, the cells' θ_0 values can be set away from zero. A uniformly positive value of θ_0 causes spherical curvature with outward-pointing normals, while a uniformly negative θ_0 causes spherical curvature with inward-pointing normals. Elliptical, cylindrical, and hyperbolic curvatures, however, require that θ_0 depend on orientation. To create such curvatures, I allow θ_0 to be controlled by two terms, a uniform curvature θ_{0u} and quadrupole term. The resulting formula is

$$\theta_0 = \theta_{0u} + \theta_{0q} \cos(2\Phi) \tag{2.14}$$

where Φ is a measure of the angle of the edge relative to some specified quadrupole moment axis in the normal plane of the cell. Setting $|\theta_{0u}| = |\theta_{0q}|$ then produces cylindrical curvatures, inward or outward, parallel or perpendicular to the axis, depending on the signs. The quadrupole axis is represented as a vector in the local coordinate system of the cell.

The preceding works well for gentle curvatures of any sign and spherical curvatures even of large magnitude but encounters some difficulties with

strong hyperbolic curvature. Spherical curvatures can be represented with flat or approximately flat cells by "cutting off the corners" (e.g. spheres represented as regular polyhedra). Hyperbolic curvatures, in general, cannot, at least without lining up cells in rows with high-order junctions (e.g. cylinders represented as ruled prisms), in violation of Plateau's laws.

With three-fold junctions, hyperbolic curvatures entail the presence of both skew angle and bending angle among the various edges.[2] Thus, junctional bending resistance confers flexural rigidity against hyperbolic bending. However, with four-fold junctions, it is possible to represent hyperbolic curvature exclusively in terms of bending angle or exclusively in terms of skew angle. With sufficiently strong hyperbolic curvatures, the unfavorability of four-fold junctions may be overcome by the unfavorability of large bending angles, driving the lattice towards a mesh of four-fold junctions among diamond-shaped, highly aplanar cells (see e.g. Figure 2.3).

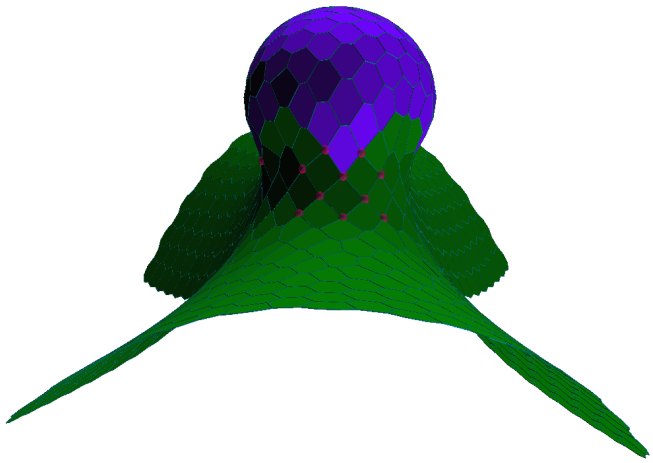


Figure 2.3: Rectangular array where a central disc of cells (lavender) have set $\theta_{0u} = 2/3$, bending the central region and its surroundings. In the middle of the view is a region of the neck with strong hyperbolic curvature, leading to stable 4-fold vertices (highlighted). $k_B = 3.2$

Whether or not this represents a real effect in hyperbolically-deformed

[2]The reason is that the traceless (hyperbolic) part of the local shape tensor can be diagonalized in one and only one orthogonal coordinate frame and off-diagonalized in one and only one orthogonal coordinate frame; non-orthogonal directions of travel necessarily must observe both bending and skew angles.

cellular surfaces such as embryonic epithelia is unclear. The answer depends on whether skew angle in the contacts between adjacent cells exacts a similar energetic penalty as the apico-basal compression and distension associated with bending angles. It's easy to imagine that skew angle would be much less stiff than bending angle, meaning that the effect observed here would be real, but I am not aware of any experimental data addressing this question.

If one wished to eliminate this effect, one could add additional bending terms to the 3D energy function to penalize cellular aplanarity or skew angle. I did not attempt to eliminate the effect, but I did wish to reduce the incidence of pathological aplanar distortion often seen in tight (nearly singular) hyperbolic curvatures, as these sometimes led to self-penetrating, tangled meshes, causing the energy minimization algorithm to stall. Thus, I added a high-order term to the energy function capturing "distortion":

$$\epsilon_{\mathsf{distortion}} = \mathsf{A_s}/\mathsf{A} - 1 \tag{2.15}$$

$$E_d = k_D \frac{A_0}{2} {\epsilon_{\mathsf{distortion}}}^2 \tag{2.16}$$

where A is the magnitude of the vector area and $\mathsf{A_s}$ is the total area of a fan of triangles covering the cell. To first order these will be the same, but with large aplanar distortions the magnitude of the vector area will be distinctly smaller, and thus distortion will be positive. The distortion energy term penalizes such aplanarity, helping to discourage mesh pathologies without strongly affecting the small-angle flexural rigidity.

3D continuum limit properties

The flexural rigidity of a regular hexagonal lattice can be estimated straightforwardly from k_B (though not calculated exactly, because of the contribution of intracellular bending). This allows us to estimate the thickness of an epithelium corresponding to the model parameters, showing that the default parameters give roughly "cuboidal" cells (thickness similar to edge lengths), and that one can model increasingly columnar epithelia by raising k_B.

For simplicity, we assume a uniaxial bend parallel to the edges of the cell and consider only the contribution of the parallel edges. Each of the two edges contributes $E_B = k_B \mathsf{L_{edge}} \Delta\Theta^2$ energy. The associated torque is

$$\tau = -dE/d\Theta = -4k_B \mathsf{L_{edge}} \Delta\Theta \tag{2.17}$$

and hence the flexural rigidity of a cell is

$$D_{cell} \approx 4k_B \mathsf{L}_{\mathsf{edge}} \mathsf{W}_{\mathsf{cell}} / \mathsf{H}_{\mathsf{cell}} \tag{2.18}$$

$$= \frac{8\sqrt{3}}{3} k_B \mathsf{L}_{\mathsf{edge}} \tag{2.19}$$

where $\mathsf{W}_{\mathsf{cell}} = \sqrt{3}\mathsf{L}_{\mathsf{edge}}$ is the width of a regular hexagon and $\mathsf{H}_{\mathsf{cell}} = \frac{3}{2}\mathsf{L}_{\mathsf{edge}}$ is the average height of a hexagon.

The flexural ridigity of a complete hexagonal lattice should be less by approximately a factor of two, because every bent edge is in parallel with the (weak) intra-cellular flexural rigidity of the adjacent cell. Equivalently put, every other row of cells does not contribute to flexural rigidity, because its bends could be assumed to be purely intracellular. Thus,

$$D \approx D_{cell}/2 \approx \frac{4\sqrt{3}}{3} k_B \mathsf{L}_{\mathsf{edge}} \tag{2.20}$$

Relating this value to the expression for the flexural ridigity of a uniform slab of isotropic, elastic material,

$$D = \frac{Y_{3d} t^3}{12(1-\nu^2)} \tag{2.21}$$

one can calculate an effective "thickness" t of the modeled surface

$$t = 4\sqrt{\sqrt{3}\frac{1-\nu^2}{Y} k_B \mathsf{L}_{\mathsf{edge}}} \tag{2.22}$$

$$= 4\sqrt{\sqrt{3}\frac{k_B \mathsf{L}_{\mathsf{edge}}}{2k_A A_r}} \tag{2.23}$$

and an equivalent cell aspect ratio

$$\alpha = 2\sqrt{3\frac{k_B \mathsf{L}_{\mathsf{edge}}}{k_A A_r^2}} \tag{2.24}$$

For default parameters, $\alpha \approx 0.55$. Note that α grows with the square root of k_B. Values of $\alpha > 2$, corresponding to $k_B > 10$ for default values of k_A, A_0, and k_P, represent highly columnar cells. Values of $\alpha < 0.25$, corresponding to $k_B < 0.16$, represent highly squamous cells, for which the model's assumption that intracellular rigidity can be ignored is probably a poor choice (and for which the mesh often behaves badly). The default parameters roughly model a cuboidal epithelium.

Contact forces

In 2D and especially in 3D, it is possible for the surface to meet itself as it deforms. In 3D, in particular, it is possible for the surface to meet itself face-to-face, far away from any free edges. In order to prevent self-penetration, it is necessary to introduce contact forces.

Because the level of abstraction used by the model omits the full 3D shape of the cells, I chose to implement contact forces very simply, as a finite-range repulsive force between cell centers. The range is set to be proportional to the square root of the lesser of the two cell areas (with proportionality constant 5/4 by default). At this distance, the contact force reaches zero, while at zero separation, it diverges. The form of the term is as follows:

$$E_c = k_C \min(\mathsf{A_i}, \mathsf{A_j})(D_r/\mathsf{D_{ij}} - 1)^2 \tag{2.25}$$

where $\min(\mathsf{A_i}, \mathsf{A_j})$ is the lesser of the two cell areas, $D_r = \frac{5}{4}\min(\sqrt{\mathsf{A_i}}, \sqrt{\mathsf{A_j}})$ is the contact force range, and $\mathsf{D_{ij}}$ is the distance between cell centers. This term is applied pairwise across all cells, excluding immediate neighbors. For efficiency, it is computed only for cell pairs within a factor of the contact range (determined using an octree index).

Using this simple form, self-penetration is still possible, because cell centers need not pass arbitrarily close to one another during penetration and hence can avoid the divergence. However, with appropriately large choice of k_C, I found that this worked very effectively in practice. More problematic is how it behaves with extremely tight bends: in principle, contact forces should prevent a surface from ever folding completely flat. However, since immediate neighbors are excluded from this formulation, it does not. On the other hand, including immediate neighbors here would interfere with the modeling of highly elongated cells. Thus, problems due to flat folds do arise occasionally, albeit uncommonly, because bending resistance already discourages such sharp curves.

For modeling some processes, it is important not to prevent self-penetration but instead to selectively facilitate surface fusion. For example, models of gastrulation, where a spherical topology becomes toroidal, require the fusion of two opposed surfaces. The lattice topological transformations required are nontrivial, however, and I have not implemented or tested the process.

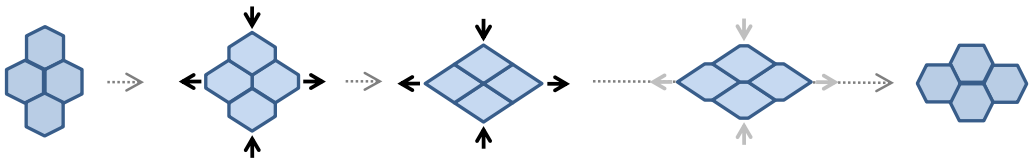


Figure 2.4: Detailed steps of a T1 topological interchange. From left to right: (a) Initial, unstressed configuration. (b) Elastically deformed configuration under shear stress, with edge lengths altered but vertex angles unchanged. (c) Metastable, degenerate configuration reached with increased stress. (d) Transient, non-equilibrium configuration as the cells autonomously rearrange into a new equilibrium. (e) Final configuration, stable even with stress removed.

2.2.3 Topological Interchange and Plastic Flow

When the cellular lattice is under sufficient in-plane stress, whether due to external forces, internal mismatches, or tractions exerted by the cells themselves, the lattice may distort to the point of permanent deformation. Permanent deformation occurs when an edge between two cells shrinks to zero length, becoming degenerate (middle frame, Figure 2.4). This configuration is metastable, because physically, a degenerate edge in the horizontal direction and a degenerate edge in the vertical direction are indistinguishable. With negligible provocation, the lattice should be expected to slide down into the neighboring equilibrium configuration (right frame, Figure 2.4), relieving a considerable amount of stress. Cells thereby exchange neighbors, and the lattice deforms hysteretically. Collectively over many cells, this mechanism provides for plastic deformation of the substrate.

In a mesh-based model, such physically obvious transformations as the transition between degenerate vertical edge and degenerate horizontal edge are not automatic. Instead, they must be encoded as explicit topological transformations applied to the lattice under specific trigger conditions. Failing to apply a necessary topological transformation typically leaves the lattice in a pathologically degenerate state, such that the energy minimization algorithm stalls until the problem is corrected.

The above topological transformation is known as the T1 process in the

foam physics literature [64]. In the model, I implement it in two steps: a *coalesce* step that joins two adjacent vertices into one of high-degree and a *split* step that splits one high-degree vertex into two of lesser degree. The simulator is not smart enough to analytically track the derivative of energy across these discontinuous transitions, so it simply tries them whenever they are feasible (checked once every 10 iterations) and reverses them if energy minimization fails to move the system away from the trigger configuration. A coalesce is triggered whenever an edge length drops below a particular programmed threshold, and a split is triggered automatically (after a few time-steps for relaxation) for any vertex of degree greater than three.[3] Preference is given to splitting in a direction distinct from the last coalesce operation at that vertex.[4]

Changes in cell number

In many scenarios, it is important to model cell division and/or death, often orchestrated as a deliberate part of an organism's developmental program. These can be implemented through straightforward transformations to the cellular lattice. Cell death is particularly easy: when triggered, the target cell is removed from the lattice, while any vertices it shared with surviving cells remain in place. As a result, a hole is left in the lattice. Under the influence of cell-cell adhesion (as captured by the difference in surface tension between an edge shared by two cells and an edge between a cell and empty space), such holes typically seal up quickly, so long as the lattice is not under strong tension. Neighboring cells expand to fill the gap, and the vertices surrounding the hole are merged together through the ordinary T1 process.

For cell division, the procedure is a little more involved. Cell division requires the insertion of new edges and vertices, in addition to the creation

[3]Default threshold is 0.05 length units, which, for typical system parameters, proved to be a reasonable compromise between too small (unreasonably hindering plastic flow) and too large (impeding elastic relaxation by injecting too much noise). On split, new vertices are spaced by twice this distance, providing some hysteresis margin.

[4]In circumstances where a high-degree vertex is stabilized, such as strong hyperbolic bending (as discussed in Section 2.2.2) or adhesive interactions that favor population mixing, the simulator repeatedly makes futile attempts to split the vertex in alternating directions, hindering the process of global energy minimization. To limit this, I implemented a per-vertex back-off counter, which, with each uninterrupted repeat of the same futile sequence of split and coalesce operations, doubles the number of time-steps that must pass before the vertex is considered for splitting again.

of a new cell. In general, a division septum will bridge across two previously unconnected edges, entailing splitting those edges and inserting two new threefold vertices. The cell can then be cloned and the vertices distributed as appropriate among the two daughters. Parameterizing this process are the choice of edges and the positions at which they are split. In the model, I generally assume cell divisions are symmetrical as far as area is concerned and so parameterize only on the orientation of the division septum. The edges closest to intersecting an imaginary line parallel to the specified axis (passing through the cell center) are selected for the division septum, and they are simply bisected at their midpoints.

Once a cell has been divided topologically, its mechanical parameters may need to be adjusted, depending on the situation being modeled. By default, the area setpoint of each daughter cell is cut in half. To model proliferative growth, the area setpoint should be restored to its original value. Alternatively, if cells are successively decreasing in size, it may be appropriate to adjust k_A, k_P, and k_B in order to maintain constant compressibility, shear stiffness, and flexural rigidity in spite of altered A_0; this is not done automatically.

2.3 Technical Details of Energy Minimization

Quasi-static mechanics can be implemented by energy minimization – relaxing the mechanical configuration of the surface to equilibrium between iterations of the cell agents. I have experimented with adaptive gradient descent, conjugate gradient, and several variant algorithms for energy minimization. In most cases, conjugate gradient is the fastest (dramatically so for large-scale bending) and hence is used for most of the examples here. A notable exception is systems that spend most of their time close to equilibrium and only occasionally experience perturbations, in which case gradient descent is faster because a single iteration of gradient descent (all that is needed when little has changed) is much simpler than a single iteration of conjugate gradient.

In practice, waiting until the minimization algorithm reaches a fixed point is too slow, and so convergence must be estimated by heuristics. Three ad-hoc metrics are tracked, all of which must fall below threshold: energy relaxation rate (mean over vertices), maximum per-vertex energy relaxation rate, and maximum per-vertex energy fractional relaxation rate. In the simulator, the

latter two are hard-wired together by a fixed scale factor, so the adjustment parameters correspond merely to global and local relaxation thresholds.

I have encountered two major pitfalls when relying on such loose heuristics for convergence. When the system passes through a buckling instability, progress may be extremely slow in the vicinity of the saddle point (especially under gradient descent), which can lead to premature detection of convergence even though the system is really very far from its steady-state conformation. More generally, premature declaration of convergence leads to a phantom "viscosity"-like effect, where points distant from a perturbation lag in responding to it. In the case of gradient descent, the effect is very similar to a drag force against a still fluid, where the viscosity increases with the convergence thresholds.

When cells are allowed to compute and actuate on such intermediate configurations, they can produce results that are unrealistic according to model assumptions and highly sensitive to choice of convergence thresholds. I have observed this particularly in the case of structures that are both growing and buckling at the same time (a common situation in organized proliferation patterning). Such simulations must be re-run with varying convergence thresholds in order to verify their reproducibility.

In the presence of saddles and multiple minima, energy minimization also introduces an ambiguity into the dynamics: which minimum? In the case of gradient descent, the minimum is chosen largely as though the system were governed by the still fluid drag described above. A real system, with inhomogeneities and internal drags, might pick a different local minimum. Conjugate gradient may pick yet another minimum, as its trajectories are visibly somewhat different from gradient decent. However, I have never observed the differences between conjugate gradient and gradient descent becoming a significant complication.

A related possible complication is the question of when topological changes occur – immediately as they are geometrically feasible, or at equilibrium, more slowly than elastic relaxation? I assume the former, which means that the trajectory followed, not just the minimum ultimately chosen, can affect which topological changes take place. In a real system, viscosity might affect which vertices interchange and which do not. In simulation, the choice of minimization algorithm might have an effect. However, I have observed that conjugate gradient and gradient descent are fairly consistent in plastic flow behavior, perhaps because in 3-space, accidentally bringing two vertices close enough to invert is both unlikely and energetically unfavorable.

2.4 Elementary Examples

2.4.1 Introductory Example

Consider the problem of evaginating a bud from an epithelial surface and growing it into a long tube. One way to achieve this goal is to elect an organizer cell to represent the tip of the tube and emit a signal instructing its neighbors to proliferate. We will need to construct a procedure for all cells that, fed information about the cell in question and its neighbors, produces the actions that need to be taken by the cell. In this case, the procedure will need several different behaviors – leader election, signaling, proliferation, and quiescence – depending both on the location of the cell and on the global state of the computation. However, it must select its behavior solely on the basis of local information – signals emitted by immediate neighbors, prior state recorded in the cell, and local environmental information.

All cells begin symmetrically, in the leader election phase. On the basis of random noise, individual cells will unilaterally proclaim themselves leaders at a very slow rate, recording this in their internal state. Such a leader emits a message indicating that leader election has completed, which is rebroadcasted by any cell receiving it. In this way, the system permanently leaves the leader election phase and enters the growth phase.

In the growth phase, the leader emits a second signal, indicating that its neighbors should proliferate. This signal is not rebroadcasted. Any cell receiving the signal adopts the proliferating behavior, and any cell not receiving it adopts the quiescent behavior, even if was previously proliferating. In addition, all cells continue to propagate the message indicating that leader election has finished, so that stragglers are properly informed.

Proliferating cells increase their sizes and periodically divide along alternate axes. The increase in size may be gradual or instantaneous, but division is always discrete and instantaneous. As a result of either effect, cells may be displaced, coming in contact with, or more often, sliding away from the organizer. Similarly, the daughter cells are identical except for location and geometry, meaning that both or possibly only one will be in contact with the organizer. Thus, the cohort of proliferating cells changes over time, though remaining roughly constant in number, while newly quiescent cells accumulate in the surrounding neighborhood.

As new, full-size cells accumulate in the neighborhood of the organizer, mechanical stress builds. Cells must increasingly compress in order to fit

among their neighbors; this compression propagates outward through the sheet in the form of (2D) hydrostatic pressure. Meanwhile, pre-existing cells farther away are stretched circumferentially as the interior of the sheet expands. The resulting circumferential uniaxial tension acts to contain the interior hydrostatic pressure, and so the two stresses decline with distance from the organizer.

Stress builds over time, and limits are soon reached. Uniaxial tensile stress is limited by the ability of cells to slide past one another when pulled or sheared; such stress cannot increase beyond the plastic yield limit. With increasing pressure at the organizer, rings of cells farther away begin to yield, thereby relieving some of the pressure but driving yet farther rings into tension (Figure 2.5a). If such progressive yielding reaches the open edges of the sheet, the sheet enlarges while remaining flat. If any edge of the sheet is slightly weaker than the others, that edge will yield and thin preferentially, until it is unable to support the circumferential tension. The organizer and a cohort of new cells will then bulge out through the weak spot and grow laterally.

Hydrostatic pressure, on the other hand, is limited by buckling instabilities. When sufficient pressure covers a sufficiently large area, determined by the flexural rigidity of the surface, the surface abruptly begins to deflect in one or the other normal direction. Pressure can now be relieved by expansion perpendicular to the surface, and so further growth feeds the growth of the buckle rather than an expanding front of plastic yielding. This leads to the formation of a dome centered on the organizer, roughly reflecting the present size of the zone of plastic yielding. Plastic yielding is now confined within the lateral edges of the dome, and the dome grows into a tube (Figures 2.5b, 2.5c).

This is an admittedly simplistic agent program – multiple leaders may be elected due to signal propagation delay, a leader that is killed will not be replaced, and the direction of tube apex growth wanders randomly under variations in growth rate and cell rearrangement. Open boundary conditions for the sheet are also not always appropriate. Nonetheless, the scenario illustrates how geometrical patterning behavior depends crucially on the mechanical properties of the substrate, in this case flexural rigidity and yield strength. Nowhere in the agent program is the idea of a tube explicitly specified, nor is the diameter encoded at all.

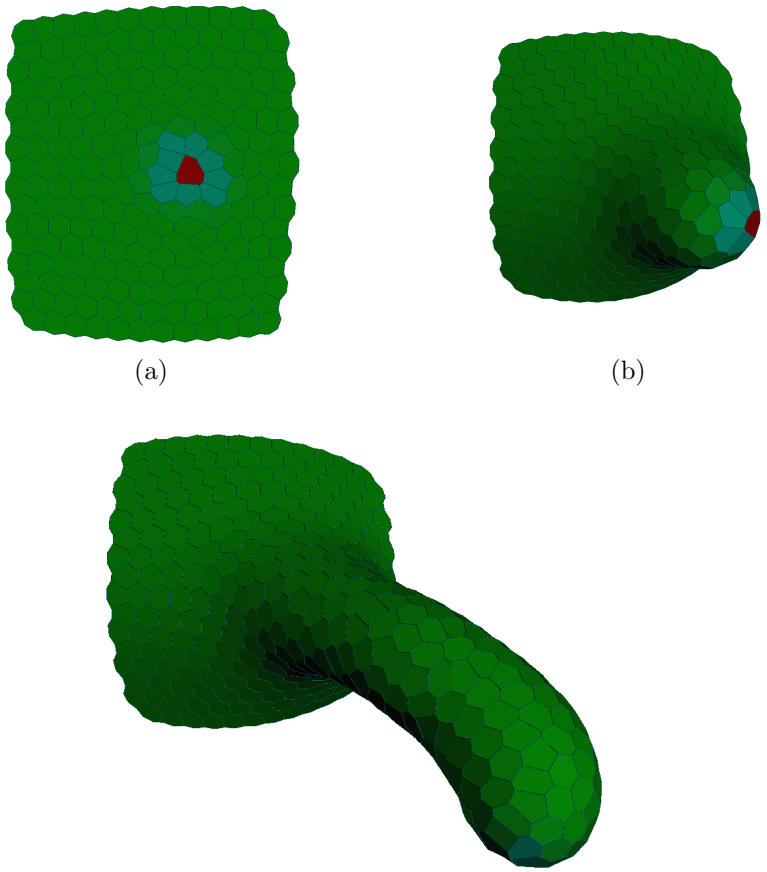


Figure 2.5: (a) After randomly electing an organizer (red) within a rectangular array, neighboring cells have begun dividing, causing local strain to the lattice but not enough to form a buckle. (b) Soon after, more growth has caused buckling. (c) Elongating tentacle beginning to drift due to fluctuations in proliferation rate and plastic relaxation.

2.4.2 More Examples of Organized Proliferation

As a first elaboration, let's consider how to make the tentacle grow towards some target rather than wandering randomly. In this toy example, the only control mechanism available is the proliferation rates of the cells adjacent to the organizer (unlike, for example, plant shoots, which can also transmit

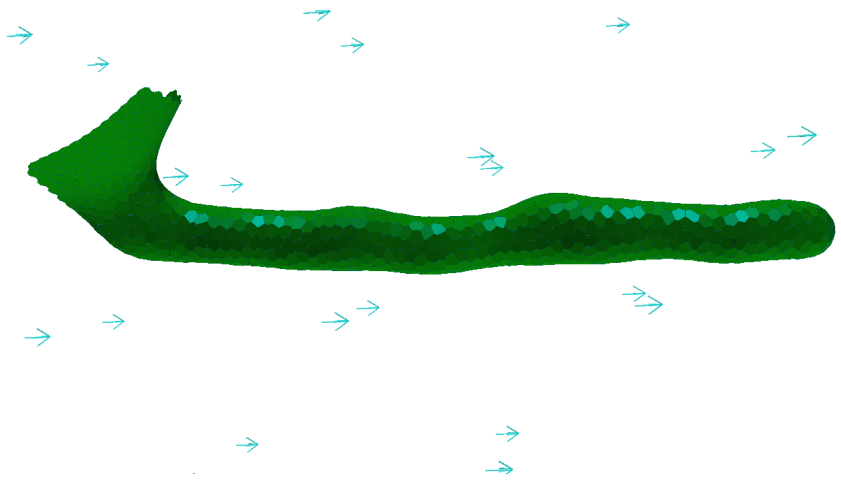


Figure 2.6: Tentacle growing along a signal gradient within the ambient medium (indicated by arrows) via proliferation rate control.

signals down the shoot and elongate existing cells far back from the apical meristem in response to light/dark signals).

How can proliferation rates be used to control the direction of growth? When cells proliferate unequally, the tip tends to bend away from areas of greater proliferation. If the tip is seeking to align with some ambient gradient, then cells that are already aligned with the gradient should grow less so that the organizer deflects in their direction, and cells that are badly misaligned should grow more. If cells can measure the orientation of the gradient relative to their apical surfaces, a simple control law would be to scale the proliferation rate by one minus the cosine of the angle (i.e. the dot product of the cell's normal vector with a unit vector in the direction of the gradient). This is quite effective, as seen in Figure 2.6.[5]

Such a solution does have the potentially unwanted side effect that the overall growth rate also varies, depending on tip orientation and other factors. If multiple, rate-controlled processes are operating concurrently, such rate skew can distort the resulting shape. This problem can be avoided by pacing total apical growth rate in some way and having cells compete for their shares. For example, the organizer could emit diffusible growth tokens which are

[5]Of course, in this simple example, the direction of the original buckle is still indeterminate; if it happens to buckle the wrong way, the tip will then grow *away* from the gradient – correctly aligning the upside-down organizer with the gradient.

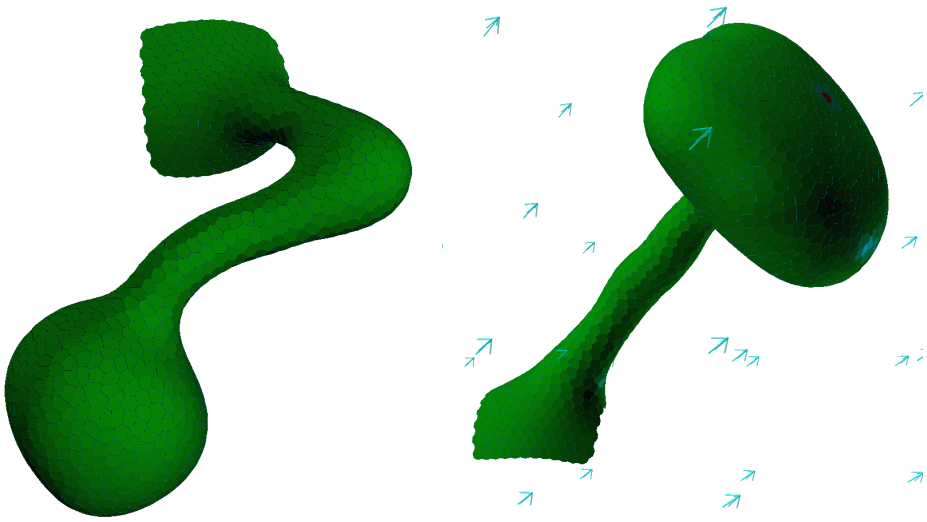
competitively consumed by its neighbors depending on their desired growth rates. Another possibility might be a priority scheme where cells naturally cycle, but those with higher priority cycle slightly faster and preempt their neighbors' divisions.

Other interesting variations are possible on this theme of appendages by patterned proliferation. For example, the tube diameter can be varied by allowing a larger annulus of cells around the organizer to proliferate. Alternatively, allowing cells away from the apex to continue proliferate at a slow, basal rate enlarges an existing tube. Operating concurrently with apical growth, the result is a tapered tentacle.

As another example, when the tentacle has grown beyond a certain length (as determined, e.g., by the decay of a diffusive signal emitted from the basal cells), the apical cells can be configured to change their behavior. If they begin proliferating on their own even when no longer adjacent to the organizer, the result is a ball on a stick (Figure 2.7a). By modulating the growth rate using orientation information provided by an ambient gradient, as in the chemotaxis mechanism above, other shapes can be formed. For example, maximal growth rate perpendicular to the gradient produces a disc (Figure 2.7b).

Tentacles can also be made to branch, by generating new organizers laterally or by allowing the apical organizer to divide. In the latter case, the paired daughter organizers are initially clustered together, contributing to a single growing apex, but eventually they are sheared apart by the proliferation of their neighbors and subsequently repel due to proliferation between them. Independent buckles can then form, leading to separate daughter tentacles (although complete separation in the face of curvature-reducing plastic flow may take some time).

Putting several of these mechanisms together, one can model the growth of a branching tree network exhibiting allotropic scaling [68, 61]. Branching tentacles with paced apical growth naturally produce an expanding tree, but how should one ensure that basal branches are scaled appropriately for the size of the subtrees beneath? If network growth were uniform and steady, basal branch cells could simply continue to proliferate according to some blind, pre-calculated schedule. However, natural networks like roots, shoots, hyphae, blood vessels, and nerves tend not to be uniform but instead respond to environmental cues, growing and expanding where they are most needed. In such cases, proper scaling demands that signals (possibly physical, such as blood flow rate) back-propagate information about subnetwork size up the



(a) (b)

Figure 2.7: Tentacles where apical proliferation behavior changes for cells sufficiently distant from the base. (a) Non-chemotactic tentacle with a ball produced by promiscuous apical proliferation. (b) Chemotactic tentacle with a disc produced by orientation-sensitive proliferation.

tree. We must therefore construct an agent program that back-propagates some sort of load signal.

A straightforward, botanically-inspired solution is to transport a conserved "hormone" flow along the tentacle walls, sourced by the apices and sinked by the root of the tree. The net flux down a tentacle is then proportional to the number of apices it supports. a simple Laplacian diffusive flow suffices, with sources at the apices and Dirichlet boundary conditions at the root (although for large networks it helps to use an out-of-band fast Laplace solver rather than relying on the agents to compute by explicit relaxation at a painfully slow pace). The tentacle wall cells can then slowly and independently proliferate until the hormone flow density falls to some target value. A fixed target produces a network with constant cross-sectional perimeter (not constant cross-sectional area, because only the surface carries hormone – Figure 2.8a). Other scaling laws can be produced by using a target that is varied as a function of local cylindrical curvature, as an estimate of tube di-

ameter. For example, constant cross-sectional area is produced with a surface flow density target proportional to diameter (Figure 2.8b).

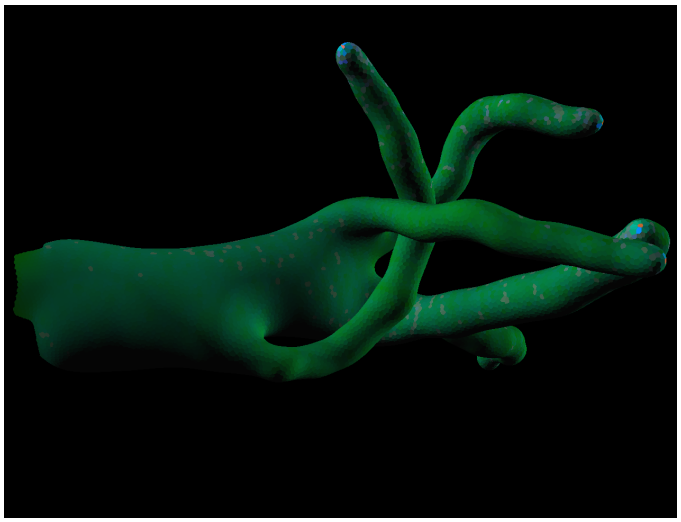
2.5 Related Models

Attempts to gain insight into morphogenesis through mechanical models have a long history, dating from His's classic 1874 experiments with a rubber tube as a model for a chick embryo [33], to modern computational finite element simulations (e.g. [14, 16, 52]) and the recent, "multi-scale" simulation work of Brodland et al. [11, 13], which shares several key insights with my deformable surface model. Contrary to the common sentiment that every particular detail of a living organism must have been selected for some specific evolutionary advantage, many of these works aspired to find simple, over-arching physical mechanisms that could explain a broad swath of observed forms merely from physical principles (e.g., [48, 34, 53], and most eloquently [61]). My work explores a middle ground, neither bare physical simplicity nor exhaustive descriptive detail, in search of new, practical insights.

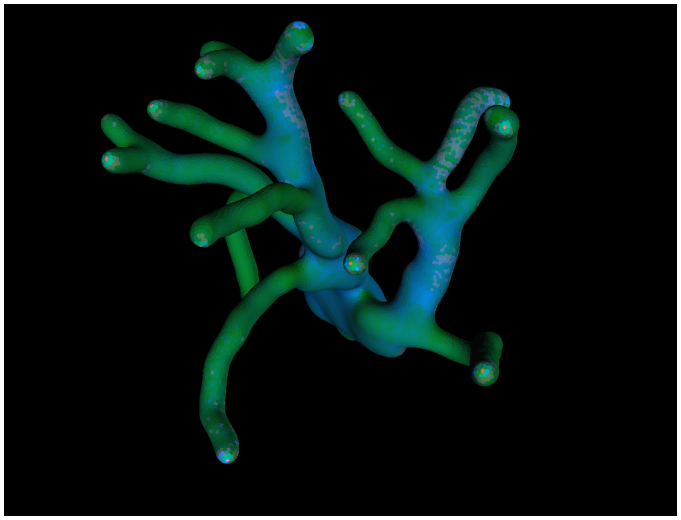
Several investigations in recent years have also used the 2D topological abstraction for epithelial cells, including T1 transformations, e.g. [46, 10, 9], yielding insights into the mechanical and topological behavior of epithelia.

In the synthetic biology community, researchers have begun exploring 2D patterning systems in engineered bacteria (e.g. [4]), driving the development of models such as Gro [35]. These models necessarily incorporate 2D mechanical deformation, because bacteria in culture are constantly proliferating and filling up the available space, although it is something of an interesting nuisance rather than an essential part of patterning. Doursat et. al. have developed several simple morphogenesis models of a similar character [17, 19], based on point cells with spring interactions, which they have used to explore simple approaches to simultaneous patterning and deformation, locomotion control, and morphogenesis-inspired engineering design.

A number of models have explored a pure, rule-based approach to deformation and development, without reference to mechanics. L-systems [49], a grammar model for plant growth, are the classic example. In the tightly-coupled world of animal development, however, context-freedom or a restriction to one-dimensional contextual interactions is unreasonable. The most direct inspiration for this work is Nagpal's origami-based self assembly [45], which models caricatures of morphogenesis using flat folds in non-deforming



(a)



(b)

Figure 2.8: Allotropically-scaled, randomly branching tentacle trees. (a) Secondary growth is triggered by the density of hormone flow from the apices, leading to constant perimeter scaling. (b) Secondary growth is triggered by an estimate of flow per unit area based on local surface curvature, leading to scaling with approximately constant cross-sectional area.

2D sheets. More recently, the MGS language [25, 56] attempts to capture fully general contextual interactions with topological rewriting rules, formalizing it as the "Dynamical Systems with a Dynamical Structure" approach. One could likely use such a tool to assist in building a similar surface model. However, in a physical system such as developing tissue, the structural dynamics are not easily expressed in terms of simple local rules. Instead, structural updates are mediated by mechanical interactions, which, though technically local, are quite complicated and effectively nonlocal on the relevant timescales. Thus, I emphasize mechanics as the core mechanism, not topological dynamics.

Chapter 3

Self-timed patterning

3.1 Introduction

In centralized systems, timing is often taken for granted. Clocks are global and consistent. Tasks block until they are completed. Combining tasks into sequences is straightforward, implicit in every imperative language.[1]

In spatially distributed systems, however, time is slippery. Time is relative, due to communication delay [38]. Clocks are decentralized, conflicting, and often drifting. Inputs may be available in different places at different times. Tasks, distributed spatially, complete in some places sooner than in others. The speed of computation may not even be consistent.

In spatial computing, and in particular developmental patterning, the problem of time is ever-present. The complications due to timing are often sidestepped through a mixture of ad-hoc, optimistic assumptions about the timing properties of the substrate or by pursuing problems that naturally lend themselves to asymptotic convergence rather than particular results at definite times.

For example, the Growing Points Language [15] assumes that spatial gradients (e.g. Bellman-Ford distances) may be trusted once they are locally detectable. This is potentially race-prone if multiple sources appearing in different places must be considered, because their influences may arrive at different times, but even in the common case of a single advancing source, it assumes that spatial distance and propagation time are comparable, or that

[1] This chapter originally appeared as a submission in the 2014 Spatial Computing Workshop [12].

propagation time is completely negligible. Temporal irregularities such as a slow patch of cells will lead to transient errors in magnitude and direction as the gradient navigates around the irregularity. Even though the gradient eventually corrects itself, because it is hard for a growing point to correct a path already traversed, such transients may lead to permanent irregularities.

The Origami Shape Language [45], on the other hand, has many features that are naturally "feed-forward", in that transients will ripple through the calculations and eventually be replaced by the final, steady-state values, because the computation has no persistent memory in which to retain transient errors. In time, the outputs will be correct. However, the language does include some operations that require convergence to be achieved beforehand, such as the repurposing of reusable communication channels, and so at a certain point it must stop waiting and accept the answers. The amount of time to delay is calibrated empirically, based on propagation time measurements during start-up using test gradients. The wait is configured to be an overestimate based on the longest possible propagation distance. So long as timing is uniform and consistent, this is reasonable, albeit wasteful. However, it will fare poorly if timing properties vary unpredictably or if the domain size can enlarge.

When such ad-hoc assumptions are unjustified, and when the structure of the computation does not naturally forget transient perturbations, the problem of timing can no longer be ignored. In particular, when irreversible or difficult to reverse actions must be taken on the basis of the results of spatial computations, it is imperative that the results be accurate and stable beforehand. This sort of difficulty arises particularly often in developmental patterning, where disruptive and destructive primitives such as cell division, death, and topological rearrangement must be choreographed along complex, dynamically constructed spatial patterns. Moreover, development time is often precious, and so hard-coding adequately conservative convergence delays may be completely unreasonable. This problem has been acknowledged only rarely (e.g. [17]) and investigated even less.

To illustrate the magnitude of the problem with a natural example, consider the early development of the chick embryo. At the beginning of incubation, it consists of a mass of largely undifferentiated cells. Within about 24 hours, it has finished gastrulation, begun its neural folds, and assembled its first somite and the beginnings of a notochord [30]. Yet, given the embryo's size on the order of a few millimeters, the diffusion time to establish a typical morphogen field is expected to be on the order of an hour, and several hours

more to reach a close approximation of steady state [54, ch. 3]. With multiple, sequential developmental steps crammed into such a tiny span, there's little time to waste. Furthermore, timing skew is acutely evident, with the head of the embryo maturing far in advance of the tail.

One can imagine two general approaches to the problem of timing in spatial patterning. One might design the distributed computations such that correct convergence can be detected by inspecting the output provided. This can be nontrivial, because the verification process must be significantly simpler and more temporally tractable than the original problem being solved or no progress has been made. Once correct values have been identified, they may be checkpointed and passed onto disruptive and irreversible actuation processes. This might be compared loosely with the manner in which common bilaterian animal body plans such as in drosophila develop, with successive cascades of gradients and compartment patterns preceding large-scale morphogenesis. Alternatively, one could design the overall algorithm such that even though individual steps are effectively irreversible, the large-scale dynamics follow a self-stabilizing trajectory, and so excursions due to acting on erroneous transient values will eventually die out like any other perturbation. This is perhaps reminiscent of the way such exemplary regenerating animals as the hydra are thought to maintain their body structure.

In this chapter, I demonstrate a simple methodology for designing spatial patterning algorithms whose local completion status is implicitly indicated by the output values themselves. Inspired by the self-timing methodology in clockless digital circuit design, I term this technique self-timed patterning. The key insight is the use of partially informative data representations, which monotonically increase in precision until the true answer is indicated. I show how they can be used to pattern disruptive, irreversible transformations both quickly and safely, completely robust to timing pathologies.

3.2 Self-Timing

When a system is decomposed into multiple, simultaneously executing modules, the modules must communicate in order to coordinate their behaviors, and their communication protocols must confront the problem of timing. A self-timed system, loosely speaking, is a system whose modules communicate through a type of asynchronous protocol such that no assumptions need to be made are made about the latencies of the modules themselves [55].

Self-timing naturally lends itself to race-free designs, and additional steps are often necessary to allow data races. The key insight is that that data signals must indicate, explicitly or implicitly, when they are ready for use, and once they have done so must remain fixed until acknowledged. General self-timed circuits must also treat the acknowledgement signals with similar care in order that circuit elements may be reused for multiple waves of data, but for our purposes, focusing on single-use developmental cascades, we can usually omit the acknowledgement pathways entirely.

The simplest approach to self-timed signaling is to bundle the data signals with a boolean completion signal that is asserted once the data outputs can be trusted. This is known as "bundled data" and it is used successfully, although care must be taken to avoid timing skew between the arrival of the data and completion signals. However, an alternative scheme, known as "dual rail", turns out to be more natural and insightful for our purposes. In classic dual rail signaling, two separate signals are provided for each bit, one indicating whether it is true and one indicating whether it is false. This scheme can articulate four possible status values per bit: true, false, unknown, and contradiction (generally unused). For multi-bit data values, the individual bits may become valid at different times, but once a bit becomes valid it will not become invalid (at least until the next acknowledgement cycle).

Many variations on dual-rail-style self-timed signaling are possible, but they share a common insight: glitch-free partial information about the answer accumulates monotonically over time, beginning with complete uncertainty and ending with an unambiguously specified value. This is the concept of monotonic partial information championed by Propagators [58, 51], and single-use self-timed systems can be represented most clearly as propagator networks.

When data values are not limited to raw digital signals but can include analog values or compound data structures, more complex partial information structures may be used, such as interval arithmetic. Modules need not wait until their inputs have reached complete convergence, either; they may compute partially informative answers based on partial input information. When an actuator determines its inputs are sufficiently precise, it may act on them even before final convergence. (If, however, such actuation may disrupt the upstream computations by triggering non-monotonic changes that may contradict the existing outputs, then the inputs must be checkpointed first and the computation disabled; this is discussed in Section 3.4.2.)

3.3 Simple Computations

The simplest sort of self-timed computations are those that depend only on local information, for example, taking local averages or detecting local maxima of a field. Such computations can simply wait for all their inputs to be available, locally and from immediate neighbors, and then produce an output. If a partial set of inputs or partially informative inputs are available, a partial output may also be computed early, if desired. For example, if the number of neighbors expressing a signal is to be counted, and two have responded with the signal, three without, while one response remains to be received, the output count can be expressed as the interval $[2, 3]$. When the final response is received, the output is narrowed to an exact answer. On the other hand, sometimes partial inputs are sufficient to produce a complete output. For example, the boolean OR of binary inputs is known to be true as soon as the first positive response is received.

Implicit in this formulation is that cells must be able to enumerate their set of neighbors or otherwise determine when a complete set of responses has been received. This is natural for cells in physical contact with one another but is a slight departure from the classical amorphous computing model [1].

3.3.1 Gradients

Most spatial patterning algorithms are not so simple, relying on long-range propagation of information from cell to cell. However, the same principle can be applied, since simple functions are the building blocks used to compute the outputs that are shared with cells' neighbors. To illustrate, let us construct a self-timed version of the Bellman-Ford hop count gradient algorithm, one of the most useful algorithms in discrete spatial patterning.

Ordinary Bellman-Ford works using a single communication channel, where each cell broadcasts its best known distance to a source, or zero if it itself is a source. Each cell iteratively replaces its best known distance with $\min_{n \in neighbors} distance(n) + 1$. Eventually, distances converge to steady-state values, but there is no local indicator of convergence.

To reformulate this as a self-timed computation, we must properly express what is known in the form of monotonic partial information. The best known distance is just that—an upper bound. If we represent distance as an interval, we can include a lower bound as well. With no information, a cell must report that its distance is in the interval $[0, \infty]$. At each local iter-

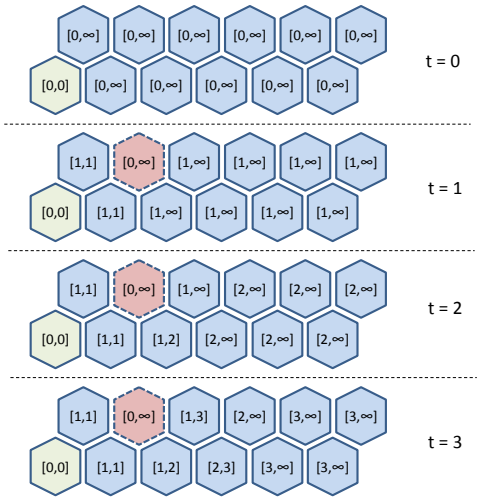


Figure 3.1: Four iterations in the computation of a self-timed gradient, with source cell at the left (green) and one uncertain or stalled cell (red).

ation, sources broadcast zero for their distance, while non-sources compute $\min_{n \in neighbors} distance(n) + [1, 1]$ and broadcast that. In interval arithmetic, the minimum value of a set of intervals is an interval spanning from the minimum over the lower bounds to the minimum over the upper bounds. As cells discover paths to sources, the interval upper bound always falls. As cells learn that successive rings of neighbors are not sources, the interval lower bound steadily rises. The interval thus brackets the actual distance between progressively narrowing bounds. When the two bounds meet, local convergence has been achieved. The time required relative to Bellman-Ford is essentially unchanged.

The same gradient algorithm works whether there is one source or many. If the input to the algorithm—the map of sources—is determined programmatically, it too must be represented as monotonic partial information. In this case, each cell may be source, a non-source, or undetermined. An undetermined cell always reports the lower bound of its distance as zero and hence will be surrounded by a "well" of cells with uncertain distance intervals, waiting to determine whether they're sitting right near a source (see Figure 3.1). Once the cell's source status is finally determined, the uncer-

tainty disappears.

Cascading gradients and simple local computations, we have a fairly powerful patterning toolkit along the lines of [18], only now completely self-timed.

3.3.2 Growing Points

Another useful patterning tool is growing points [15]. Growing points are virtual excitations that move about a substrate according to programmable tropisms, tracing out paths. With self-timed gradients to direct their tropisms, we can construct self-timed growing points fairly easily, though some limitations will arise, affecting the tropisms that can be practically implemented.

The output of a growing points computation is a field of "secretions" left behind in the growing points' wake. For a complete self-timed formulation, we must know not only where the secretions are, but where they are not. Thus, we must be able to determine both the paths traversed by the growing points and the places that, at the conclusion of propagation, they will not have traversed. The presence of one or another determination indicates that, locally, the computation has completed.

The forward propagation of a growing point is straightforward. The cell hosting the growing tip waits until sufficient tropism-relevant information has been posted by all of its neighbors and then nominates the most favorable among them as a successor. Upon noticing the nomination, the designated successor changes its status from "unknown" to "in path" and performs its own nomination process. Secretions are then a local function computed on the path status.

In order to compute the complete secretion field, cells that will never be in the path must ultimately know to change their status from "unknown" to "not in path". The constraints used to generate this information necessarily relate to the global behavior of the growing point (and of all other indistinguishable growing points from the same family). For example, if the path has a definite termination condition and it is known that there will be only one trace of that family, then upon reaching the termination condition the growing point can broadcast a globally propagated completion message, signaling to all remaining "unknown" cells they are actually "not in path".

With more detailed information about the tropism, however, more prompt constraints can be used. For example, a lone, orthotropic growing point climbing a gradient can broadcast each level of the gradient passed; all "unknown" cells below that gradient level can conclude they are not in the path.

More locally, any growing point family known not to self-intersect can back-propagate not-in-path status; any "unknown" cell whose preferred successor is in path must not be in path. Additionally, any cell whose preferred successor is not in path must also be not in path. Together, these naturally cover many cases, with the exception of diatropism (propagation parallel to the contours of a gradient), where back-propagation must be bootstrapped e.g. by emitting perpendicular orthotropic growing points to turn off neighboring cells.

As a consequence of the fact that secretions are a function of partial information not complete until the growing point has exited the area, it is also generally not possible to use "auto-tropism", sensitivity to a gradient emitted by a trace's own secretions. Thus, other techniques must be used for purposes such as inertia and preventing diatropic traces from doubling back on themselves. Restricting successors to be non-neighbors of one's predecessor or favoring candidates farthest away from the predecessor are useful alternatives [44].

3.4 Putting the Pieces Together

3.4.1 Composition

The composition of two self-timed computations is naturally another self-timed computation. In the simplest case, the downstream function merely waits for its inputs to arrive before generating an output; more generally, monotonic partial information propagates from one end to the other. As a result, local computations, gradients, growing points, and other self-timed computations can be cascaded seamlessly.

For example, a local computation based on the orientation of per-cell polarization vectors identifies which cells are on the left edge of the substrate and which are on the right (Figure 3.2a). Left edge cells are then used as the source for a left gradient; right edge cells are used as the source for a right gradient. The difference in value of the two gradients is used to divide the substrate into two compartments (3.2b). Within each compartment, similar computations can be nested recursively (3.2c). Note that this can converge even faster than the traditional feed-forward equivalent, because the gradient lower bounds allow compartment determination at the far ends well before a signal can arrive from the opposite end's source.

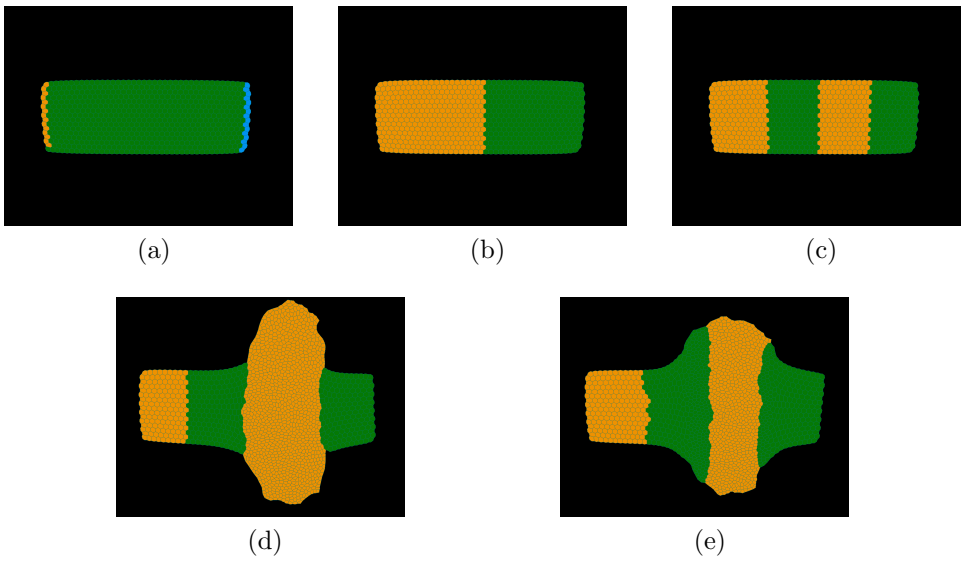


Figure 3.2: 3.2a-3.2d: Steps in patterning a crossbar. 3.2e: Crossbar patterned without checkpointing inputs prior to cell division and adhesion changes.

Since self-timed computations must know which neighbors to expect responses from, nested computations must be made aware of which cells belong to the same compartment and hence should be expected to participate. Like the input arguments, this information must be provided in a self-timed fashion.

3.4.2 Actuation and Checkpointing

Eventually, however, it comes time to act based upon the patterns established. Some operations, e.g. cell differentiation, are merely *irreversible*. These operations may be performed locally as soon as the necessary inputs are locally available.

Other operations, such as cell division and motility, are *disruptive*, causing non-monotonic changes to the properties and neighborhoods of cells. These must be treated with greater care, as they may cause the very inputs on which they depend to change, changing their own behavior and giving rise to race conditions in neighboring cells. To accommodate these in a self-timed computation, a checkpointing step must be employed.

Checkpointing is a process that serves to hand control from one stage of a patterning cascade to the next, where the stages would interfere if they ever overlapped in time. Checkpointing can be local to a cell or coordinated across a wider region, depending on the needs of the computation. Checkpointing entails halting the previous stage of computation (or at least preventing it from having further side effects), since subsequent results may no longer be trusted, and saving its outputs for future reference. This includes saving intermediate values shared with neighboring cells, because neighbors of a cell performing a checkpoint may not yet have completed the computation preceding the checkpoint and will need access to all their neighbors' broadcasts. Once prior computations have been halted and their outputs recorded, locally or over a sufficiently wide radius, the next stage of computation may be launched—e.g. a disruptive actuation process. Through checkpointing, disruptive actuation becomes a form of irreversible actuation.

For example, in Figure 3.2d, the cells in the third compartment are programmed to change their adhesive properties to reduce their affinity for the remaining cells and to undergo three rounds of oriented division. This leads to the formation of a crossbar. In the absence of checkpointing, the resulting rearrangement and increase in cell number leads to changes in gradient values and hence changes in the boundaries of the compartment, leading cells to reassess their fates. Some cells that have already divided will conclude that they do not belong in the crossbar after all, while other cells may be freshly recruited, dividing and changing their adhesive properties. The results of such churn in this example are shown in Figure 3.2e, where values were allowed to converge but checkpointing was disabled; the extent to which this matters varies, but here it yields a much sloppier result. Without an indication of a stage's completion such as provided by self-timing, checkpointing is impossible, on top of the hazard of actuation based on premature results.

When disruptive effects are merely local, checkpointing and actuation can be local as well. Neighboring cells may still be continuing to work on the prerequisite computations, relying on checkpointed copies of the cell's broadcasts. On the other hand, when the disruptive effects are non-local, for example, by exerting forces sufficient to cause neighboring cells to rearrange, a barrier computation may be needed to ensure that all cells within the radius of potential disruption have produced and checkpointed their outputs before the disruptive stage is allowed to begin. In the example above, a short-range barrier would probably be warranted (although in practice omitting it produces acceptable results). After actuation, barriers may be used again to

verify completion of the disruptive steps, and then another wave of patterning can be initiated.

If only nearest neighbors are disrupted, the barrier simply waits for all neighboring cells to indicate completion. More generally, an arbitrarily ranged barrier can be constructed by the Bellman-Ford lower bound calculation: completed values are treated as non-sources, and incomplete values are treated as unknown. As nearby input values converge, the distance lower bound rises monotonically. When the distance to a source is known to be greater than the desired range, all cells within that range must have completed. The time required for such a barrier computation is proportional to its range.

When the effects of actuation are predictable and consistent, one can reasonably "prepattern" and checkpoint a map of actuations to be performed, then perform them—patterning by dead reckoning. One can expect to achieve a particular final configuration, albeit with some difficulty in specification given that the actuation map itself may be distorted by actuation. If, however, instability, noise, variation in cell layout, and other uncertainties lead to the actuation being not only disruptive but also *unpredictable*, such naive prepatterning can make only small changes before accumulated errors make a mess of both the structure and the prepattern. This sort of difficulty often arises, for example, when cells rearrange under stress or after division. Note how in Figure 3.2d the border of the crossbar is quite irregular.

In the presence of such unpredictability, a single round of patterning followed by actuation may be insufficient. Feedback measurements must then be used to correct the system towards its goal. A second set of barriers is necessary to verify the completion of the actuation, and then a round of re-measurement and corrective patterning can begin, driving a round of corrective actuation. This process may be repeated several times, if necessary. Unlike traditional closed-loop feedback, where error measurement and actuation take place simultaneously, self-timed feedback control must cycle through discrete stages (potentially in the form of traveling oscillations when timing skew is present).

3.5 Self-timed vs. Self-correcting

In the preceding, we have been exploring the technique of static prepatterning—constructing a spatial template and then performing irreversible and disruptive operations based on that pattern, in order to achieve some desired prod-

uct. Self-timed patterning has played a valuable role, facilitating aggressive actuation that neither jumps the gun nor trips over itself by destroying its own inputs. Multiple stages of prepatterning and actuation can be cascaded, facilitated by the barrier mechanism. Results can be used as soon as they are available, with no need to wait for a conservative estimate of worst-case convergence time. Self-timed patterning and partial information thus improve the speed, robustness, and composability of the patterning process, compared to mechanisms that rely on pre-programmed delays or ad-hoc assumptions about timing.

In prepatterning, the target pattern is not a stationary point of the control algorithm. Indeed, careful steps must be taken to prevent the control algorithm from recursing on its outputs. However, prepatterning runs into difficulties when faced with unpredictable operations. Unpredictability demands closed-loop feedback to correct errors, and such feedback is most easily phrased in terms of recursion seeking a steady state. Furthermore, in the face of *asynchronous* damage (e.g. unexpected rearrangement that may occur during a patterning stage rather than merely during disruptive actuation), self-timing becomes only approximate; answers may be forced to change non-monotonically or to remain stale. Ultimately, the entire notion of prepatterning breaks down if serious damage can occur at any time, independent of the state of the computation. It is possible in this case to create self-timed, self-correcting loops, but it is not obvious whether self-timing really aids in solving the problem anymore.

In the next two chapters, I investigate the opposite solution to the timing problem, self-correcting patterning, particularly as applied to traction-driven cell rearrangement. In this approach, cells apply forces to their neighbors in order to incrementally rearrange themselves towards the desired pattern. A self-correcting template defines the layout of the pattern within the substrate, while each region of the pattern thus designated runs a closed-loop control algorithm to enforce desired local features. The results are similarly robust to timing pathologies, but in many other ways they are dual to self-timed prepatterning: convergence is slow and indeterminate; completion is difficult to ascertain, and it is not obvious how to compose sequential stages. Even if convergence can be reliably identified, sequential composition is likely to interfere with the self-corrective ability of earlier stages. On the other hand, unpredictability is assumed, not avoided, and damage naturally heals.

3.6 Conclusion

Self-timed prepatterning and closed-loop self-correction represent two extremes, both useful, both with some precedent in nature. Self-timed prepatterning is fast and composable; self-correction is robust and regenerative. I have demonstrated how the monotonic propagation of partial information facilitates convergence detection and checkpointing, crucial for implementing prepatterning. On the other hand, unpredictable operations, noise, and asynchronous damage degrade prepatterning, to the point that it can be come unviable. Bridging the gap between static prepatterning and self-correction remains the goal of ongoing work.

Chapter 4

Normal neighbors patterning

How might one assemble a creature from un-differentiated tissue or regenerate a missing piece of pattern destroyed by injury? One of the common themes observed in developmental biology is the Rule of Normal Neighbors [43]: a point in a patterned tissue knows what elements of the pattern belong adjacent to it, its "normal neighbors". If it finds its neighbors are wrong, it will take steps to correct the situation, such as regrowing a more appropriate neighbor or changing its own fate to better fit its environment. This general rule captures many striking experimental results, such as the regeneration of additional, backward segments in cockroach limbs when the distal portions of their limbs are excised and replaced with excessively long explants [23]. Though the regenerated leg pattern is dramatically incorrect, the discontinuities are gone.

The simple, above statement of the rule leaves an important question unanswered, though: How does a cell know what action to take? Even ignoring the question of whether to regenerate new cells or re-differentiate old cells (which I have made little progress elucidating), there are often multiple possible new pattern elements to choose from. The neighboring environment may not uniquely identify what is missing, or it may be over-constrained and contradictory, demanding corrections. One possibility, suggested by Mittenthal, is that that cells might attempt to intercalate the shortest path through pattern space (under some metric) to bridge discontinuities, but with their limited resources, how would cells even perform such a complicated, nonlocal computation as a shortest-path search in an abstract space?

I propose one possible mechanism by which patterning and pattern repair under the rule of normal neighbors can be implemented, with minimal

and purely local computational resources. I represent the topology of a desired pattern as an adjacency graph over discrete pattern states. Using this graph, I show how to construct a potential function using local interactions for which the desired pattern is (usually) a minimum. Cells can then explore this potential by a process mathematically analogous to thermal annealing, seeking a minimum. The resulting algorithms show promise for engineering use in amorphous computing and synthetic biology and may also help elucidate the mathematical and dynamical properties of pattern formation in vivo.

4.1 Adjacency Graphs

The core idea of the rule of normal neighbors is topological: what can lie adjacent to what. It turns out that this, alone, is not generally enough to form a pattern, but it's a good starting place. For simplicity, we represent the regions of a pattern as discrete states, a "color-by-numbers" abstraction of what might really be continuous variation. Each region is assigned a state, and the adjacency graph captures the neighbor relationships between homogeneous regions of a single state. An implicit self-edge exists for each state, in order that the representation be scale-invariant and meaningful both in continuum and on a discrete lattice.

Example desired pattern:

```
11112222
11112222
11155222
44455333
44443333
44443333
```

Adjacency graph:

Obviously, the template pattern from which an adjacency graph was constructed satisfies the adjacency graph. However, a variety of other patterns will as well. Indeed, any arbitrary continuous distortion (diffeomorphism) of the original pattern will satisfy the same adjacency graph. Also, any simply connected (but possibly overlapping) cut through the pattern, even a spiral or a wild squiggle that repeatedly cut through the same regions, will still satisfy the adjacency graph, albeit with some regions and neighbors duplicated and others absent.

An assortment of other patterns satisfying the same adjacency graph:

```
11111111 11111111 11111111 11112222 11115555 11222111
11111111 22222222 22111122 11112222 11115555 45525545
11111111 33333333 22255222 45555552 11122555 44333444
11111111 22222222 33355333 35555553 22222333 45525545
11111144 33333333 33444433 33334444 22233333 11222111
11112553 55555555 44444444 33334444 22233344 45525545
```

In order to avoid arbitrarily pathological deformations, we might use patterning algorithms that favor compact, blob-like regions and insist that patterns be broken up into convex regions only (requiring non-convex elements of the pattern to be broken up into approximately convex sister sub-regions). Such a preference turns out to be easy to implement, as a kind of "surface tension".

To avoid the problem of arbitrary cuts with missing regions, we might impose boundary conditions (assuming there are boundaries, absent in the spherical and toroidal topologies common in early embryos) that force every region contacting the boundary to appear in the pattern. This will ensure that at least all boundary-contacting regions are present, and that no region that should not be cut by a boundary is. (Indeed, it even lets us weaken the convexity requirement a little, by allowing non-convex, boundary-attached regions, as long as the boundary conditions are sufficient to determine the general shape.) Alternatively, or additionally, we could demand that every region be uniquely named, and rely on algorithms that strongly favor the existence of exactly one contiguous instance of each region. Although this does encounter some complications, it can be accomplished by including a quorum sensing mechanism.

There remains the issue of duplicated or absent neighbor contacts, even when all regions are present in the correct numbers. It turns out that this is already disfavored in most cases by surface tension, but it can arise pathologically in some scenarios, particularly around poorly-specified boundary conditions and undesired local minima. One might be able to augment region quorum sensing with quorum sensing for each neighbor-neighbor contact pair. Alternatively, one might be able to add interlayer regions (themselves quorum sensed) between all true regions. I have not explored these possibilities.

4.2 Potentials and Probabilities

Given a starting configuration, how should cells determine what changes to make in seeking a pattern that fits the given criteria (i.e. the adjacency graph)? One would like a representation that points the way but avoids limit cycles and inertia. A plausible choice is the minimization of an energy function on a bounded configuration space, e.g. by gradient descent. Moreover, for suitably-constructed energy functions, gradient descent can be implemented purely locally, with local information and local actuation, a crucial trait for distributed implementation. However, spatial patterns are especially prone to getting locked in local minima; gradient descent fails almost immediately.

The failure of gradient descent suggests that perhaps a probabilistic strategy is in order. Energetically favorable transitions can be made with high probability, and unfavorable transitions can be made with low probability; this allows a gradual, stochastic search of the landscape, unimpeded by small hills and ridges. Interestingly, with appropriate choice of probability as a function of energy difference, this becomes quite analogous to the process nature uses in forming crystals, ferromagnetic domains, and other lattice-based patterns. By computing probabilities weighted according to the exponential of the energy reduction divided by some constant, which we'll call "temperature", we get a Boltzmann-weighted probabilistic automaton. This may have a very close physical correspondence with how differentiated cells exhibiting differential adhesion sort themselves into organized clusters [29] (although perhaps more complicated than needed [57]). However, I will show how to use it as an abstract computational mechanism for differentiating cells in the first place.

The algorithm sketched thus far works fairly well. However, it is noisy. As a stochastic exploration with no termination condition, it will continue rattling about the space of reasonable solutions indefinitely, showing fluctuations from the ideal and a fair amount of drift. One solution is to gradually lower the temperature to zero, "annealing" and freezing in the pattern. This is a useful strategy which I will address in more detail later, but it has some limitations. As an alternative, can we formulate a "thermodynamic limit" to the stochastic theory, where fluctuating discrete states are replaced by mean field values? Indeed we can, and this appears to be the most useful and biologically plausible version of the algorithm.

4.3 Constructing the Energy Function

On a lattice or other cellular tiling, we can define the energy function to be minimized as the sum, over all the cells of the domain, of individual contributions from each cell. Based on the principles above, the following terms are used:

- For each neighboring cell, an "adhesion" energy if and only if the neighboring cell's state has an edge connecting to the local cell's state in the abstract adjacency graph. For the special case of a neighboring cell with the same state as the local state, a larger energy bonus used; this gives rise to surface tension. The adhesion term is averaged over neighbors rather than summed, to limit effects due to variation in number of neighbors.

  We write this as: $-\frac{1}{|n_i|}\sum_{j\in n_i} U(S(i), S(j))$

  Where n_i is the set of spatial neighbors of cell i, $S(i)$ is the state of cell i, and U is the adhesion energy matrix, equal to the abstract adjacency graph's adjacency matrix plus a surface tension constant k_s times the identity matrix.

- A quorum sense energy, reflecting the rarity of the local state across the whole domain. This is a nonlocal measure, and so to be computed in terms of local information, it must be re-expressed in terms of some information sharing mechanism. A diffusion/decay process (i.e., screened Poisson equation) works well for scale-invariant quorum sensing, provided the screening length is significantly larger than the domain size. If each cell emits one diffusible token per unit time when in a given state, and all cells absorb and destroy tokens at a rate proportional to their local concentration, the token concentration will converge approximately to the fraction of all cells in the given state. The screening length, the rough maximum distance at which cells can sense quorum members, is determined by the ratio of the token diffusion rate to the token destruction rate.

  With a quorum level q_i in hand, we can construct an energy penalty associated with being in that state. I happen to have used minus its reciprocal plus some anti-divide-by-zero softening, scaled by a gain parameter constant. This works, and has the interesting property that

gain grows dramatically with dwindling numbers. There might be better choices, however.

We write this as: $-k_q/(q_s(i) + k_{soft})$

Where k_q is the quorum sense gain constant, q_s is the quorum sense level for state s, and $k_{soft} << 1$ is the softening constant.

- A hysteresis energy, which adds an artificial tendency to keep the the same state. This is not strictly necessary; in the stochastic case, it merely serves as a means to adjust the update frequency. However, a possible generalization under the mean field theory does have some useful influence on stability and can be used to inject potentially useful hysteresis behavior.

  We write this as: $-k_h\delta(S(i), S_{-1}(i))$

  Where k_h is the hysteresis constant, S_{-1} is the previous state of the cell, and δ is the Kronecker delta.

The sum of these terms over all cells constitutes the energy function. Observe that the difference in energy due to changing a cell's state can be computed locally by the cell, knowing only the quorum levels and the states of its immediate neighbors (at least, if quorum sensing is exact rather than an approximation).

4.4 The Stochastic Algorithm

In the stochastic version of the patterning algorithm, at every timestep, each cell (possibly with some probability) randomly picks a new state according to Boltzmann distribution. That is, each cell computes the change in energy ΔE for each possible state and then weights the probabilities of new states by the Boltzmann factor, $e^{-\Delta E/T}$, where T is the temperature parameter.

In practice, for ease of computation and to facilitate strict locality, we make a slight approximation. Instead of computing the change in global energy for each possible new state, cells compute the local energy contribution for each state given the previous neighbor states and quorum sense levels. Cells then weight states using $e^{-E_i/T}$. This is quantitatively very similar, but it ignores the local and global impact of the state change on quorum

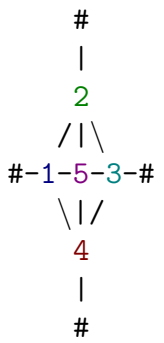
levels (usually tiny) and it undercounts adhesion energy by a factor of two, effectively scaling the constants by 1/2 (only approximately, if cells have varying numbers of neighbors). This difference in formulation is reflected in the analytical work below. I don't expect this makes any practical difference in behavior, but I have not investigated.

4.4.1 Examples

For simplicity, these examples use global quorum sensing rather than screened Poisson quorum sensing. Essentially the same results can be achieved either way, provided care is taken that the screening length is much larger than the domain size and that appropriate numerical finesse is used so that accurate solutions are produced in a reasonable amount of time. (Poor accuracy thresholds can, in some cases, systematically inject artificial energy into the system, preventing convergence.)

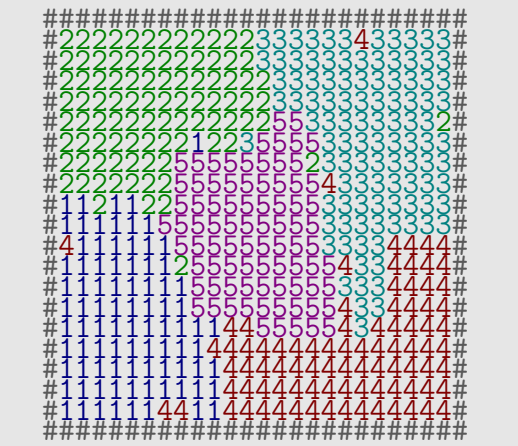
Example 1

Graph:



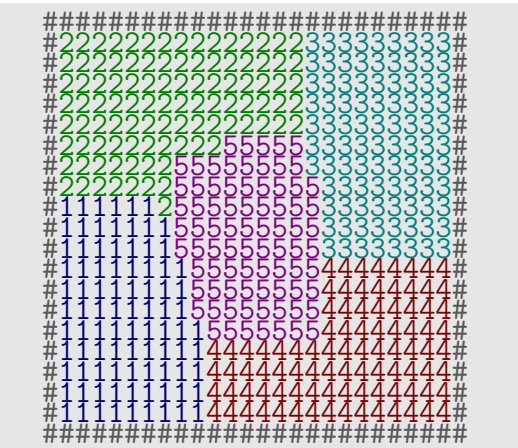
The '#' symbol is special marker indicating domain boundary (sensed as if it were a neighbor state). 1-4 are boundary-compatible states, while 5 is boundary-incompatible (more on this later).

Representative snapshot: ($T = 0.125$, $k_s = 0.5$, $k_q = 0.0125$)



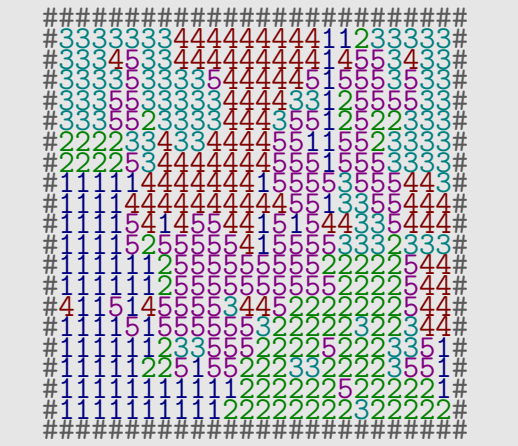
Notice how the pattern is clearly realized as specified by the adjacency graph, albeit with somewhat irregular boundaries. Also note the presence of small noise inclusions (e.g. the lone '4's and '2's embedded in the 3 and 1 regions). Both the irregularities and the inclusions are transient consequences of thermal churn; every few timesteps, they disappear and are replaced anew.

Colder snapshot: (T reduced to 0.0625 from 0.125)



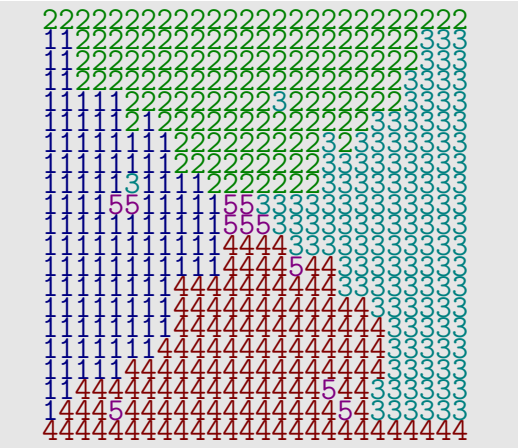
Notice the smooth boundaries and the complete absence of noise inclusions. The pattern is cleaner and much more stable in time, showing smaller fluctuations. Quickly producing such a clean result depends on cooling from a higher temperature, however. Starting out with temperature this low can lead to slow convergence, with a variety of long-lived defects.

Hotter snapshot: (T increased to 0.175 from 0.125)



Here, the pattern has almost disappeared, though its rough shape is still somewhat visible. Correlation over time has not completely vanished yet, but thermal churn is rapid and intense.

Snapshot with fixed 1/2/3/4 boundary conditions and quorum sensing disabled: ($T = 0.125$, $k_s = 0.5$, $k_q = 0$)



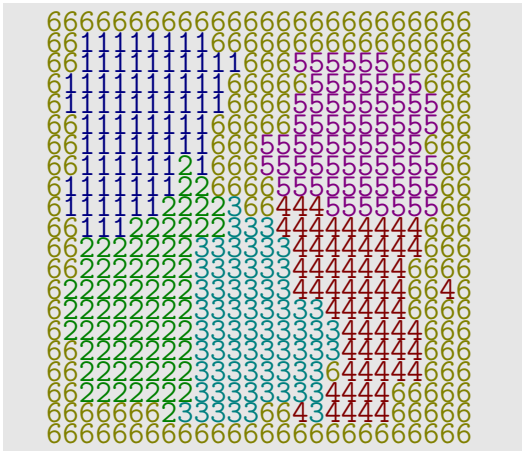
Notice how being anchored to the boundaries gives states 1-4 a tremendous advantage, while 5 nearly disappears. If the boundaries are not fixed, in the absence of quorum sensing, states typically disappear one by one until only one fills the entire domain.

Example 2

Graph:

Fixed boundary conditions at 6.

Representative snapshot: ($T = 0.125$, $k_s = 0.5$, $k_q = 0.0125$)



Notice how the 1-5 chain pattern curls up to fit more comfortably into the approximately square domain, while the non-convex 6 region attached to the boundaries acts like a fluid bath. The chain pattern will tend to straighten out in a wider domain or if anchored to opposite boundaries, but otherwise it curls under the influence of surface tension trying to contract individual regions into more symmetrical shapes.

4.5 The Mean-Field Algorithm

To avoid the noise of the stochastic algorithm without incurring the curse of dimensionality associated with the complete probability distribution over configurations, we can construct a mean-field theory. This will discard correlations between cell states but preserve the uncertainty inherent in a poorly constrained choice of state. It may also be more biologically plausible, being based on continuous variables rather than discrete states with discrete

updates, and it offers some rough hints for how final convergence may be detected.

Instead of a distinct, discrete cell state S, we assign each cell a vector p_i of "probabilities" over the different states, representing the degree to which a cell is committed to each of the states. Then, in computing the local energy contribution for an individual, discrete state, we compute the expected energy given the prior state probability vectors. We use these expected energies to compute what the Boltzmann-weighted state probabilities should be locally, and we relax current probabilities towards those values, repeating until convergence.

Using matrix notation for p and U, the mean field adhesion energy is thus

$$\langle E_{adhesion}(i) \rangle = -\frac{1}{|n_i|} \sum_{j \in n_i} (\mathbf{p}(i)^\mathsf{T} \mathbf{U} \mathbf{p}(j)))$$

The hysteresis energy term has to be reinterpreted somewhat, since in continuous time there is no notion of "previous state". Instead, it most plausibly becomes either a constant expected value (boring) or a quadratic nonlinear term reflecting how well committed the cell is to a given state. The latter has some impact on stability, so it is worth exploring. Its contribution to the mean-field energy takes the form $\sum_{s \in states} k_h p_s(i)^2$ – i.e., $k_h p_s(i)$ in the pure state energies.

So, in steady state, the system must satisfy

$$p_s(i) = e^{-E_s(i)/T} / \sum_{t \in states} (e^{-E_t(i)/T}) \tag{4.1}$$

$$E_s(i) = -k_h p_s(i) - \frac{k_q}{q_s + k_{soft}} - \frac{1}{|n_i|} \sum_{j \in n_i} (\hat{\mathbf{e}}_s^\mathsf{T} \mathbf{U} \mathbf{p}(j))) \tag{4.2}$$

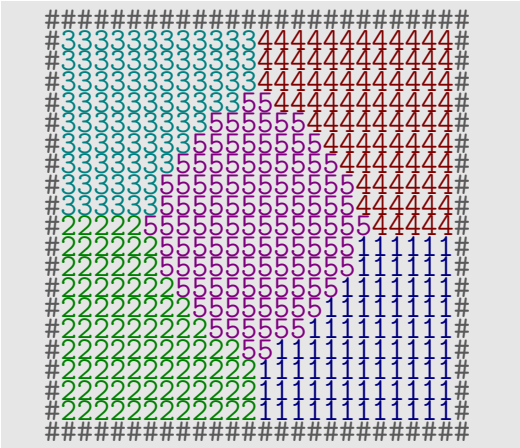
where $\hat{\mathbf{e}}_s^\mathsf{T} \mathbf{U}$ is the s-th row of $\mathbf{U}$.

The only remaining question is how we tabulate quorum population q_s. The obvious answer, using the sum of all cell probabilities for a given state instead of the count of cells in that state, turns out to work poorly in practice. Regions tend to be indistinct, not clearly stabilizing on a preferred location. To combat this, we can inject a positive nonlinearity into quorum measurement. I found that squaring the state probabilities before summing worked well. The rest of quorum sensing, including using a screened diffusion process for tabulation, is the same.

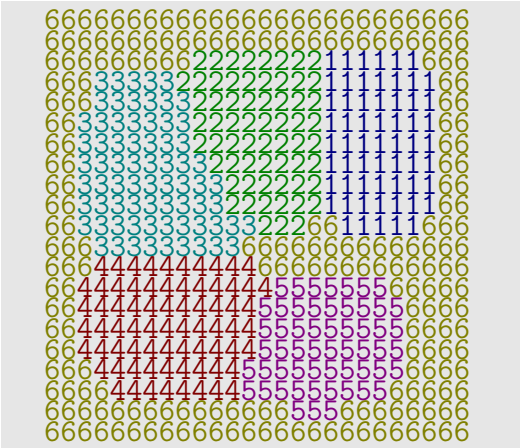
With judicious choice of relaxation step size, trading off speed for convergence stability, we now have a complete mean field algorithm. Unlike the stochastic algorithm, it has a smooth trajectory and a clear steady state. The particular choice of steady state solution is determined by initial conditions, through a symmetry-breaking process. With appropriate choice of temperature, the results are often quite good.

4.5.1 Examples

Maximum likelihood states in representative steady-state solution to Example 1 under mean field algorithm: ($T = 0.1875$, $k_s = 0.5$, $k_q = 0.0125$, $k_h = 0.125$)



ML representative steady-state solution to Example 2 under mean field algorithm: ($T = 0.1875$, $k_s = 0.5$, $k_q = 0.0125$, $k_h = 0.125$)



4.6 Problems and Local Minima

The previous sections presented some encouraging results, and indeed, the desired pattern is almost always a robust attractor for the system, but convergence is not necessarily guaranteed, and complications do arise. In the stochastic algorithm, improper configurations may appear during convergence and re-appear intermittently at later times. In the mean-field algorithm, successful convergence is generally permanent, but the pattern may also become permanently trapped at a local minimum.

Convergence is particularly sensitive to temperature. At low temperatures, convergence is slow, and numerous, long-lived (even permanent) topological defects can appear. At higher temperatures, the pattern becomes smoother and less sharp-edged (ultimately indistinct in the stochastic algorithm), while defects (apart from stochastic noise inclusions) are rare and few in number, most disappearing quickly. At sufficiently high temperatures, however, bulk symmetry never breaks, and convergence never occurs; states share the interior of the domain equally, and one could say the pattern has "melted". The complete melting temperature depends mainly on energy function parameters and is minimally influenced by the adjacency graph and boundary conditions. The sharpest, most defect-free patterns are produced by starting with a temperature above the melting point and slowly reducing it to a temperature well below. The parallels with phase transitions,

crystallization, and annealing are obvious.

Just shy of the complete melting point, however, portions of a pattern may appear melted, with certain unbroken or weakly broken symmetries between states, while the overall structure remains. Partial melting is a pattern-dependent phenomenon, occurring where internal, boundary-independent symmetries need to be broken. The effect arises mainly in two circumstances, either when the domain size is excessively small or when several identical, mutually contacting states (sharing all neighbors and all parameters) must break symmetry. For the effect to be severe, three or more states must be involved. In such cases, hysteresis is also often observed. Note that none of the examples above are susceptible (no two states are identical); some examples will be provided below, along with analytical insights from linear stability analysis.

Once fully solidified, the kinds of defects characterizing poor convergence fall into a few common patterns. Completely missing regions are very rare (unless quorum sense is disabled), as are erroneous contacts between incompatible regions. Instead, an apparent improper contact will typically have a thin sliver of states spanning the gap, duplicate regions reduced to minimal size but unable to disappear completely. This often occurs in the context of a "twist": a 4-cycle (or larger) in the adjacency graph in which two different segments have chosen opposite chirality. For example, if quenched too rapidly,

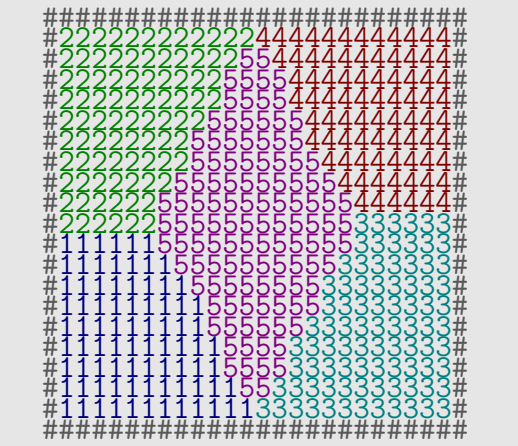
```
   2
  /|\
 1-5-3
  \|/
   4
```

may converge to

```
   2
  /|?
 1-5-4
  ?|/
   3
```

where the question marks indicate improper or ambiguous contacts. In practice, these "contacts" are usually either occupied by thin slivers of an intervening state (1 or 3 in the upper right and 2 or 4 in the lower left), or they are four-fold junctions, where, for example, 2, 4, a distorted 5, and the boundary all meet at a single point. The latter situation is illustrated below.

ML plot of twisted solution to Example 1: ($T = 0.1875$, $k_s = 0.5$, $k_q = 0.0125$, $k_h = 0.125$)



Notice how the 5 region is vertically distended and how the four-fold contacts at the top and bottom are different from the ordinary three-fold contacts on the left and right. Such a solution occurs perhaps one in three times for this pattern started at the above parameters with no annealing.

A closely related defect is twinning. Here, regions or entire segments of the pattern are repeated, possibly with the same chirality, more commonly with flipped chirality. In this case, redundant regions are stabilized, probably geometrically, and do not shrink down to slivers. The precise causes that can allow twinning to persist stably remain unclear. Twinning is often seen as a defect when twists are impossible due to absence of vulnerable 4+cycles (particularly when partial melting hysteresis disrupts annealing) or if the algorithm is extended to include chiral preferences at 3-way junctions, but it also seems to appear with some frequency simply when patterns become sufficiently large and complicated. Twinning is also a common symptom of insufficiently long screening length under screened Poisson quorum sensing.

Twisting and twinning are the most common convergence-related defects, and are amenable to cure by careful choice of temperature or by annealing. They typically arise when convergence proceeds by domain wall migration: temperature is low enough that symmetry breaks quickly and regions rapidly stabilize and saturate, so any reduction in defects takes place by slow movement of the boundaries between regions. This is a highly local process, and global defects such as twists, exhibiting baked-in symmetries, are largely immune to domain wall migration. Twinning is often eliminated by domain

wall migration, but the result may be a twist rather than a correct pattern. Alternatively, twins may be stabilized if the twinned region has floating subregions that are partitioned among the two halves (because neither twin can be continuously eliminated).

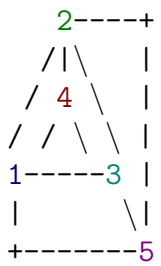
On the other hand, if temperature is high, symmetry breaking works globally, in the weakly unstable, sub-saturation regime. Just slightly below the melting point, symmetry-breaking is extremely slow, but twisting and twinning are largely absent from the result. Typically, with careful annealing from a melt, twisting and twinning can be avoided completely.

Domain size, interestingly, also seems to have a strong effect on twisting and twinning. Small domains are more forgiving about temperature and tend to break symmetry completely in the sub-saturation regime for a wide span of temperatures below the melting point. Large domains, however, must be kept closer and closer to the melting point in order to achieve a clean, global symmetry break before saturating and transitioning to domain wall migration. There is, in some sense, a correlation length associated with a given temperature. In practice, precisely picking the right temperature to achieve fast, defect-free convergence for the whole domain may be impractical. More robust strategies for preparing large domains include annealing slowly from a melt and growing slowly from a small domain size.

Example 3

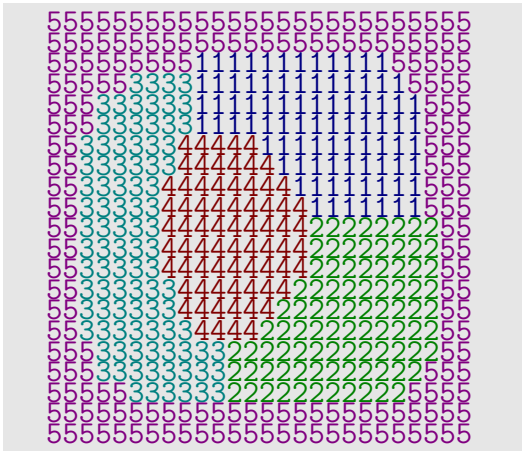
A pattern immune to twisting but prone to fairly clean twinning. (Other patterns, such as example 1, will twin under the same temperature conditions, but in complicated ways that often involve a mix of multiple twinning and twist defects.) This pattern is also notable for showing partial melting with hysteresis in the 1-2-3 ring (three identical states) and hence is somewhat difficult to anneal, probably contributing to the twinning.

Graph:

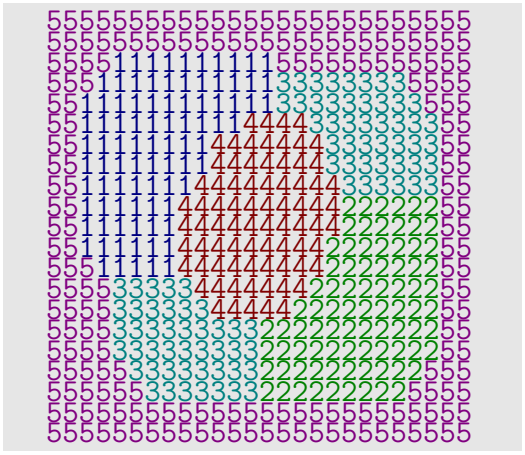


Fixed boundary conditions at 5.
Representative solutions: ($T = 0.1325$, $k_s = 0.5$, $k_q = 0.0125$, $k_h = 0.125$)

Correct:



Twinned:

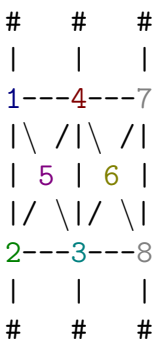


Note the duplication of region 3.

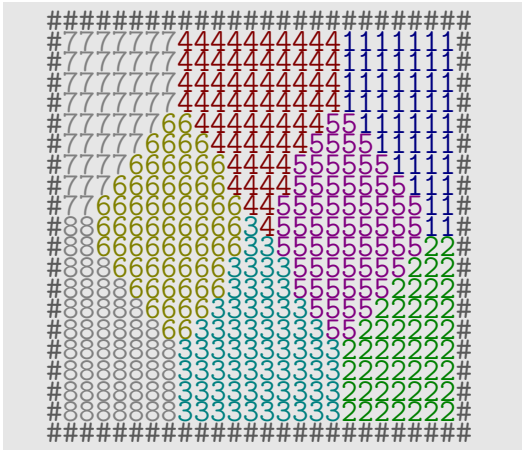
Example 4

A more complicated pattern, with several twistable cycles.

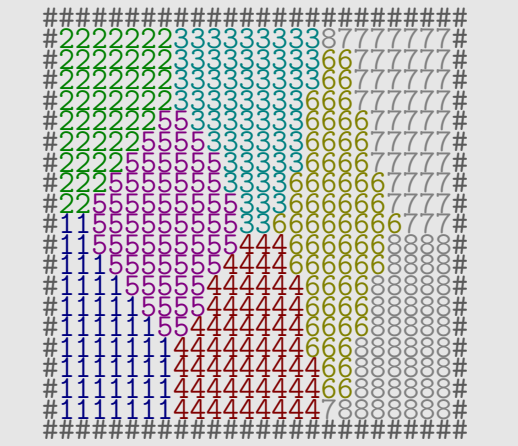
Graph:



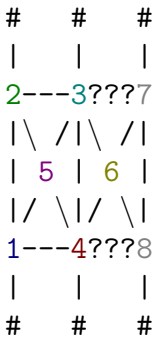
Representative steady state annealed from melt in two steps:
($T = 0.375 \to T = 0.25 \to T = 0.1875$)



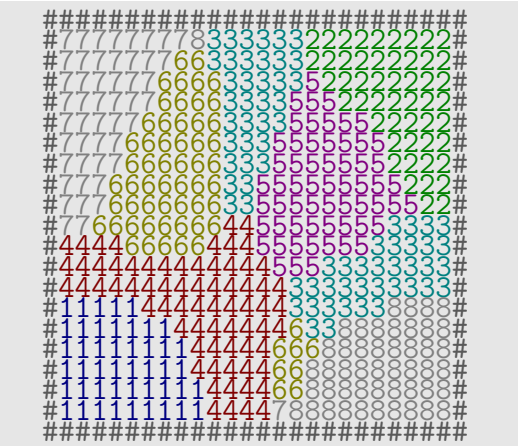
Annealed from melt in one step: ($T = 0.375 \to T = 0.1875$)



Note the single twist defect (with marginal sliver twinning), corresponding to an erroneous topology:



Without annealing:

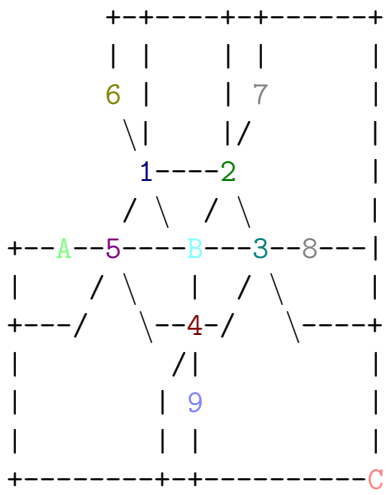


Now there is rampant twinning, and it's hard to see how the result even relates to the intended pattern. Without annealing, sometimes the pattern comes up perfectly (frequently on a diagonal) or close to perfect with recognizable twists, and sometimes it's a total mess.

Example 5

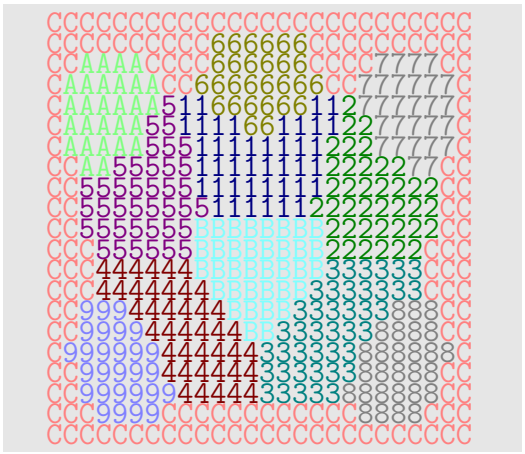
A complicated pattern but with fewer vulnerable cycles.

Graph:

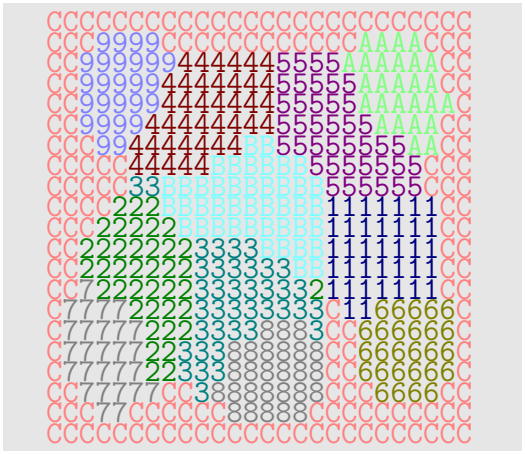


Fixed boundary conditions at C.

Representative steady state annealed from melt in two steps:
($T = 0.375 \rightarrow T = 0.25 \rightarrow T = 0.1875$)



Without annealing:



Notice the single twist defect.

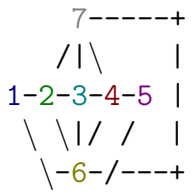
Several other common pathologies not related to convergence or annealing are worth mentioning:

- Boundary stiction: The previous examples all treated boundary conditions very carefully, either by identifying boundary-compatible states in the adjacency graph, by clamping the boundaries at the appropriate states, or by providing a special boundary-attached state which served as a sort of lubricating fluid bath for a free-floating pattern. If such steps are not taken, regions may become inappropriately "stuck" to the

boundary, resulting in missing neighbor contacts. Boundary contact, especially corner contact, also provides a strong foothold to regions, even under open boundary conditions; it can be unusually difficult to dislodge a region inappropriately stuck on an edge or corner, and high quorum sense gain is needed to prevent appropriately attached regions from becoming inappropriately large.

- Topological incompatibility: It is entirely possible to construct an adjacency graph that is incompatible with the patterning domain. A planar domain must be patterned with a planar graph, a toroidal domain with a toroidal planar graph, and so on. If the graph is incompatible with the domain, duplicated regions and missing neighbor contacts will typically result.

- Steric hindrance: Even if an adjacency graph is topologically compatible with its domain, given a set of quorum sizes and boundary conditions, the pattern may require so much distortion of the regions from their equilibrium shapes and sizes that some neighbor contacts will never form or some regions will be duplicated. For example, the graph



with boundary conditions fixed at 6 and uniform quorum sizes, will, in some orientations, produce a pattern that is too "fat" around the middle to bend and fit within its domain. In that case, 3 or 7 may be duplicated, may be missing a contact with 2 or 4, and/or may show patches of poor symmetry breaking. (3 and 7 are also examples of completely identical, mutually contacting states, subject to the partial melting effect, which exacerbates poor symmetry breaking.) In general, the solution is to prescribe a pattern that is not only topologically but also metrically compatible with its domain, and, if necessary, to impose boundary conditions that prevent a pattern from inappropriately wedging itself into the domain and curling up. Careful annealing may

also help but is sometimes insufficient to ensure an orientation that avoids steric hindrance.

4.7 The Continuum Limit

In order to help better understand and predict the melting point and correlation length phenomena, we can construct a continuum limit version of the mean field theory and study it analytically in special cases. In the continuum limit, the only important difference from the mean field formulation is that there are no longer discrete cells. Instead, the p_i become fields over some continuous spatial domain, one probability field for each state. Quorum sensing can be generalized straightforwardly, using the screened Poisson equation. The only term seriously affected by this change is the neighbor adhesion energy.

How, in this version, should we formulate adhesion energy, given that we no longer have any clear idea what a neighbor is? Since the adhesion term in the mean field theory is linear over neighbor state probabilities, we can re-interpret it as the expected adhesion energy for the "average" neighbor. In the continuum, we can attempt to measure the average state over some closed surface surrounding each point and interpret that as the average neighbor state. A simple two-term Taylor expansion for the average over a sphere of radius l in D dimensions is $(1 + \frac{l^2}{2D}\nabla^2)f$. Indeed, when discretized to lowest order on a square mesh, this turns into precisely the average neighbor calculation used by the mean field theory. The length scale l corresponds to the size of a cell in the mean field algorithm.

With this construction, we can write the steady state equations for the continuum limit:

$$p_i = e^{-E_i/T} / \sum_{j \in states} e^{-E_j/T} \tag{4.3}$$

$$E_i = -\frac{k_q}{q_i + k_{soft}} - k_h p_i - \sum_{j \in states} ((p_j + \frac{l^2}{2D}\nabla^2 p_j) U(i,j)) \tag{4.4}$$

$$\nabla^2 q_i = -p_i^2 + \gamma_q q_i \tag{4.5}$$

where i ranges over states.

Unfortunately, this system of PDEs is quite hairy, with two equations for every state and fully nonlinear. In the general case, it may be difficult if not

impossible to study. In the special case of a two-state system, however, we can manipulate it into a much more tractable form.

Noting that $\sum_{i \in states}(p_i) = 1$, we can model a two state system with only one independent probability field, p, letting $p = p_1$ and $1 - p = p_2$. Carrying the exponential term into the denominator, we can write

$$\begin{aligned} p &= 1/(1 + e^{(E_1 - E_2)/T}) \\ \implies E_1 - E_2 &= T \ln(1/p - 1) \end{aligned} \tag{4.6}$$

We can expand $E_1 - E_2$ into

$$\begin{aligned} E_1 - E_2 = & k_h(1 - 2p) + \\ & k_q(1/(q_2 + k_{soft}) - 1/(q_1 + k_{soft})) + \\ & (p + \frac{l^2}{2D}\nabla^2 p)(2U(1,2) - U(1,1) - U(2,2)) + \\ & (U(2,2) - U(1,2)) \end{aligned} \tag{4.7}$$

This can be solved explicitly for the derivative terms, yielding an elliptic equation that is no longer fully nonlinear but rather semilinear, easily amenable to linear stability analysis.

4.8 Analytical Properties from the Continuum Limit

The form of the continuum limit probability equation is very much like an inhomogeneous Helmholz eqution or screened Poisson equation, with a nonlinear logarithmic term added. The character of the solutions, either oscillatory (Helmholz) or exponential (screened Poisson), depends both on choice of parameters and on the value of p, acting through the nonlinear term. This dichotomy provides an analytical explanation for the melting point phenomenon, for the near scale-invariance of steady state patterns, and for other details of behavior.

If we further simplify things by ignoring quorum feedback ($k_q = 0$), a reasonable assumption for symmetric or nearly symmetric solutions, we're left with a single PDE:

$$\frac{l^2}{2D}\nabla^2 p = -\frac{T}{U_\Delta}\ln(1/p-1) + \frac{k_h}{U_\Delta}(1-2p) - p + (U(2,2) - U(1,2))/U_\Delta \tag{4.8}$$

with

$$U_\Delta = U(1,1) + U(2,2) - 2U(1,2)$$

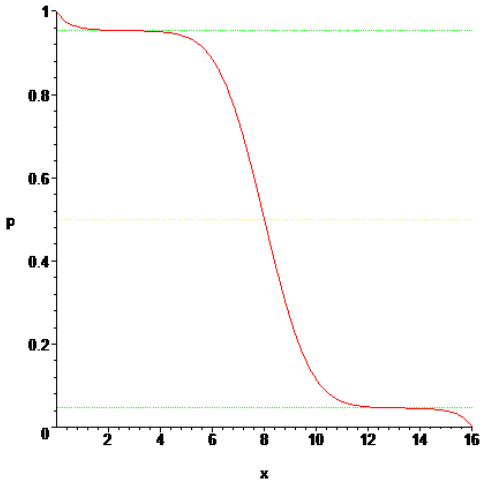
which is typically positive as a result of surface tension terms. If U is constructed as prescribed above, with states 1 and 2 symmetric, $U_\Delta = 2k_s = 2(U(2,2) - U(1,2))$. In that case, the right hand side becomes

$$-\frac{T}{2k_s}\ln(1/p-1) + \frac{k_h}{2k_s}(1-2p) - p + 1/2 \tag{4.9}$$

In 1D, or for problems with planar symmetry, $\nabla^2 p = d^2p/dx^2$, and this formula is just an ODE. It is well-behaved as a boundary value problem, albeit a little twitchy to solve numerically, given the explosive singularities at $p = 0$ and $p = 1$.

For low temperatures, p'' has three stationary points: two unstable points corresponding to saturation in one or the other state, and one stable point corresponding to a balanced mix ($p = 1/2$, assuming the states are symmetric) – see figure below. An ideal domain wall between regions corresponds to a heteroclinic orbit from one unstable point to the other. Such an orbit can spend arbitrarily long escaping from one unstable point and arbitrarily long approaching at the other, but the space devoted to crossing the middle is fairly consistent. This captures, in a nutshell, why steady state patterns are approximately scale-invariant, and yet why there is a definite correlation length and domain wall thickness.

Figure: Plot showing the three stationary trajectories at low temperature and an example boundary value trajectory going from $(1-\epsilon)$ to ϵ. ($T = 0.1875$, $k_s = 0.5$, $k_h = 0.125$)



We can derive an analytical estimate for the domain wall thickness W from the wavelength of low-amplitude oscillations about the stable point. Noting that $\lambda = 2\pi/\sqrt{-\frac{d}{dp}p''}$, we have

$$\frac{l^2}{2D}\frac{d}{dp}p'' = \frac{T}{2k_s}\frac{1}{p-p^2} - \frac{k_h}{k_s} - 1$$

and hence, defining W as the half-wavelength,

$$W = \pi l/\sqrt{2D(-2T/k_s + k_h/k_s + 1)} \tag{4.10}$$

In the case corresponding to the discrete examples above ($T = 0.1875$, $k_s = 0.5$, $k_h = 0.125$, $l = 1$, $D = 2$), this comes out as $W \approx 2.2$.

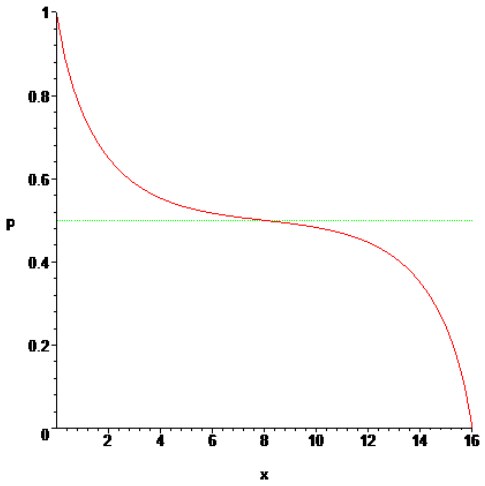
The smallest possible value of W is $\pi l/\sqrt{2D}$, fairly close to l – a domain wall can be no narrower than a cell. As temperature increases, W increases, first slowly, then rapidly, until T reaches

$$T_m = (k_h + k_s)/2 \tag{4.11}$$

where W diverges and the three stationary points collapse into one unstable stationary point. This T_m is the melting point. In the example above, $T_m \approx 0.31$. Above the melting point, instead of lingering indefinitely in saturation, orbits rapidly collapse to a symmetric mixture, linger there indefinitely, and

only depart for the opposite boundary condition at the last minute (see figure below). Just below the melting point, the saturated states remain distinct but are only minimally differentiated from each other and from a symmetric mixture.

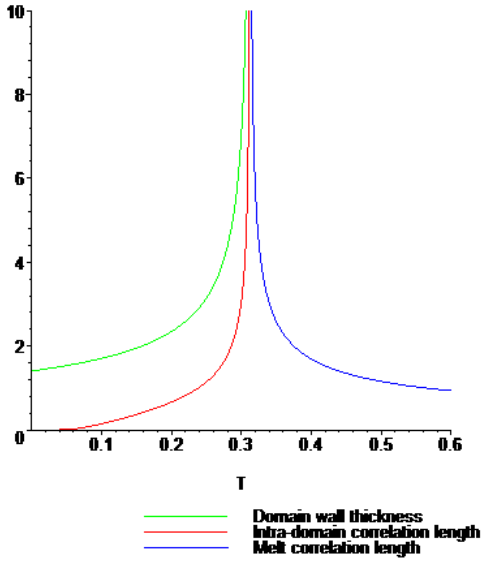
Figure: Plot showing the lone stationary trajectory at high temperature and an example boundary value trajectory going from $(1 - \epsilon)$ to ϵ. ($T = 0.375$, $k_s = 0.5$, $k_h = 0.125$)



Within a domain – below the melting point – probabilities decay exponentially toward the associated (unstable) stationary point. The characteristic $1/e$ length scale of this decay can be thought of as an intra-domain correlation length, indicating how quickly the interior of a domain forgets about the domain's neighbors. The intra-domain correlation length is not conveniently analytically solvable (the associated p values can only be determined implicitly), but it can be seen to approach zero as T approaches zero and to diverge similarly to W as T approaches T_m (illustrated in the figure below).

Beyond the melting point, a similar correlation length can be defined, about the lone unstable point. This "melt correlation length" describes how far information about boundary conditions penetrates. It can be seen to diverge at the melting point and then fall away toward zero as temperature rises further (see figure).

Figure: Domain wall thickness and correlation length versus temperature. ($k_s = 0.5$, $k_h = 0.125$)



Thus far, we have explained the emergence of discrete domains separated by domain walls of a characteristic thickness and a melting point at which the domains disappear, in terms of the state correlations propagated by cells to their neighbors.

What about partial melting? Some insight into the phenomenon of partial melting can be gained by now analyzing the boundary-independent case: patterning under either periodic boundary conditions or boundaries clamped to a symmetric mixed state (i.e. $p = 0.5$). This yields a symmetry-breaking problem, and by analyzing a time-dependent version of the continuum theory, we can determine what circumstances will cause symmetry to spontaneously break.

A simple time-dependent formulation (omitting quorum sense) is

$$E_i = -k_h p_i - \sum_{j \in states} ((p_j + \frac{l^2}{2D} \nabla^2 p_j) U(i,j)) \tag{4.12}$$

$$P_i = e^{-E_i/T} / \sum_{j \in states} e^{-E_j/T} \tag{4.13}$$

$$dp_i/dt = P_i - p_i \tag{4.14}$$

For the symmetric 2-state case, we can again reduce to a single p variable and consider solutions of the form

$$p = 1/2 + \epsilon e^{ik\cdot x + \rho t} + O(\epsilon^2)$$

Solving to first order yields

$$\rho = \frac{1}{2T}(k_h + k_s(1 - \frac{l^2}{2D}|k|^2)) - 1 \tag{4.15}$$

When $\rho > 0$, perturbations grow exponentially and symmetry spontaneously breaks. For uniform perturbations, $k = 0$, this condition is identical to $T < T_m$. However, if perturbations are constrained to be symmetric (e.g. by quorum sense), then the temperature at which symmetry will break depends on how much space is available. The wavenumber of the longest wave that will fit determines the reduced temperature of partial melting:

$$T_{m|k} = (k_h + k_s(1 - \frac{l^2}{2D}|k|^2))/2 \tag{4.16}$$

This explains why partial melting affects interior symmetry breaks but not domains constrained by boundary conditions. It does not, however, explain the surprisingly low partial melting temperatures seen in competition among identical, mutually compatible states. For this, we must take into account more than two states.

To keep the algebra tractable, we consider a special, restricted N-state problem. Two states are modeled in full, while the other $N - 2$ states are assumed to act completely independently, such that their particular values have no effect on the probabilities associated with the two distinguished states. It is assumed that the two distinguished states always share a total probability of R and the remaining $N - 2$ states share $1 - R$ (their energy functions constrained to keep this steady). Under these assumptions, the two distinguished states can be represented with p and $R - p$, and we can perform a similar (albeit messier) analysis, yielding

$$\begin{aligned} T_{m|k,R} &= (k_h + k_s(1 - \frac{l^2}{2D}|k|^2))R/2 \\ &= T_{m|k}R \end{aligned} \tag{4.17}$$

This surprisingly simple result shows that, when more than two states compete during symmetry breaking, the partial melting temperature is depressed by a factor of R. When the states are completely identical, $R = 2/N$. This accurately predicts the dramatically low partial melting temperatures seen when many identical, mutually compatible states are set in competition independent of any boundary constraints. However, the appearance of hysteresis, frequently seen when $N > 2$, remains unexplained.

4.9 Abstracting the Strategy

So far, I have described our normal neighbors patterning method in concrete terms, based on specific mathematical operations. Can the principles be generalized beyond this particular system? Can they be applied in synthetic biology or even in the study of natural organisms? In this section, I demonstrate how the normal neighbors strategy can be implemented without the details of the Boltzmann distribution, without the requirement of cell-cell contact, and without the rigid state encoding we have assumed. The limits have yet to be reached.

Our patterning strategy can be re-expressed as the combination of five core mechanisms:

- State representation: A means of locally identifying the possible roles an element may take in the pattern. We have already seen two different representations – a discrete, n-ary value and a vector of likelihood weights. More abstract representations are possible, for example, by partitioning some large state space into basins of attraction. Given an existing state representation and a new, "desired" one, there must be some means to update the existing representation to make it more like the desired one – e.g. by relaxing or randomly updating. For continuous state representations, this leads to one important restriction, which will be discussed below.

- State sharing: A means of propagating state information to nearby points. Thus far, we have assumed this meant sharing the local state representation with a cell's nearest neighbors, but the information could be encoded differently, perhaps imperfectly, and the range over which it is shared, though it needs to be locally biased, need not have an

absolute cut-off. Decay-diffusion based mechanisms, for example, are a viable alternative.

- Quorum sense: A means of detecting whether a given state occupies a globally appropriate fraction of the domain. Not all states will require quorum sensing; some may already be adequately constrained by boundary conditions or neighboring states. For regions with some operating function (e.g. an organ), quorum sense may simply be based on a measure of whether the function is keeping up with the organism's needs. In such circumstances, quorum sizes will vary depending on how much demand there is for their different functions. Quorum sensing must generally converge quickly compared to state update times, to prevent oscillations.

- "Suitability" function: A metric indicating how favorable a given state is given the prior state, the neighborhood, quorum feedback, and other environmental cues, abstracting the idea of an energy function.

- Discriminator function: A mechanism for picking out the most favorable state(s) according to their relative suitability metrics. The Boltzmann distribution provides the prototypical example, but the essential function is that of a differential amplifier exhibiting saturation (and its generalization to more than two inputs). The different suitability metrics are compared, and the largest one is selected if a clear winner is present. If several are similarly suitable, they are all partially selected, with some gain applied to their differences. The "temperature" parameter describes the sharpness of discrimination – the differential gain – determining how close is close enough to be ruled an approximate tie and how much the remaining differences are amplified. Such residual variation facilitates symmetry breaking and annealing.

As an example of how these mechanisms can be realized differently, consider how normal neighbors patterning might be implemented in a reaction-diffusion system, as a prototype for how it might be realized biochemically in a synthetic bacterial colony. Quorum sense and energy functions can be implemented as before (aside from the requirement that quorum sense morphogens have high diffusivity compared to state sharing morphogens, or are otherwise faster to converge than state updates) but state sharing, the discriminator function, and possibly state representation must be revisited.

State sharing presents several problems. There is no corresponding notion of a cellular neighborhood with which to share state, and the Taylor interpolation trick is not clearly feasible. Instead, we can use a family of decaying, diffusible morphogens to share state, defining the neighborhood to be their response kernel. The source density and attenuation rates may not be uniform, however, e.g. as cells proliferate. The size and shape of a neighborhood can tolerate some variation, but the energy functions expect to be fed a consistent state representation, without wild variations in magnitude. Rather than directly feeding in morphogen levels, a simple solution is to add a normalization step – an automatic gain control – on the receiving end. Automatic gain control can be realized through competitive binding or negative feedback motifs.

Because exponentials are a rarity in chemical kinetics, the Boltzmann distribution is probably impractical as a discriminator function. The tremendous virtue of the Boltzmann distribution is its unlimited dynamic range – it is completely insensitive to common mode inputs. During symmetry breaking, as the environment goes from completely indeterminate to fully determined, the energy functions can easily vary by an order of magnitude. Thus, wide dynamic range is an important feature. The Boltzmann distribution also provides the option to configure extremely high gains, by reducing the temperature parameter. A replacement must, at a minimum, be able to produce sufficiently high gains in order to chill beneath the melting point.

Lacking a convenient design for a biochemical differential amplifier, we can construct an adequate discriminator function by cascading three stages: compressing the dynamic range, applying gain, and normalizing the results. The first and last stages can both be implemented with automatic gain control circuits, while the gain stage can be implemented with a cooperative Hill function. Typical cooperative binding interactions (e.g. tetramerization) cannot provide sufficiently high gains, so we increase the gain by applying positive feedback. This can be internal to the discriminator or embedded in the energy functions via the hysteresis term. Given that cooperativity is not usually adjustable, the feedback term can also be used to adjust the effective temperature. The result is not nearly as clean as the Boltzmann distribution, and it is probably still too complicated for a practical biochemical implementation, but it works.

Given that transcription factors and signal transduction pathways are scarce resources in a cell, it may also be desirable to revisit state representation and the representation used for state sharing. The vector of weights used

previously is an extremely verbose encoding, and much denser encodings are possible, using, for example, random sparse combinatorial encoding. However, an important restriction must be observed: if two states have an edge in the adjacency graph, their intermediate mixtures must be uniquely representable. That is, their basins of attraction in representation space must contact each other, and the interpolation path connecting the two states must cross directly from one basin to the other, confusable nowhere with any other states. We may term such an encoding "conflict-free". As an example, if states are placed on the vertices of a hypercube in R^n and linear interpolation is used for adjusting states, then the (possibly diagonal) line connecting two vertices representing adjacent states in the adjacency graph must never venture closer to any other state's vertex than its endpoints. This means the smallest sub-hypercube containing both endpoints must contain no other states. If the adjacency graph is a subgraph of a hypercube, then all 2^n vertices can be used to represent states. If the adjacency graph is completely arbitrary, however, it may be necessary to use significantly fewer than $2^{n/2}$ vertices in order to avoid conflicts.

The reason for the conflict-freedom restriction is that, in general, a domain wall cannot form if its intermediate states are not representable or cannot identify which domain wall they are part of. If the intermediate states are confusable with another state, that state will typically insert itself instead. The only observed exception is when the temperature is sufficiently low that domain walls are significantly thinner than a single cell, in which case intermediate states are not needed. However, such temperatures are, by necessity, far too low for effective annealing.

4.10 Comparing with Other Patterning Mechanisms

Spatial patterning strategies can be roughly classified by the properties of their method of computation. Three obvious categories stand out, which could be termed as follows: feed-forward algorithms, feed-back algorithms, and imperative algorithms.

Feed-forward algorithms, e.g. [45] (excluding a few features), [18], employ positional information encoded in boundary conditions, planar polarizations, or prepatterns, and compute a cascade of spatial functions based on this in-

put. In the steady state limit, none of these functions have any memory, and as there are no cycles in the computation, their composition is memoryless as well. Such algorithms cannot break symmetry, lacking memory and a global cone of influence. However, they are often scale-invariant by virtue of deriving all scale measures from boundary information, and may be approximately invariant under much more general transformations (e.g. the restricted class based on diffusion under Dirichlet boundary conditions is completely invariant under arbitrary conformal transformations). They show some limited self-repair due to their memorylessness but often respond poorly to damage to their boundary information.

Feed-back strategies, such as Turing patterns and their many relatives [62, 50, 42, 24], on the other hand, typically generate their own positional information through instabilities, or bootstrap it from weakly asymmetric initial conditions. They typically have baked-in length scales associated with their instabilities and show very limited transformation invariance, apart from some response to boundary shape and a little scale flexibility when trapping a partial wavelength. However, they exhibit extremely robust self-repair, recovering even from the loss of boundary information. Historically, there seems to have been little work on fully programmable feed-back algorithms, but it's not clear whether this reflects any fundamental difficulties.

The normal neighbors algorithm presented here is a feed-back algorithm, but it shows a mixture of properties typically seen among feed-back and feed-forward algorithms. It is strongly symmetry-breaking, yet it shows approximate scale invariance (at least, if annealing is allowed) and transformation invariance (within the limits of surface tension and quorum sensing). It also shows excellent self-repair, even the ability to re-construct missing boundary information.

Locally checkable patterning [71] is another unusual, programmable feed-back strategy, with some important similarities, although it is tightly wedded to a discrete, static world. Normal neighbors patterning is, in a sense, locally checkable patterning adapted to an isotropic continuum without angular discrimination (albeit using completely different algorithms). However, as I argued, non-local constraints such as quorum sensing are inevitable if scale invariance or boundary independence are goals. Yamins does demonstrate some discrete "scaffolding" constructions for providing approximate scale invariance via geometric constructions (e.g. bisection), but they appear extremely fragile, in addition to relying crucially on anisotropic angular discrimination.

A third category of algorithms might be termed "imperative". In a sense, this is just the generalization of feed-back algorithms to include discrete, mutable state, but such algorithms often show a distinct character, with strict spatial and temporal sequencing and hand-offs of control, e.g. [15, 65]. Although the most general category computationally, typical examples of such algorithms rely on external positional boundary information and do not show any automatic self-repair without significant design effort. They often show an interesting feature, however: the ability to signal their completion. Some feed-forward algorithms can signal completion given static, known domains (e.g. [45]) but not on dynamically changing domains. As far as I know, completion detection for feed-back algorithms, including the algorithms presented here, is largely unsolved. The absence of outright adjacency errors can be detected, but it is less obvious how to distinguish a stationary state from slow domain wall migration.

Of course, it is not necessary to use these different techniques in isolation. A complex patterning process might use all three. For example, normal neighbors-type feedback patterning might be used to establish initial coordinates and a layout of compartments, providing the necessary boundary information for feed-forward patterning to fill in details. Sub-compartments thus defined might then be textured with additional Turing-type feed-back patterning. Cross-cutting networks like nerves might be traced out by a growing point-type imperative mechanism but then maintained by a high-hysteresis normal neighbors controller, allowing for the correction of simple defects.

Feedback patterning such as normal neighbors patterning is probably better suited to highly curved surfaces than typical feedforward and imperative systems that reckon based on distance and direction to landmarks, due to the effect of physical bottlenecks in the geometry, e.g. at a budding structure. With only a narrow connection to the base substrate, one needs need to re-amplify the asymmetry cues, or even break symmetry all over again, unless out-of-surface landmarks are available.

Additionally, normal neighbors patterning may be better suited to actively deforming surfaces (e.g. as in Chapter 5) because distances and angles may change wildly, but in-surface adjacency rarely changes. Quorum sensing will be an issue if area changes, however, as will surface tension if boundaries become eccentric or convoluted. Further refinement of the algorithm may still be worthwhile.

When large-scale surface deformation is directed by the patterning al-

gorithm itself, feed-forward control per se is probably impossible. Because changes to the substrate feed back into the behavior of the spatial communication mechanisms used by the patterning algorithm, the algorithm is no longer properly feed-forward and can develop hysteresis and instability. I have not identified under what circumstances this is manageable. Normal neighbors patterning, however, behaves fairly stably.

The basic normal neighbors phenomenon, the consistency of adjacency relationships, is not the exclusive province of patterning algorithms that treat such normal neighbor relationships as first-class entities. A similar property can be found in several other patterning processes, albeit without the full regenerative, bootstrapping, and scale invariance powers seen in biology and in our normal neighbors algorithm. For example, any pattern computed as a function of a non-redundant coordinate system will preserve normal neighbors when that coordinate system is disrupted by a continuous perturbation. (The same is not true of redundant coordinate systems, which can be perturbed off of the constraint surface used by the normal pattern.) Also, coupled dynamical systems may be found to converge into typical normal neighbor relationships, in space and in time [24].

Physically, the discrete normal neighbors algorithm has a lot in common with Winfree tiling [69] and the kinetic tile assembly model [70]. It might even be possible to implement a physical variant of the stochastic algorithm with DNA tiles, using limited tile concentrations in a small fluid volume to provide quorum feedback. (I have not investigated whether this could be practical.) Winfree et. al. have pursued a very different direction with their mechanism, however, using a one-pass imperative patterning style to produce precisely-defined, fixed-scale patterns.

The stochastic algorithm is also closely related to the classic Ising cellular model of ferromagnitism (of which it is a generalization), and similarly, the mean-field algorithm overlaps much with the Ising model's mean-field approximation. The melting point phenomenon seen in the normal neighbors model is mathematically equivalent to the 2nd-order phase transition representing the Curie point in the Ising model, complete with critical fluctuations. Other generalizations of the Ising model, such as the cellular Potts model [29], share important similarities, although typically designed for different purposes. Their potential as a model for spatial signal regeneration in patterning seems to have gone unnoticed.

Algorithmically, the stochastic algorithm is closely related to the optimization technique of simulated annealing, while the mean field algorithm

has close parallels with loopy belief propagation [72].

4.11 Future Directions

A variety of interesting open problems remain in relation to normal neighbors patterning. The chief such problem I have been investigating is the application of normal neighbors patterning to controlling active surface deformation (Chapter 5). However, much more remains to be understood about normal neighbors patterning itself.

Several natural extensions to the model remain to be explored. Anisotropic patterning using substrate polarization may prove useful and biologically insightful, probably in association with angle discrimination in neighbor-neighbor relationships and pattern-driven control of polarization. Temperature gradients may also yield interesting effects and may be useful in modeling gradients of developmental maturity, as seen in some embryos.

Pattern composition is an important space that remains to be investigated. One might wish to hierarchically nest normal neighbors patterns, so that coarse structure is laid out before fine structure and opportunities for topological defects are minimized. Another strategy for minimizing the complexity of patterning stages might be to layer multiple planes of patterning, combinatorially encoding the regions of interest. Additionally, patterns of indefinite depth with repeated or nested elements are not obviously possible with single passes of normal neighbors patterning, but they may be possible with iteration and recursion. It is not immediately clear how to provide such mechanisms with finite state and communication resources. In all such cases, context free nesting and composition may sometimes be appropriate, but other times some communication between the pattern layers will be needed. Bidirectional constraints may re-introduce the problems of undesired, topologically unresolvable local minima (e.g. twists); unidirectional constraints from coarse to fine may prove more robust.

A formal understanding of the properties of reliable and unreliable patterns would be valuable in order to combat defects. Other defect-control strategies might include self-paced annealing and weakly pre-breaking symmetries. Such strategies may even prove useful in elucidating the robustness of natural developmental patterns.

The combination of quorum sense and surface tension provides a stable geometry from an otherwise purely topological adjacency graph. This raises

the question, what other mechanisms might there be that can provide stable geometry? Not all patterns embodying normal neighbor relationships are necessarily characterized by scale invariant, mostly convex blobs. In Turing patterns, finite wavelengths help provide stable geometries; can this be incorporated? It is also possible to produce interesting stabilization effects at high hysteresis, where regions no longer need be blob-like and do not spontaneously form and anneal; instead, they must be initially traced out, for example, by a growing points mechanism; subsequent corrections will be highly local.

Finally, and perhaps most importantly, does all this have anything to do with biology? I have shown how the mechanisms here can be generalized to much more biologically realistic encodings and functional operations. (An important detail not yet addressed is the question of what happens when different state components have different neighborhood ranges, likely the case in biology.) Biological cell fates are surely not arranged so formally, but do the same principles apply? Can we identify, for example, the notions of temperature, melting point, correlation length, surface tension, geometric stabilization, conflict-free encodings, and topological defects in developmental patterning and regeneration? Formulating experiments to probe for these phenomena may be the next great challenge.

Chapter 5

Morphological homeostasis

The physical forms of multicellular organisms are amazingly robust, developing correctly in spite of substantial environmental and genetic variation. This phenomenon was dubbed the "canalization" of development by Waddington [63], reflecting the notion that there seems to exist some sort of restoring force pulling developing organisms back to their expected phenotype whenever perturbed. The most dramatic example may even span entire phyla, as organisms within a single phylum start from dramatically different initial conditions yet converge to a common "phylotypic" stage of development, before differentiating into their characteristic larval forms [36]. Similar convergence effects in spite of environmental perturbations can also be seen to varying degrees in the adult forms of animals, ranging from wound healing, to limb regeneration, to complete body reassembly after disaggregation, as in the hydra [27].

Waddington's hypothetical "restoring force" cannot be completely hypothetical. For the dynamics of a physical system, such as an organism, to converge to a common attractor, the dynamics must be sensitive to the present state of the system – there must be feedback. In particular, the dynamics must be sensitive to the relevant variables that characterize the expected equilibrium state, or to strong predictors thereof, regardless of disturbances. Though such sensitivity can be a natural consequence of inanimate dynamics, for example, the surface tension that draws a droplet into a sphere, with the complexity of biological forms, it is strongly suggestive of explicit feedback control. We might dub Waddington's phenomenon, as extended to the adult, "morphological homeostasis".

The aim of this chapter is to explore this problem of morphological home-

ostasis from the perspective of forward (re-)engineering, focusing on the specific problem of self-stabilizing surface geometry. We can expect a self-correcting approach based on feedback control to be valuable for engineering and evolvability alike, since it helps to buffer adverse interacting effects of local changes (pleiotropy), in addition to responding to environmental insults. Attempted modifications are not necessarily themselves neutralized away; instead, selected control points become "orthogonal" in a sense, achieving their aims while leaving other vital properties unaffected. Morphological homeostasis thus embodies not only an elusive engineering vision but also a valuable design strategy for complex systems.

5.1 Decomposing the Problem

Natural biological structures are complicated, combining multiple subparts with differing characteristics. We can simplify the problem of engineering morphological homeostasis by breaking it into a cascade of two subproblems: "what goes where" and "what happens here". "What goes where" represents a pure patterning problem with no actuation, a body plan for the structure. "What happens here" is the problem of actuating within a region to produce some desired local result, given that the high level, global pattern is already specified. Of course, these problems are not independent – patterning and pattern updates affect the downstream actuation, while actuation changes the substrate and thereby changes both the pattern and the informational signals on which the patterning process depends. However, I show that, given suitably robust and self-stabilizing solutions to the two problems, able to withstand such cross-talk effects by treating them as perturbations, if the controllers' goals are compatible, their combination can yield a complete solution that retains stability.

The presence of conserved compartment maps in animals, an invisible and highly conserved pattern of gene expression prior to detailed morphogenesis [36], suggests that nature may use a similar decomposition strategy. Since perturbations in early, pre-morphogenesis development as well as local injuries to the final form can heal, global patterning and local actuation are both likely to involve feedback mechanisms.

The first problem in our factored approach to morphological homeostasis, "what goes where", can be solved by a patterning mechanism that is robust to fairly general substrate geometries including narrow bottlenecks and pro-

Parameter	*Description*	*Value*
T	Virtual temperature	0.2
k_s	Virtual surface tension	0.5
k_q	Quorum feedback	0.0125
k_h	Hysteresis feedback	0.125
γ_q	Quorum signal decay coefficient	10^{-4}
p_{min}	p threshold allowing actuation	0.5
(T_m)	Theoretical melting point (derived)	0.31

Table 5.1: Parameter values for the mean field thermal normal neighbors patterning algorithm used throughout this chapter. Parameters are constant except where specified.

duces meaningfully consistent patterns both before and after deformation. The patterning mechanism must also self-correct in the face of perturbations, without requiring a clean slate restart; incremental corrections to pattern and geometry must eventually converge, after all. These requirements all but eliminate self-timed pre-patterning and likely disfavor fixed-wavelength Turing-type mechanisms. However, the normal neighbors patterning mechanism of Chapter 4 fits almost perfectly.

Throughout this chapter, I use the mean field normal neighbors algorithm to construct and maintain a body plan pre-pattern. For simplicity, temperature and other parameters are left fixed (see Table 5.1; temperature annealing remains a refinement for future work). In order to keep simulation running time reasonable while allowing diffusion-based quorum sense to converge faster than cell state updates (necessary for stability), quorum sense morphogen concentrations are computed out-of-band by a physics module using successive over-relaxation, rather than by the cell agents themselves.

The core of this chapter, then, will be devoted to the problem of "what happens here": how to produce and maintain simple geometric features in spite of perturbations. We have at our disposal several actuation mechanisms, including cell shape change, apico-basal constriction, and neighbor traction forces (for simplicity, I don't consider changes in cell number here). We already know something about how to produce geometric features using these mechanisms, given a known initial state. However, given perturbations, the initial state is *not* known. Instead, we must find techniques that respond appropriately to the system's pre-existing state.

Sensitivity to the state of the system – feedback – requires either that the intrinsic physics of the system be sensitive to system state (e.g. me-

chanical restoring forces) or that explicit feedback sensors be deployed by the control algorithm. Geometric structure involves numerous degrees of freedom, and many of these degrees of freedom are uninteresting (e.g. the relative arrangement of equivalent cells) or undesirable (e.g. high-frequency Fourier components). It can be valuable to leave such degrees of freedom to autonomous energy-minimization dynamics, for example, viscous relaxation, avoiding the control algorithm having to treat them explicitly. On the other hand, certain degrees of freedom represent key control targets. For these, we require sensors.

5.2 Building Distributed Sensors

Within a region, the key control targets for morphological homeostasis tend to be distributed properties. Even simple properties like area, length, aspect ratio, and total curvature are measurable only in a distributed sense. When cells are largely indistinguishable, some of these distributed quantities are a challenge to measure, because they entail a subtle requirement for symmetry breaking.

To illustrate, consider the problem of quorum sensing. Suppose you wished to measure the total number of "blue" cells in a region. How would you count them? If cells were indistinguishable except for their colors and relative positions, and the topology of the surface were cylindrical but the exact size was unknown a priori, how would you avoid re-counting the same blue cells over and over again? The obvious answer is to somehow break the symmetry and elect a leader cell. By a variety of means, the leader cell can then facilitate a count (e.g. by establishing a spanning tree and performing a tree sum). Without breaking symmetry, however, the problem seems impossible.

By contrast, the closely related *fractional* quorum sensing problem is easily solved without symmetry breaking. For example, blue cells each emit a diffusible morphogen at a defined rate. All cells, blue or not, degrade the morphogen at a slow rate proportional to the morphogen's local concentration. The equilibrium morphogen concentration is then approximately proportional to the fraction of blue cells. The approximation can be made as good as needed by suitable choice of decay rate (slower for larger regions – the characteristic exponential decay radius must be much larger than the region size). A variation on this solution can also be applied to the absolute

quorum sensing problem – only the leader decays the morphogen – but again, a leader is required.

This limitation can be summarized in a general principle: an extensive quantity over a network can be measured only relative to some reference, which can be no more selective than the finest broken symmetry.

In the case of an average, the entire region is used as the reference, requiring no symmetry breaking. In the case of a sum, a leader appears to be required as a reference. As an example of an interesting intermediate case, although total area of a region apparently cannot be measured without electing a leader, the ratio of area to perimeter (the 2D analogue of surface area to volume ratio, inverted) can be measured if cells are aware of whether or not they are on the region boundary. The boundary cells can act as a reference, relying only on the intrinsically broken radial symmetry. Similar discrete quantities, such as the maximum distance to an interior cell, can also be measured using this broken symmetry.

Note that the choice of units is a separate issue from the nature of the reference. If cells can measure their own dimensions (e.g. by virtue of internal references), one can equally easily compute average cell area (dimensions L^2), average cell perimeter (L), and average cell aspect ratio (1), all referenced against the region as a whole. Averages can also be arbitrarily weighted, e.g. by cell number or by area. For coarse body plan structure, dimensionless measures are often preferable, so that the structure can be produced equally well in different sizes.[1] For similar reasons, measures that are also independent of total cell number are preferable. Such measures can be termed "scale-invariant".

5.3 Sensing Curvature

As a simple, 1-parameter control target for geometry, let's try to control spherical curvature, to produce spherical caps of varying curvature radii (and hence varying subtended angle). First, we need to construct a scale-invariant measure of curvature.

Classical local measures of spherical curvature, such as Gaussian curvature (intrinsic) and mean curvature (extrinsic) are not scale-invariant but

[1]Fine, repeated features with a characteristic size, such as intestinal villi or epidermal scales – "textures" on top of the coarser structural elements – must not, of course, be blindly re-scaled.

instead are expressed in terms of curvature radii; they reflect how tightly curved the surface is locally but not how much curvature the surface as a whole encompasses. Gaussian curvature can be integrated over the entire region area to produce a dimensionless invariant related to the subtended angle (by the Gauss-Bonnet theorem), but this is an extensive quantity, requiring leader election to measure. It would be preferable to avoid introducing this complication unless absolutely necessary.

Another approach is to consider global properties based on length and area. For example, on a spherical cap, the ratio of area to the square of some linear dimension (e.g. perimeter or radius) uniquely identifies the angle subtended. Radius is measurable, though somewhat awkward. On the other hand, the ratio of area to perimeter is easily measured (as explained in the preceding section), providing an additional non-scale-invariant measure of curvature. This can be combined with an average of one of the local measures of curvature – for example, multiplying by the average mean curvature – to produce a scale-invariant measure of global curvature.

I have not rigorously studied all the possible combinations of measures here, but by trial and error I found an interesting variation that worked particularly well. Rather than combining the ratio of area to perimeter with another global measure, I combined it with a purely local measure of curvature, producing a hybrid metric that is partly local, partly global. This appears to be useful for controlling actuation, because the effects of actuation are also partly local, partly global. The particular measure I found most reliable was the product of the area-perimeter ratio and the extrinsic radius of curvature along the axis parallel to the region boundary – that is, the local circumferential curvature. The necessary axis is determined from level curves of a decaying gradient emitted by the boundary.

5.4 Actuating Curvature

Now that we have a sensor for curvature, how do we build an actuator? The answer is not obvious. As noted before, surfaces have numerous degrees of freedom; all of them need to be stable, and some of them need to reach particular control targets. In almost any representation, they are cross-coupled, due to the constraints of surface geometry and the complicated dynamics of deformation and flow.

For example, as a first attempt, one might instruct each cell to locally

bend in accordance with the sign of the error reported by the curvature sensor. Such "extrinsic" curvatures can be driven by mechanisms such as apical/basal constriction. This approach, however, suffers from two serious flaws: it both geometrically inconsistent, and it does nothing to keep undesirable degrees of freedom under control. It is inconsistent for the same reason one cannot flatten an orange peel without tearing it: extrinsic curvatures require, in general, non-Euclidean geometries within the surface. Distances between points within the surface must change in order to accommodate the extrinsic curvature. As the surface deforms extrinsically, non-Euclidean "intrinsic curvature" will necessarily be generated by elastic and plastic deformation, at the cost of high stresses, which in turn fight the original the extrinsic curvature and often lead to buckling instabilities, oscillations, and worse.

As a simple example, a sufficiently small circular disc subject to uniform extrinsic bending will produce a spherical cap, but beyond a certain critical size, it will spontaneously buckle cylindrically. The spherical conformation becomes unstable, and artificially stabilizing it requires high-gain, online control, analogous to supporting an inverted pendulum – not a practical morphogenesis strategy. Ideally, plastic deformation would set in before buckling, and the equilibrium intrinsic curvature would relax to allow the symmetric, spherical configuration without elastic instability. This is difficult to achieve, however, requiring surfaces that are plastically soft yet flexurally quite stiff, and regardless, the high stresses involved remain a liability.

The complementary strategy, actuating on intrinsic curvature, is similarly geometrically inconsistent but has some notable properties worth investigating. Unlike extrinsic curvature, which cells can very directly manipulate, the relationship between what a cell can do locally and the resulting effects on intrinsic curvature is quite nontrivial (given by the Brioschi formula), making the engineering design and control problems more complicated. Small changes to curvature can be produced by each cell changing its size and shape – adjusting its aspect ratio, for example. The effect on curvature is then a function of the differences in changes expressed by nearby cells. However, in order to avoid demanding that cells flatten into pancakes or stretch into spaghetti, large changes must be achieved by plastically rearranging cells rather than simply distorting them. A more useful actuator for large intrinsic curvatures is thus cell-cell traction, by which cells can intercalate with their neighbors.

How should cells exert traction forces in order to produce a given cur-

vature? This is, in general, quite complicated. For the particular case of axisymmetric curvature, however, as in a spherical cap, the "purse string" strategy is a viable option: if curvature is too small, cells near the edge should pull on their circumferential neighbors, so as to shrink the circumference of the mouth of the region. If curvature is too large, cells should pull on their radial neighbors, so as to enlarge the circumference.[2]

This sort of boundary-focused purse-string traction can be orchestrated, for example, by having the boundary emit a decaying gradient proportional in strength to the locally reported curvature error. The shape of the gradient then informs cells which direction and how hard to pull on their neighbors. The simplest approach might be to derive the orientation from the level curves or the gradient vector (choosing depending on the sign), and this works. I used an alternative source, the principal axes of the Hessian (negative axis along the boundary (due to sources), positive axis elsewhere), which seemed a little more effective in informal experiments.[3]

The effects of such purse-string traction are several. The application of traction forces leads to net stresses and bending moments in the surface, tending to open up or close the mouth of the region precisely as intended. In response, cells intercalate as expected, circumferentially or radially, leading to changes in intrinsic curvature. However, so long as curvature error persists, e.g. due to competing forces, the rearrangement is incessant. Reorienting after each rearrangement, cells continue to grapple on one another, rearranging repeatedly. This continuing churn nullifies the yield strength of the lattice and leads to viscous-like relaxation, a phenomenon which is both an asset and a liability. Such churn relaxation is helpful because, as alluded to previously, it provides a natural mechanism for uninteresting and undesired degrees of freedom to relax and stabilize, without explicit control. It is problematic because the desired target degrees of freedom relax as well, making it difficult to sustain more than small deformations.

Additionally, there is a subtle mathematical limitation to purse-string

[2]Interestingly, one cannot use a similar mechanism to produce hyperbolic curvature. Just as one cannot push with a string because it will buckle, so I have observed that pushing with a purse string mechanism causes the boundary of the region to buckle and foliate. Successful attempts at actuating hyperbolic curvature using traction have required cells near the center rather than cells near the perimeter to exert traction forces.

[3]Note that such actuation profiles are not scale-invariant, because of the fixed characteristic length scale of the gradient's decay. However, because the feedback sensors are scale-invariant, the resulting control algorithm is still quite flexible across a range of scales.

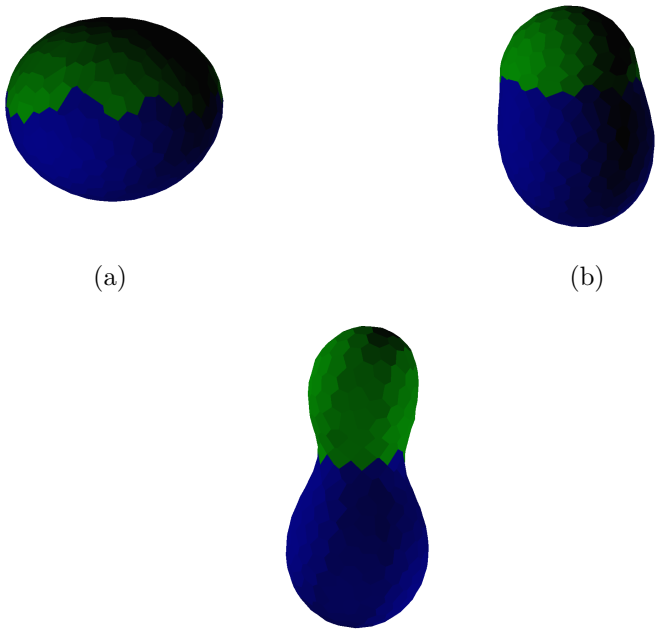
traction and all other intrinsically-based actuation methods: they become singular when the surface is flat. Starting from a flat conformation, purse-string traction is weak and has no way to influence which way the surface will buckle. The sign of its influence depends on the sign of the existing extrinsic curvature. If the sign of curvature is wrong, it cannot be corrected.

The complementary problems exhibited by extrinsic bending and purse-string traction suggest that their combination might be more successful than either in isolation. Indeed, merely running them simultaneously, without any coordination, produces a drastic improvement. The combination of purse string traction as described above and an integral controller on extrinsic bending, both using the same curvature feedback sensor, yields a stable and robust algorithm for producing spherical caps of arbitrary desired curvature. Figure 5.1 shows this tandem actuation mechanism in action, illustrating the results for several different target values of curvature.[4,5]

At first glance, one might expect that the two actuation mechanisms ought to be tightly correlated, so that consistent intrinsic and extrinsic curvatures would be correctly produced. However, this is not the case; the precise combination turns out to be quite forgiving. As the integral controller governing extrinsic bending ratchets up, intrinsic churn relaxation begins to lead towards rather than away from the desired equilibrium. At the same time, as cells rearrange, both autonomously and deliberately, the stresses generated by inconsistent curvatures are relaxed. Indeed, even without any coherent direction at all to the traction forces – a traction random walk – the combination of traction and extrinsic bending is sufficient. Convergence is slower and stresses are higher, but it works. In general, the relative calibration of intrinsic and extrinsic control affects the time to convergence and the stress profile, but the ultimate equilibrium is robust.

[4]The controller expects a complete surface, as in these examples. Boundary conditions matter; results, especially stability, can be quite different with, for example, open boundaries.

[5]Note that the example structures in this figure never quite reach steady state – they fluctuate between several similar conformations. The more complicated examples below eventually seem to stabilize; I am unsure of the reason for the difference.



(a) (b)

(c)

Figure 5.1: Lobes with controlled curvature – spherical surfaces divided into two regions (via normal neighbors), where green pursues a target curvature using purse-string traction and extrinsic bending, while blue relaxes passively (see Section 5.5). Three different target curvatures are illustrated, with ratios $1 : 3 : 5$ respectively.

5.5 Complex Structures from Simple Pieces

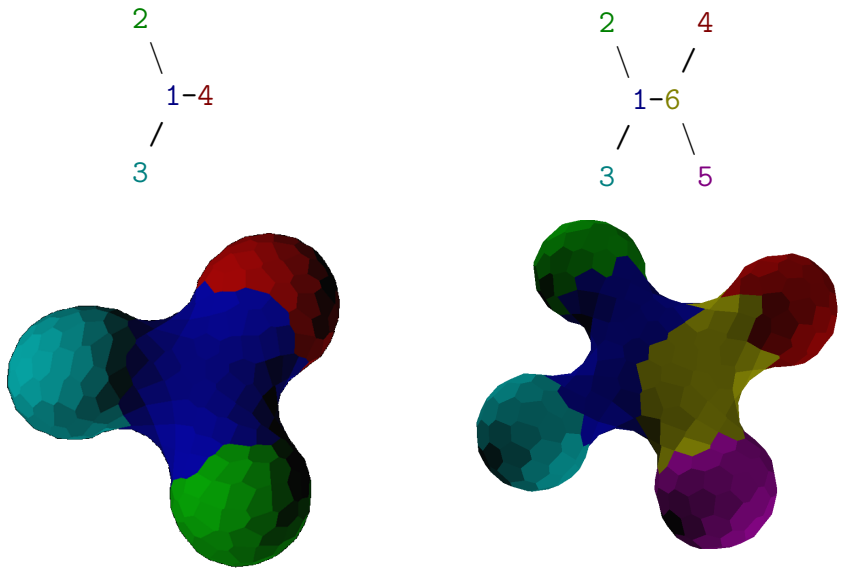
Now that we have the beginnings of an understanding of geometric control for simple, homogeneous regions, how might we proceed to more complicated structures? Rather than developing a slew of more complicated sensors, actuators, and controllers, each with multiple degrees of freedom, it would be simpler if we could instead combine multiple regions, each with a simple control law, to produce a more complex structure. With controllers like our simple spherical curvature controller above, however, simply cutting and pasting regions together does not work well. The controllers must behave compatibly along shared boundaries, or they will fight each other. Even if curvatures are carefully selected to be matched, evolvability is impaired,

because further changes require consistent modifications in multiple places simultaneously. Were we to employ controllers of heterogeneous character on neighboring regions, it might not even be possible to achieve a consistent match.

Instead of directly coupling tightly controlled components to each other, a better strategy might be to connect them through weakly controlled combiner (or "combinator") regions. Instead of tightly specifying all properties of the structure, one could specify only certain key regions and features, allowing combiner regions to interpolate between them for the remainder. Such combiner regions would insulate individual components from the geometrical and mechanical side effects of other components, allowing their controllers to operate quasi-independently.

Using the principle of relaxation, it turns out simple combiner regions of this sort can be implemented quite easily. For sufficiently small structures, I found that no controller is needed at all, just a simple routine to ensure cells are reset their default properties. Even though the substrate has a plastic yield limit that must be overcome, the churn injected from the jostling of neighboring regions is enough to cause mechanical relaxation, producing smooth connector regions with minimal curvature. For larger structures, I found it necessary to add a controller that deliberately relaxes the surface through cell-cell traction. A simple random walk of traction will often suffice. A more aggressive approach, less reliant on mechanical properties and randomization, is to use a smoothing geometric flow. A flow of this sort can be produced, for example, by exerting traction along the major axis of the Hessian of Gaussian curvature.

By definition, a weakly controlled relaxation combiner does little to dictate the relative positions of the regions it connects. Where, then, if not constrained by external forces, do they end up? The "what goes where" patterning mechanism may initially lay out the connected regions in some predictable fashion, but they effectively "float" within the combiner, and in the long run, they move to occupy positions that minimize mechanical energy. Typically, this process is dominated by the bending energy. Regions can be modeled, in a sense, as interacting by virtual forces, dependent on their curvatures. Regions of the same sign of curvature typically repel, while those of opposite sign attract. If the global conformation leads to the formation of a bend in the combiner region, subsidiary regions may interact with this as well. For example, when several regions of large curvature of the same sign float within a spherical combiner, they frequently align themselves along



(a) (b)

Figure 5.2: Simple compound structures and their associated adjacency graphs: (a) 3-lobe structure where red, green, and cyan regions control curvature while blue combiner region relaxes geometry. (b) 4-lobe structure where the lobes are split across two combiners (yellow and blue).

a circumferential ring, evenly spaced. These virtual forces can often be relied on to produce a particular, final conformation in space.

Figure 5.2 shows several examples of this approach, where independently controlled lobes are arranged by virtue of their interaction forces within a relaxation combiner. The number of lobes, their sizes and curvatures, and the divisions of the combiner can all be independently specified. However, there is no direct control available over the relative positions of the lobes. Even breaking the lobes into groups under different combiners does not meaningfully affect their positions; pure relaxation combiners are, to a good approximation, fully associative.

A more sophisticated combiner might try to apply explicit tractions and bending moments in order to customize the interaction forces among the subsidiary regions. More simply, however, we can break the associativity of the combiners with additional passive forces and use the resulting non-associative combiners to produce more complex shapes. An easy way to do

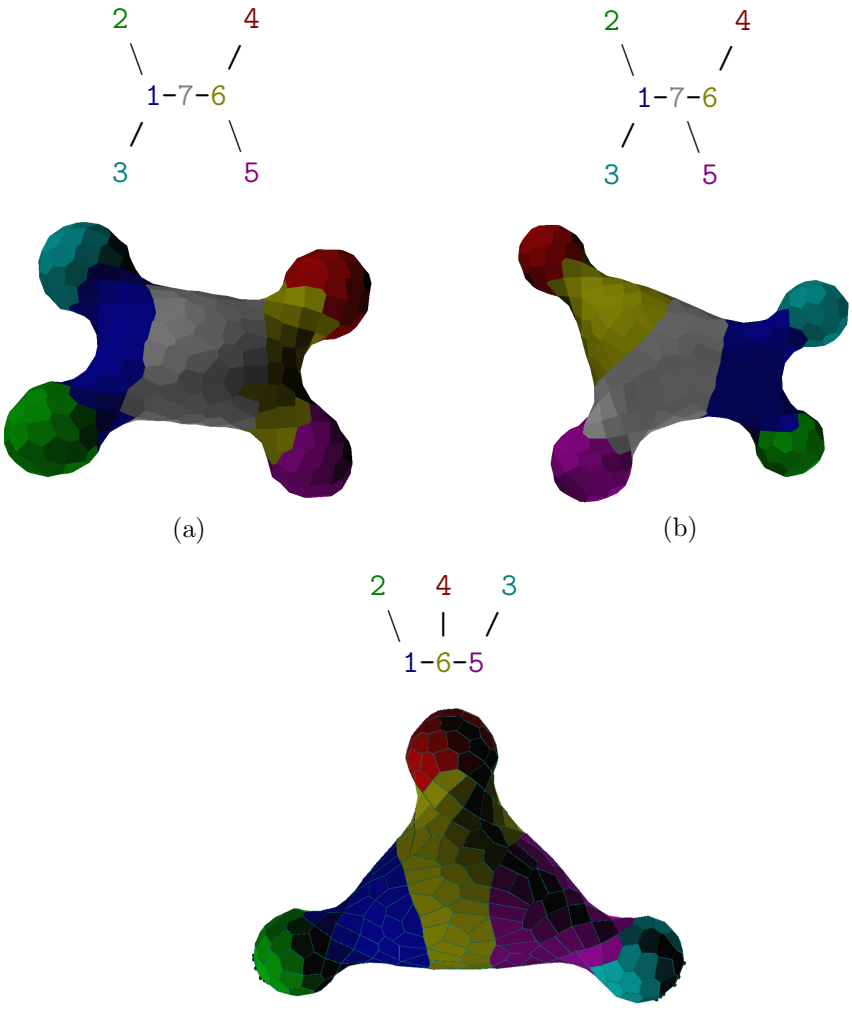


Figure 5.3: Compound structures using relaxation combiners whose associativity is broken by surface tension. Leaf nodes control curvature; non-leaf nodes are combiners. Combiner cells have adhesive self-affinity and mutual disaffinity such that internal edge tension is reduced and mutual edge tension increased by (a) 40% and (b), (c) 80%. (The stronger surface tension in the latter two helps produce more distinct conical features.) Additionally, the normal neighbors quorum feedback k_q is varied on a per-region basis to adjust relative areas – halved in leaf nodes, and, in (a), doubled in region 7.

this is with differential adhesion, such that different combiners have mutual disaffinity and hence are shaped by the surface tension forces along their boundaries. Figure 5.3 shows several examples of structures grown this way.

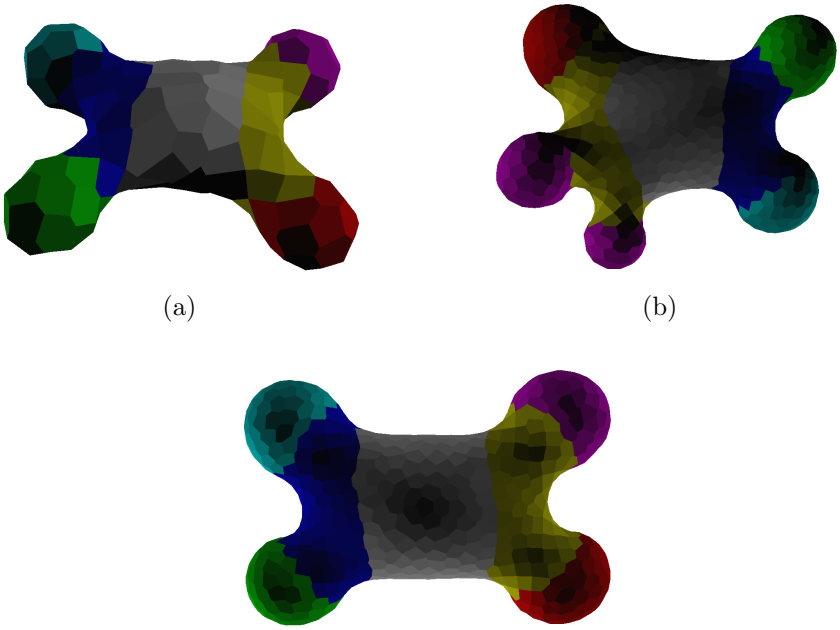
5.6 Evaluation

In spite of our meager toolbox consisting of one control law and two closely related combiners, the variety of structures we can declaratively produce is now beginning to get interesting. It remains to be shown that the structures exhibit the robustness properties I have claimed, including self-repair, approximate scale invariance, and tolerance of unexpected parameter variations.

Geometric self-repair follows easily from the feedback control mechanism. One can even take geometric results of running under one program, switch to a different program, and watch the structure reform. The results are essentially indistinguishable from structures produced starting from a sphere.

Approximate scale invariance can be demonstrated by running the same program on different size domains. Figure 5.4 demonstrates the program of Figure 5.3a running on different sized substrates. Using the same set of parameters as before), originally tuned for the middle size (400 cells), the small size (192 cells) works perfectly. The large size (900 cells) has a tendency to twin lobes but otherwise converges well (Figure 5.4b). In fact, while the small and middle sizes develop directly with little transient twinning, the large size develops extensive twinning with fully-actuated curvature, which only resolves through churn and domain wall migration. The highly curved lobe regions show a particular tendency to remain twinned, probably due to the influence of their mutual mechanical repulsion. Such twinning can be prevented by increasing the screening length of the quorum sense morphogens (i.e. decreasing their decay rate), as seen in Figure 5.4c, a change perfectly compatible with the smaller domain sizes, at the cost of longer convergence times. Alternatively, the normal neighbors temperature parameter may be increased (not necessarily compatible with small domains), which, even if it does not prevent twinning, assists in eventually resolving it.

The most interesting case to explore is that of unexpected parameter variation. For this purpose, I vary the stiffness of the substrate. This also affords the opportunity to explore the relative roles of the two actuation mechanisms in tandem. When substrates are stiffer, one should expect the



(a) (b)

(c)

Figure 5.4: Program of Figure 5.3a running on different domain sizes. (a) Small, 192-cell domain. (b) Large, 900-cell domain, showing typical twinning that fails to resolve. (c) Large, 900-cell domain that avoids twinning via 10x reduced quorum sense morphogen decay rate.

extrinsic actuation to be more powerful, while on softer substrates, intrinsic actuation should be stronger.

Table 5.2 summarizes the results of informal trials under several different values of bending stiffness constant k_B and with several different "knockout variants" of the curvature control algorithm. As claimed, only mechanisms that combine both extrinsic bending and traction able to succeed in all cases (and at all with the middle stiffness). A lack of *directed* traction is a hindrance, but only inasmuch as it reduces the speed of convergence. Interestingly, there are cases where each of the other mechanisms still succeed. With high stiffness, one hardly notices the total loss of traction. With low stiffness, some patterns develop successfully even without bending (although their precise shapes are visibly altered).

The tandem actuation mechanism thus exhibits *partial redundancy*: for

	$k_B = 0.8$	$k_B = 2.4$	$k_B = 8$
Tandem actuation	Default: Fail[a] Reduced limit stops: Ok	Ok	Ok
Bending only	Marginal (slow; complex patterns fail)[b]	Marginal (slow; high failure rate)[b]	Ok
Traction only	Marginal (complex patterns are slow or unsuccessful)[c]	Unsuccessful[c]	Unsuccessful[c]
Bending + random traction	Slow	Slow	Ok

[a]Fails by a lobe pinching off. I hypothesize this is due to excessively strong actuation collapsing the base of the lobe rather than allowing sufficient time for the main combiner body to slowly relax. Pinch-off can be prevented by putting tighter limit stops on actuation of *either* bending angle θ_{0u} or traction strength. The latter gives somewhat more consistent shapes.

[b]Fails through the development of tight, hyperbolic creases.

[c]Unsuccessful cases never produce definitive lobes; only slight curvatures form.

Table 5.2: Summary of results with varying substrate bending stiffness for default algorithm and several "knockout" variants.

many situations, multiple overlapping mechanisms are available, such that reduced function or complete failure of one pathway is quite survivable. However, due to the physical constraints of the problem, employing the full complement of mechanisms is often still helpful and sometimes absolutely necessary. The resulting combination mechanism is quite robust but irregularly so, giving confusing and seemingly contradictory results to knock-out experiments: Is the bending pathway necessary for curvature development? Is the traction pathway necessary for curvature development? Is the gradient field that feeds into traction necessary for curvature development? Differing conditions may produce differing "answers" to these questions. The situation is surprisingly reminiscent of the difficulties encountered in knockout experiments on real, live organisms [40].

5.7 Future Work

This chapter, I hope, has shed some fresh insight on the problems of developmental canalization and regeneration and how they can be achieved. It has, however, barely scratched the surface of what is possible, even without

exploring other mechanical substrates.

There are, of course, many more geometric feature controllers to develop. Of particular interest are ones that accept multiple neighbor connections, such as cylinders, as well as combiners that allow a greater degree of control in shape and placement. An interesting path to explore might be to use functions on gradients emitted by the subsidiary regions (or their boundaries) to construct a virtual "stress tensor" that is then actuated through neighbor traction.

The pinch-off pathology, briefly mentioned in Table 5.2, represents a larger problem that has only been crudely addressed: physical substrates have limits, beyond which they fail. Actuation mechanisms for deformation must be careful not to exceed these limits, or they will destroy their own substrates. The solution used here, enforcing fixed limits on actuator outputs, is crude because it is both hand-tuned and because it may unnecessarily limit outputs (and hence speed and control authority) even in situations where there is no imminent danger of damage. A more elegant mechanism might be for the substrate to recognize its own limits and express "pain" when over-exerted, causing actuation to back off.

A significant limitation with the current approach is that all patterning happens simultaneously in a single stage, which is both biologically unrealistic and limits the amount of complexity that can be implemented without getting stuck in local minima. Hierarchical and cascaded patterning would alleviate this limitation, but how can such sequential mechanisms be reconciled with regeneration? The answer is not clear, but perhaps a backtracking process is involved.

Chapter 6

Conclusion

The explorations in this thesis do not reveal any magic bullet – a simple, universal mechanism to explain and control development. If anything, they cast doubt on whether such a mechanism could exist. In my experiments, no one mechanism satisfied all possible desiderata. Fast, time-efficient mechanisms required different techniques from robust, self-repairing mechanisms. Small deformations could be created in a variety of easy ways, but large deformations required more sophisticated approaches. The best results, such as Chapter 5's tandem actuation, combined multiple low-level mechanisms to cover for each others' deficiencies. In many cases, compromises appear possible by combining multiple mechanisms, although much of the details remain to be elucidated.

Nonetheless, in spite of the diversity, the mechanisms studied here exhibit several powerful, unifying themes, even if differently instantiated in different cases. Two of the most prominent are energy minimization and constraint propagation.

6.1 Energy Minimization

A key tool that appeared again and again in this work was the use of energy minimization, whether virtual energy functions or physical mechanical energy. In Section 2.4, the natural process of physical energy minimization allowed us to construct complex, consistent 3d structures with the most minimal of blunt tools, control over the rate of cell proliferation. The structures themselves were not so much a record of material deposited by growth

processes (as, for example, in anthill fabrication [66]) as they were an energy-minimizing response of the substrate to particular mechanical provocations. Energy minimization freed us from having to worry about the minute details of smooth and mathematically consistent curvatures, at the cost of being able to specify structures only up to a certain level of precision.

In normal neighbors patterning (Chapter 4), I brought in energy minimization much more prominently, as a mechanism for solving soft constraint propagation subject to certain "well-behavedness" criteria. This time, the energy function was programmable, responsible for establishing and maintaining the key features of the pattern. The well-behavedness criteria and their resulting energy terms, however, particularly surface tension, served a similar role as mechanical energy did in Section 2.4, simplifying the level of detail that needed to be specified and ensuring smooth, self-consistent results.

For morphological homeostasis (Chapter 5), energy minimization was once again the great simplifier, in addition to its underlying role in the body plan produced by the normal neighbors patterning mechanism. Churn relaxation – energy minimization explicitly provoked through the injection of noise or hints – allowed surface curvature to be controlled without precise coordination of intrinsic and extrinsic actuation. Relaxation combiners – physical subunits whose sole control mechanism was energy minimization subject to boundary constraints – allowed heterogeneous modules to be combined into a single complex structure, shielding them from mutual mechanical interference and interpolating between them to form a minimally complex connector body. Slight adjustments to the mechanical energy properties of the boundaries even allowed the combiners themselves to be used as a building block. Energy minimization appeared at all levels of the system, serving similar, simplifying roles for each of the different layers.

In both normal neighbors patterning and morphological homeostasis (but not organized proliferation), the injection of noise (or the mean field equivalent) was critical. These systems operate at nonzero temperature, both for seeing over ridges in the energy landscape in order to reach the desired minima and for self-correcting in response to insults. When the minimum energy of the system, virtual or physical, can be forced to represent the desired structure, noise injection becomes a powerful tool both for convergence and self-repair.

Organized proliferation, by contrast, lacks a mechanism to force the global minimum of energy to represent the desired structure. It relies on catch-

ing the system in some satisfactory local minimum. As a result, though it may regrow new substructures, organized proliferation cannot self-correct an existing structure beyond the basic level provided as a consequence of the autonomous physics of the substrate (such as sealing holes when individual cells die). Injecting further noise will cause the details of the structure to melt away.

Of course, blind energy minimization is not everything. For example, scramble the cells of one of the multi-lobe structures of Chapter 5 and shut off the controller; will the desired structure re-assemble, even with mechanical noise injection? Unlikely. Trajectories do matter, as evidenced by the need for careful annealing in certain patterns. Reasonable perturbations can be repaired by blind energy minimization, but not *arbitrary* perturbations. An important challenge for ongoing work is how to effectively shepherd trajectories, whether for healing severe perturbations (which may require temperature to be locally or temporarily increased) or for guiding the system through mechanically difficult transitions, such as invagination.

An interesting and unusual feature of the energy minimization techniques shown here, particularly in the normal neighbors patterning mechanism, is that the energy function is modular and programmable. The normal neighbors mechanism is, in a sense, like a sort of artificial chemistry. By following simple "valence" rules, a wide, open-ended variety of complex structures can be built up. On the other hand, there is a limit to the complexity that can be practically implemented in a single pass, before "side reactions" (i.e. local minima) drive down the success rate. To improve the success rate, external scaffolding mechanisms ("catalysts") can be used to hint the desired structure, or the process can be broken down into multiple separate stages (both subjects of ongoing research). Occasionally, an unusual interaction akin to a resonance structure is observed (e.g. partial melting), for which the valence rules fail and additional special rules must be formulated.

6.2 Constraint Propagation

Constraint propagation proved to be a surprisingly apt language for expressing problems in distributed patterning. The self-timing strategy of Chapter 3 is, in essence, the expression of pattern computations in the form of hard, local, quasi-acyclic constraints, with a little bit of cleverness to ensure that the system is neither under-constrained nor over-constrained and will con-

verge by arc consistency alone. Partial information is passed around from neighborhood to neighborhood and repeatedly merged. Complete information – the set of possible values for a node narrowing down to a singleton – indicates local convergence.

The rule of normal neighbors, too, is all about the propagation of local constraints. There is, however, no simple rule for unfolding the answers; the system is loopy and badly under-constrained. Instead, values are conjured up through a local optimization process, combining soft constraints with incentives for well-behaved geometry. The system can also be locally over-constrained due to inconsistent hints and initial conditions, in which case the optimization process arbitrates among competing claims, providing a measure of robustness to contradictory information.

One of the great virtues of constraint propagation is the ability to run a computation in many different directions, exchanging inputs for outputs. This is used to great effect in normal neighbors patterning, allowing hints to be derived from any number of sources (reminiscent of the myriad hints that can be used to cajole stem cells to differentiate) and supporting regeneration of a partially destroyed pattern. The cost of breaking symmetry without any hints is slow convergence and a risk of defects if the pattern is complex and annealing conditions are poor. I observe that adding hints that fully break symmetry can dramatically improve speed of convergence.

In the self-timing approach, however, multi-directional computation is largely ignored. The constraints themselves can be multi-directional and are usually full of cycles between neighboring cells, but any given problem uses only one particular acyclic chain of dependencies. Such an inflexible approach is employed because there is no mechanism to arbitrate between incomplete or conflicting information and because the notion that components might fail randomly and asynchronously, necessitating recovery or regeneration of information, is impossible to reconcile with guarantees that answers are accurate according to the current state of the system: news of a failure takes time to arrive. Is it possible to generalize self-timing in a way that meaningfully accommodates failure recovery or that supports symmetry breaking for under-constrained problems? The answer is not yet clear.

The reverse problem also remains unsolved. Is it possible to reliably flag the completion of a symmetry-breaking feedback patterning mechanism such as the normal neighbors mechanism? This is a critical question, particularly for constructing cascaded and hierarchical patterning processes. Relying on conservative, hard-coded delays, though a possibility, would be a disappoint-

ing solution.

6.3 Biological Questions and Implications

In software engineering folklore, there is a design strategy known as the *Principle of Least Surprise* (or least astonishment). It posits that the best design is the one that surprises its user least, i.e. that is consistent with prior design patterns they are familiar with. In essence, it represents a form of implicit modularity: even when common patterns cannot be factored out into an explicit module, their consistency provides for a simplified representation in the mind of the human observer.

Despite the many similarities between engineered artifacts and natural organisms, as enumerated in Section 1.2.1, this is a major point of departure. In engineering, there is great benefit to solving different problems in similar ways, even when the commonality is not abstracted out, because the modularity still exists in the mind of the engineer. In biology, while it is quite common to reuse existing components for new solutions, solutions developed independently using independent parts need not bear any similarity at all.

On the other hand, different-looking mechanisms may nonetheless share deep mathematical similarities, because of constraints imposed by mathematics and by physical phenomena. Should we expect biological processes to follow the particulars of the mechanisms proposed in this thesis? Perhaps not. But, the mathematical phenomena demonstrated may well be quite common. Constraint propagation and annealing, energy minimization, and closed-loop morphological control are worth looking out for.

In many cases, these abstract mathematical mechanisms should have observable experimental signatures. Mean field annealing is suggested by a sigmoidal transition in upstream cell fate indicators across a region boundary that steadily sharpens over time. Closed-loop morphological control is indicated when an otherwise resilient morphology can be disrupted by experimentally manipulating only the behavior of a mechanical sensor. Constraint propagation is a natural model of mutual induction, as, for example, if a swap of tissue grafts transforms both recipient regions to be more similar to their corresponding donor environments. Such effects are well known in the case of organizers, and as a practical matter, some tissues will be more powerful than others at propagating their influences. Successfully demonstrating that most tissues have some organizing influence, however, even if weak, would

be strong support for the constraint model.

Additionally, closed-loop control is not merely a strategy for robust development, it is also a strategy for regeneration. As the results of Chapter 5 suggest, robust, canalized developmental processes may already be 90% of the mechanism necessary for regeneration. Regeneration is not necessarily some complicated, add-on feature evolved only in exceptional circumstances. With closed-loop morphological control, instead, it may be the default, and rapid procedures for blood loss prevention and infection control are the sophisticated addition. Mammalian embryos, for example, show impressive, scar-free wound healing; it is only with maturity that scar formation occurs. Evolutionary pressure for good surgical wound healing or recovery from a violent limb amputation probably never existed; infection, foreign objects, and blood loss, on the other hand, were very real threats [22]. Of course, there is much more to investigate; I have only explored single-stage development using an existing population of cells, not the complex, multi-stage development characteristic of higher animals. I expect the principle will generalize, but questions remain.

In biology, common mathematical motifs aside, not only is there little reason to solve different problems in similar ways, there is also little reason to solve each problem in one way only. If the inclusion of multiple solutions provides any benefit, without introducing embrittling redundancy that impedes evolvability, one can reasonably expect that multiple solutions will indeed be found; partial redundancy is a common sight in biology. Such redundancy may even improve evolvability, by allowing a wider space of initial embryo conditions to converge to the desired state. One of the important observations made in this thesis is that, in patterning, and especially in the case of pattern-driven mechanical deformation, there *is* significant benefit to having redundant mechanisms.

This observation suggests the possibility that a serious lamppost problem may be obscuring the mechanistic side of developmental biology. In experiments, model organisms are overwhelmingly favored, and good model organisms develop extremely rapidly and reliably even in artificial conditions. Yet, model organisms like drosophila are far from simple. Do organisms like drosophila really make sense as models for the physical aspects of development, given that they are optimized for rapid, reliable development and exhibit highly derived developmental features (e.g. syncytial development)? One should expect to find therein not the simplest, essential mechanisms for mechanical development but instead a wealth of partially redundant, in-

teracting mechanisms, for the sake of improved speed and reliability. My model suggests that experiments aimed at identifying the essential mechanical drivers are likely to be misleading. In search of the deep principles and simplest, essential elements, one should expect more clarity from organisms that develop slowly under highly stable environments, lacking evolutionary pressure to pile on redundancy.

6.4 Going Forward

In this thesis, I showed how to attack the problem of patterning geometric form in a deformable substrate. I developed a computational model for deformable cellular surfaces and showed how simple agent programs running within the individual cells can orchestrate the development of large-scale structures. I showed how small deformations could be produced with a variety of techniques, including naive pre-patterning, but how large deformations required more sophisticated approaches, implying simultaneous patterning and deformation. I showed how self-correcting patterning and closed-loop geometry control can reliably produce structures using large deformations, while also featuring both symmetry breaking and self-repair.

I hope I have inspired biologically-inclined readers to ponder whether, where, and how the various mechanisms described may appear in living organisms – and to question whether conventional explanations of what known developmental networks accomplish are sufficient to describe a robust artifact living and evolving in the real world. Even if the particular mechanisms suggested never appear, understanding how the hurdles and bugs are avoided could furnish a fine, new set of questions.

I hope I have inspired roboticists to consider the potential of distributed, deformable robotic substrates. We need not restrict our imaginations to rectilinear arrangements of rigid bodies. Development is, after all, a natural example of swarm soft robotics.

At present, synthetic biologists have their hands full trying to tame cells into robust, scalable computational environments, tractable through forward engineering techniques and computer-aided design. I hope, however, that I have mapped out some of the path ahead, for when we design synthetic cells to grow into microstructures and engineered tissues.

Even within this semi-theoretical line of inquiry, the questions are far from resolved. Can self-timing and self-stabilization be reconciled? How might one

demonstrate multi-stage yet self-repairing development? Can these strategies be extended to iterated features and recursive structures? By what means can closed-loop developmental trajectories be shepherded through narrow, difficult transitions?

Somehow, nature has solved these problems. Can we?

Bibliography

[1] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman, and R. Weiss. Amorphous computing. Technical Report AIM-1665, MIT, 1999.

[2] Andrew Adamatzky, Benjamin De Lacy Costello, and Tetsuya Asai. *Reaction-diffusion computers*. Elsevier, 2005.

[3] Jonathan Bachrach, Jacob Beal, and James McLurkin. Composable continuous space programs for robotic swarms. *Neural Computing and Applications*, 19(6):825–847, 2010.

[4] Subhayu Basu, Yoram Gerchman, Cynthia H. Collins, Frances H. Arnold, and Ron Weiss. A synthetic multicellular systems for programmed pattern formation. *Nature*, 434:1130–1134, April 2005.

[5] Jacob Beal. Persistent nodes for reliable memory in geographically local networks. Technical Report AIM-2003-11, MIT, 2003.

[6] Jacob Beal. Programming an amorphous computational medium. In *Unconventional Programming Paradigms International Workshop*, September 2004.

[7] Jacob Beal. Functional blueprints: An approach to modularity in grown systems. In *International Conference on Swarm Intelligence*, 2010.

[8] Jacob Beal and Jonathan Bachrach. Infrastructure for engineered emergence in sensor/actuator networks. *IEEE Intelligent Systems*, pages 10–19, March/April 2006.

[9] Guy B. Blanchard, Alexandre J. Kabla, Nora L. Schultz, Lucy C. Butler, Benedicte Sanson, Nicole Gorfinkiel, L. Mahadevan, and Richard J.

Adams. Tissue tectonics: morphogenetic strain rates, cell shape change and intercalation. *Nature Methods*, 6(6):458–464, 2009.

[10] G. Wayne Brodland. Do lamellipodia have the mechanical capacity to drive convergent extension? *International Journal of Developmental Biology*, 50:151 – 155, 2006.

[11] G. Wayne Brodland, Daniel I-Li Chen, and Jim H. Veldhuis. A cell-based constitutive model for embryonic epithelia and other planar aggregates of biological cells. *International Journal of Plasticity*, 22(6):965 – 995, 2006.

[12] Micah Brodsky. Self-timed patterning. In *7th International Workshop on Spatial Computing (SCW 2014)*, May 2014.

[13] Xiaoguang Chen and G. Wayne Brodland. Multi-scale finite element modeling allows the mechanics of amphibian neurulation to be elucidated. *Physical Biology*, 5(1):015003, 2008.

[14] D. A. Clausi and G. W. Brodland. Mechanical evaluation of theories of neurulation using computer simulations. *Development*, 118(3):1013–1023, 1993.

[15] Daniel Coore. *Botanical Computing: A Developmental Approach to Generating Inter connect Topologies on an Amorphous Computer*. PhD thesis, MIT, 1999.

[16] L.A. Davidson, M.A. Koehl, R. Keller, and G.F. Oster. How do sea urchins invaginate? Using biomechanics to distinguish between mechanisms of primary invagination. *Development*, 121(7):2005–2018, 1995.

[17] R. Doursat. Programmable architectures that are complex and self-organized: from morphogenesis to engineering. In S. Bullock, J. Noble, R. Watson, and M. A. Bedau, editors, *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 181–188. MIT Press, Cambridge, MA, 2008.

[18] Rene Doursat. The growing canvas of biological development: Multiscale pattern generation on an expanding lattice of gene regulatory networks. *InterJournal: Complex Systems*, 1809, 2006.

[19] René Doursat, Carlos Sánchez, Razvan Dordea, David Fourquet, and Taras Kowaliw. Embryomorphic engineering: Emergent innovation through evolutionary development. In *Morphogenetic engineering: toward programmable complex systems*, pages 275–311. Springer, 2012.

[20] René Doursat, Hiroki Sayama, and Olivier Michel. *Morphogenetic engineering: toward programmable complex systems*. Springer, 2012.

[21] A. Eldar, B. Z. Shilo, and N. Barkai. Elucidating mechanisms underlying robustness of morphogen gradients. *Current Opinion in Genetics & Development*, 14(4):435–9, August 2004.

[22] Mark W. J. Ferguson and Sharon O'Kane. Scar-free healing: from embryonic mechanisms to adult therapeutic intervention. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 359(1445):839–850, 2004.

[23] V. French. Pattern regulation and regeneration. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 295(1078):601–617, 1981.

[24] Chikara Furusawa and Kunihiko Kaneko. Robust development as a consequence of generated positional information. *Journal of Theoretical Biology*, 224(4):413 – 435, 2003.

[25] Jean-Louis Giavitto and Olivier Michel. MGS: a rule-based programming language for complex objects and collections. *Electronic Notes in Theoretical Computer Science*, 59(4):286–304, 2001.

[26] Lorna J. Gibson and Michael F. Ashby. *Cellular solids*. Cambridge University Press, 1997.

[27] Alfred Gierer, S. Berking, H. Bode, Charles N. David, K. Flick, G. Hansmann, H. Schaller, and E. Trenkner. Regeneration of hydra from reaggregated cells. *Nature/New Biology*, 239(88):98–101, 1972.

[28] S.C. Goldstein, J.D. Campbell, and T.C. Mowry. Programmable matter. *Computer*, 38(6):99–101, June 2005.

[29] François Graner and James A. Glazier. Simulation of biological cell sorting using a two-dimensional extended Potts model. *Phys. Rev. Lett.*, 69:2013–2016, September 1992.

[30] Viktor Hamburger and Howard L. Hamilton. A series of normal stages in the development of the chick embryo. *Journal of Morphology*, 88(1):49–92, 1951.

[31] Mahiar Hamedi, Robert Forchheimer, and Olle Inganas. Towards woven logic from organic electronic fibres. *Nature Materials*, 6:357–362, 2007.

[32] E. Hawkes, B. An, N. M. Benbernou, H. Tanaka, S. Kim, E. D. Demaine, D. Rus, and R. J. Wood. Programmable matter by folding. *Proceedings of the National Academy of Sciences*, 107(28):12441–12445, 2010.

[33] Wilhelm His. *Unsere Krperform und das physiologische Problem ihrer Entstehung*. Verlag von F.C.W. Vogel, 1874.

[34] Antone G. Jacobson, George F. Oster, Garrett M. Odell, and Louis Y. Cheng. Neurulation and the cortical tractor model for epithelial folding. *Journal of Embryology and Experimental Morphology*, 96(1):19–49, July 1986.

[35] Seunghee S. Jang, Kevin T. Oishi, Robert G. Egbert, and Eric Klavins. Specification and simulation of synthetic multicelled behaviors. *ACS Synthetic Biology*, 1(8):365–374, 2012.

[36] Marc W. Kirschner and John C. Gerhart. *The Plausibility of Life: Resolving Darwin's Dilemma*. Yale University Press, 2005.

[37] Attila Kondacs. Biologically-inspired self-assembly of 2d shapes, using global-to-local compilation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[38] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, July 1978.

[39] Lev Davidovich Landau and Eugin M. Lifshitz. *Course of Theoretical Physics Vol 7: Theory of Elasticity*. Pergamon Press, 3rd edition, 1986.

[40] Yuri Lazebnik. Can a biologist fix a radio? – or, what I learned while studying apoptosis. *Cancer Cell*, 2(3):179 – 182, 2002.

[41] Hanying Li, Joshua D. Carter, and Thomas H. LaBean. Nanofabrication by DNA self-assembly. *Materials Today*, 12(5):24 – 32, 2009.

[42] Hans Meinhardt. Biological pattern formation as a complex dynamic phenomenon. In Alessandra Carbone, Misha Gromov, and Przemyslaw Prusinkiewicz, editors, *Pattern Formation in Biology, Vision, and Dynamics*, pages 99–132. World Scientific, 2000.

[43] Jay E. Mittenthal. The rule of normal neighbors: A hypothesis for morphogenetic pattern regulation. *Developmental Biology*, 88(1):15 – 26, 1981.

[44] Alyssa S. Morgan and Daniel N. Coore. Modeling intertia in an amorphous computing medium. In *The 6th International Workshop on Spatial Computing (SCW 2013)*, May 2013.

[45] Radhika Nagpal. *Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics*. PhD thesis, MIT, 2001.

[46] Radhika Nagpal, Ankit Patel, and Matthew C. Gibson. Epithelial topology. *BioEssays*, 30(3), 2008.

[47] Andrew C. Oates, Luis G. Morelli, and Saúl Ares. Patterning embryos with oscillations: structure, function and dynamics of the vertebrate segmentation clock. *Development*, 139(4):625–639, 2012.

[48] G. M. Odell, G. Oster, P. Alberch, and B. Burnside. The mechanical basis of morphogenesis : I. epithelial folding and invagination. *Developmental Biology*, 85(2):446 – 462, 1981.

[49] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

[50] M. I. Rabinovich, A. B. Ezersky, and P. D. Weidman. *The Dynamics of Patterns*. World Scientific, 2000.

[51] Alexey Radul. *Propagation Networks: A Flexible and Expressive Substrate for Computation*. PhD thesis, MIT, 2009.

[52] Ashok Ramasubramanian, Kimberly Latacha, Jessica Benjamin, Dmitry Voronov, Arvind Ravi, and Larry Taber. Computational model for early cardiac looping. *Annals of Biomedical Engineering*, pages 1–15, 2006.

[53] Ashok Ramasubramanian and Larry A. Taber. Computational modeling of morphogenesis regulated by mechanical feedback. *Biomechanics and Modeling in Mechanobiology*, 7(2), 2007.

[54] J. M. W. Slack. *From Egg to Embryo: Regional Specification in Early Development (Developmental and Cell Biology Series)*. Cambridge University Press, 1991.

[55] Jens Sparsø and S. Furber. *Principles of Asynchronous Circuit Design: A Systems Perspective*. European low-power initiative for electronic system design. Springer, 2001.

[56] Antoine Spicher and Olivier Michel. Declarative modeling of a neurulation-like process. *BioSystems*, 87:281–288, February 2006.

[57] Malcolm S. Steinberg. Reconstruction of tissues by dissociated cells. *Science*, 141(3579):401–408, 1963.

[58] Gerald Jay Sussman and Alexey Radul. The art of the propagator. Technical Report MIT-CSAIL-TR-2009-002, MIT CSAIL, January 2009.

[59] L. A. Taber. Compression of fluid-filled spherical shells by rigid indenters. *Journal of Applied Mechanics*, 50(4a):717–722, 1983.

[60] Xiaoming Tao, editor. *Smart fibres, fabrics, and clothing*. CRC Press, 2001.

[61] D'Arcy Wentworth Thompson. *On growth and form*. Cambridge University Press, 1945.

[62] A. M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72, 1952.

[63] C. H. Waddington. Canalization of development and the inheritance of acquired characters. *Nature*, 150:563–565, November 1942.

[64] Denis Weaire and Stefan Hutzler. *The Physics of Foams*. Oxford University Press, 1999.

[65] Ron Weiss. *Cellular Computation and Communications using Engineered Genetic Regular Networks*. PhD thesis, MIT, 2001.

[66] Justin Werfel. *Anthills built to order: Automating construction with artificial swarms*. PhD thesis, MIT, 2006.

[67] Justin Werfel, Kirstin Petersen, and Radhika Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science*, 343(6172):754–758, 2014.

[68] Geoffry B. West and James H. Brown. Life's universal scaling laws. *Physics Today*, 57(9), 2004.

[69] Erik Winfree. On the Computational Power of DNA Annealing and Ligation. In *DNA Based Computers, volume 27 of DIMACS*. American Mathematical Society, 1995.

[70] Erik Winfree. Simulations of computing by self-assembly. In *Proceedings of the 4th DIMACS Meeting on DNA Based Computers*, June 1998.

[71] Daniel Yamins. *A Theory of Local-to-Global Algorithms for One-Dimensional Spatial Multi-Agent Systems*. PhD thesis, Harvard, December 2007.

[72] Jonathan S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, July 2005.

[73] Mark Yim, Paul White, Michael Park, and Jimmy Sastra. Modular self-reconfigurable robots. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 5618–5631. Springer New York, 2009.