Form From Within: Scaling Up Self-Constructing Biological Architectures through a Novel Application of Synthetic Morphogenesis

by
Gizem Gumuskaya
B.Arch., Istanbul Technical University (2015)
Submitted to the Department of Architecture and the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of
Master of Science in Architecture Studies
and
Master of Science in Electrical Engineering and Computer Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September 2018

Signature redacted

Author .....
Department of Architecture and the
Department of Electrical Engineering and Computer Science
August 10, 2018

Certified by.... Signature redacted .....
Ron Weiss
Professor of Biological Engineering and Electrical Engineering and Computer Science
Thesis Supervisor

Certified by... Signature redacted .....
George Stiny
Professor of Design and Computation
Thesis Supervisor

Accepted by . Signature redacted .....
Sheila Kennedy
Professor of Architecture
Chair, Department Committee on Graduate Studies

Accepted by Signature redacted .....
Leslie Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Thesis Committee

Thesis Co-Advisors:

Ron Weiss, Professor of Biological Engineering and Electrical Engineering and Computer Science

George Stiny, Professor of Design and Computation

Thesis Reader:

Terry Knight, Professor of Design and Computation

Form From Within: Scaling Up Self-Constructing Biological Architectures through a Novel Application of Synthetic Morphogenesis

by

Gizem Gumuskaya

Submitted to the Department of Architecture and the
Department of Electrical Engineering and Computer Science
on August 10, 2018, in partial fulfillment of the
requirements for the degrees of
Master of Science in Architecture Studies
and
Master of Science in Electrical Engineering and Computer Science

Abstract

In this thesis, I introduce a novel biofabrication method *Architectures from Staged Self-assembly of Morphogenetic Building Elements (ASSEMBLE)*, that brings about self-constructing biological structures at the architectural scale by merging scientists' newly developing ability to control the morphogenetic power of living matter with architects' and builders' discrete assembly method, which they have used for centuries to scale up their structures.

ASSEMBLE arose from a recognition that in nature, simple building blocks, such as biological cells, self-organize into higher-order, complex structures with no descriptive blueprints at hand and no intelligent designer telling them what to do; instead, they execute a set of generative rules encoded in their DNA. By editing these rules, synthetic biologists can now program living cells to undergo *synthetic morphogenesis*, and thereby construct higher-order structures *by design*. However, so far, the biggest programmable structures we have developed in this way are merely on the order of millimeters, which is too small to be relevant in architectural practice. ASSEMBLE bridges this gap by employing these millimeter-scale structures as *morphogenetic building elements* that can self-assemble with one another through a set of physical assembly cues they are programmed grow on their surfaces. To identify which assembly cues needed on a group of morphogenetic building elements for them to self-assemble into a target structure, I also introduce a 3D global-to-local structural compiler.

In this way, ASSEMBLE enables us to create self-constructing architectures by exploiting biological cells as an infinite supply of building material, into which desired structural

specifications can be directly encoded via the generative language of DNA. Using this approach, we can employ the morphogenetic power of biological cells in architecture to not only overcome the limitations of current fabrication methods, but also to start building in entirely novel environments, such as the outer space, which do not permit the use of our current top-down approaches since we can transport neither the hefty construction materials, nor the machinery to such remote sites. The bottom-up architectural fabrication approach introduced in this thesis can finally enable us to build in extra-terrestrial sites by leaving the orbit only with a tube of engineered cells.

Thesis Supervisor: Ron Weiss
Title: Professor of Biological Engineering and Electrical Engineering and Computer Science

Thesis Supervisor: George Stiny
Title: Professor of Design and Computation

Acknowledgments

I would first like to thank my thesis advisors Prof. Ron Weiss and Prof. George Stiny, whose doors were always open to me throughout this challenging process, as well as my thesis reader Prof. Terry Knight.

I would then like to acknowledge my colleagues in the MIT Weiss Lab, especially Katherine Kiwimagi and Jesse Tordoff, for their contributions to the project FACETS, as well as to my growth as a synthetic biologist. I also would like to acknowledge my colleague at the Harvard University Department of Computer Science, Saketh Rama for his contributions to the project ASSEMBLE, especially to its global-to-local compiler component. His knowledge of and enthusiasm for self-assembly was a great source of inspiration and encouragement as I set out to pursue this project.

I thank the United States Department of Defense Advanced Research Projects Agency (DARPA) as well as MIT Department of Architecture for providing funding during my three years of graduate education at MIT.

I would also like to thank the experts at the MIT Writing and Communication Center Marilyn Levine and Rebecca Thorndike-Breeze for their rigorous and passionate participation in helping me put my thoughts down into the pages of this thesis document, as well as proofreading it with diligence.

I also acknowledge with gratitude the Department of Architecture Graduate Administrator Cynthia Stewart and Department of Biological Engineering Senior Administrative Assistant Olga Parking for their endless patience and help. Furthermore, I thank from the bottom of my heart to Blanche Staton, Senior Associate Dean for Graduate Education, as well as Assistant to the Senior Associate Dean Patricia Glidden, for helping me navigate through some of my hardest times at MIT.

Finally, I must express my very profound gratitude to my parents Betul and Hayrettin Gumuskaya, as well as my siblings Merve and Eren Gumuskaya for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Author

Gizem Gumuskaya

Contents

Chapter 1

Introduction

Whether the development of an embryo or the formation of an indigenous city, construction in nature is a self-organizing process governed by a set of *rules* embodied within the builders or building blocks themselves. Notably, these rules are not *descriptive* like a blueprint, specifying a final architecture; instead, they are *generative* like a series of if-then statements, allowing these builders to make autonomous decisions based on local cues they encounter, and thereby self-organize into orderly architectures.

Take, for example, the development of an embryo from a single cell. The initial building block of the embryo, this single cell, within its *merely ten microns-wide* nucleus encapsulates a *two meters-long* DNA strand encoding the generative rules necessary for it to self-replicate and develop into a dynamic, multicellular, living structure via the process of morphogenesis.

1.1 Synthetic Morphogenesis

With the advent of synthetic biology [1], and more specifically with the emerging field of *synthetic morphogenesis* [2], we have entered an era where we not only can read but also *edit* these generative rules stored in a cell's DNA. By way of editing these rules, we begin

to produce programmable multi-cellular structures *of our own design* with the structural specifications, such as shape and materiality, directly encoded in a single precursor cell [3] [4]. However, currently, the biggest multi-cellular structures we can grow using this way are at the millimeter scale only, which is too small to be relevant in architectural scale all by itself. Therefore, in this thesis, I introduce a novel application of synthetic morphogenesis, where I employ each one of these programmable multi-cellular structures as a distinct building element, which I term a *morphogenetic building element*, and facilitate their attachment to one another via *physical assembly cues* programmed into them. To identify what kind of assembly cues are needed on these morphogenetic building elements for them to reliably self-assembly into an arbitrary target structure, I also introduce a global-to-local structural compiler. I name this method *Architectures from Staged Self-assembly of Morphogenetic Building Elements (ASSEMBLE)*, which, in this way, takes advantage of the discrete assembly method architects and builders have been using to scale up their structures for centuries, and marries it to scientists' newly developing ability to control the morphogenetic power of living matter. In doing so, ASSEMBLE enables us to create self-constructing structures in architectural scale with desired structural specifications directly encoded in the building material itself (i.e., cells) using the generative language of DNA.

For this purpose, in the framework of ASSEMBLE, I also introduce a global-to-local structural compiler, which, preceding the fabrication process in the wetlab, inputs a descriptive 3D model of the target structure prepared by the user, and compiles it into the generative rules that a cell can execute. In this way, by translating descriptive blueprints to generative rules, this structural compiler enables us to encode the structural specifications of a target architecture into a single cell's DNA. This single cell then grows into an army of builders collectively constructing the target architecture.

ASSEMBLE is nether the first one to propose programming cells to construct multi-cellular structures in a petri dish [5], nor the first one to suggest utilizing biological cells to construct

architectural building blocks. As noted, the field of synthetic morphogenesis has already been repurposing the self-organizational capacity of cells to build artificial organoids "in-vitro" [5] [6]; and architects have already been using top-down fabrication methods, such as 3D printing or molding, to shape biomaterials produced by organisms including bacteria and fungi into bricks, or furniture [7] [8]. Yet, what is novel about ASSEMBLE is that it scales-up the size-wise limited morphogenesis approach into lengths where biofabrication has so far only been possible via top-down shape-giving equipment. ASSEMBLE achieves this not only by integrating the scientific method of bottom-up synthetic morphogenesis and the architectural method of top-down discrete assembly, but also by investigating the fundamental difference between the *generative* way nature constructs itself and the *descriptive* way human architects fabricate.

Multi-cellular structures construct themselves using the *generative language* of DNA, whereas architects and engineers build using the *descriptive language* of blueprints. The latter is extremely suitable for the top-down way in which we have been constructing our buildings and cities, but it is simply not the language cells understand [9]. Current efforts in the biodesign field in general continue to overlook this fundamental difference and hence fail to exploit living matter's ability to generate *form from within.* They fabricate with machinery, such as a mold or a 3D printer, that instead generate *form from imposition.* As a result, their products look *as if* they are formed by a bottom-up, self-organizing, and generative construction process, whereas they are in fact fabricated with the same old top-down methods and mentality that has nothing to do with the generative language by which nature constructs itself.

1.2 Blind Origamist

To better understand why this dichotomy between the descriptive vs generative languages is so critical to harnessing biological systems' ability to generate form from within, which is arguably the biggest advantage of using living matter as building material, one can think of nature as a blind origamist, akin to Dawkins' blind watchmaker. This blind origamist builds intricate architectures by following habitual formulae for *how* she should fold the paper, analogous to generative rules. The origamist does not have a specific goal; it just does what it does and determines which rules to employ via a random process of trial-and-error. Now, what happens if we try to show this blind origamist a picture of an intricate shape and ask her to recreate it with a flat piece of paper? The blind origamist would have no way of understanding what shape is shown on the picture; she is simply unable to communicate in that descriptive visual language —let alone able to come up with the set of folding rules. To get the blind origamist build our target shape, we have to be the ones to compile the shape into a set of folding instructions, with which she can then work with.

Similarly, biological cells, which can build intricate architectures by following a set of generative rules, have no way of understanding the descriptive language with which we represent our target shape. Therefore, we need to come up with a way to compile the descriptive language in which we represent our purposeful designs into the generative language by which cells can construct them. Only then, as designers, we can that exploit what living matter, and only living matter, has to offer, such as programmable self-organization of an infinite material supply. Acknowledging and working with this fundamental difference between biology and architecture, instead of pretending like it does not exist, can enable us to more intelligently bridge the gap between the biological scale of microns and the architectural scales of centimeters and meters. In this way, by scaling up the self-constructing biological structures into the architecturally-relevant scales of centimeters, and even meters, we not

only can sustainably and profitably mass fabricate architectural components but we can also build in ways that were previously impossible. For example, with such self-replicating and self-organizing building elements, we can now build with ease in remote sites, extreme environments and even in outer space [10], because we no longer need to transport hefty construction materials or machinery to these remote sites. All we need is a tube of engineered cells!

Roadmap

This thesis is organized around five chapters. In the Chapter 2, *Generative Rules: A New Biodesign Heuristic*, I introduce case studies to provide a background for the generative way in which nature constructs itself across orders of magnitude. In the Chapter 3, *Methodology*, I introduce the term *Generative Spatial Computation* to formally describe this computational construction process and argue that editing these generative rules can enable us to edit the resulting high-level constructs. To demonstrate this claim, I introduce the project *FACETS*, where a new set of generative rules are introduced into a single cell for it to then undergo synthetic morphogenesis and develop into a human-designed, artificial structure. Then, by introducing *ASSEMBLE* and its compiler component *LACE*, I show how the programmed self-assembly of FACETS into arbitrary target structures can be facilitated. In the following Chapter 4, *Results and Discussion*, I introduce my findings from my wetlab experiments to demonstrate how such generative rules can be turned into DNA sequences and introduced into cells for them to execute artificial generative spatial computation. Finally, in Chapter 5, *Conclusion*, I summarize ASSEMBLE and elaborate on the practical and intellectual contributions of such a novel biofabrication method for growing structures that embody the rules necessary to build themselves.

Chapter 2

Generative Rules: A New Biodesign Heuristic

At one time, even scientists believed that all self-organizing building activities in nature, such as the growth of a tree or the healing of a wound, are governed by an inexplicable, enigmatic force —a belief system known as vitalism[1] [11]. Today we know that not only are there rational mechanisms underlying these activities, but also that these mechanisms are not entirely ad-hoc, or unique to each building activity. Using a set of core principles, nature constructs itself across scales and in different contexts —whether this is the folding of a protein, construction of an insect nest, development of an embryo, or the formation of an indigenous city [Fig2-1].

[1]Even a scientist as prominent as Pasteur was a vitalist, which caused him a lot of confusion regarding his own scientific findings that instead were suggesting an underlying systematism in metabolic processes.

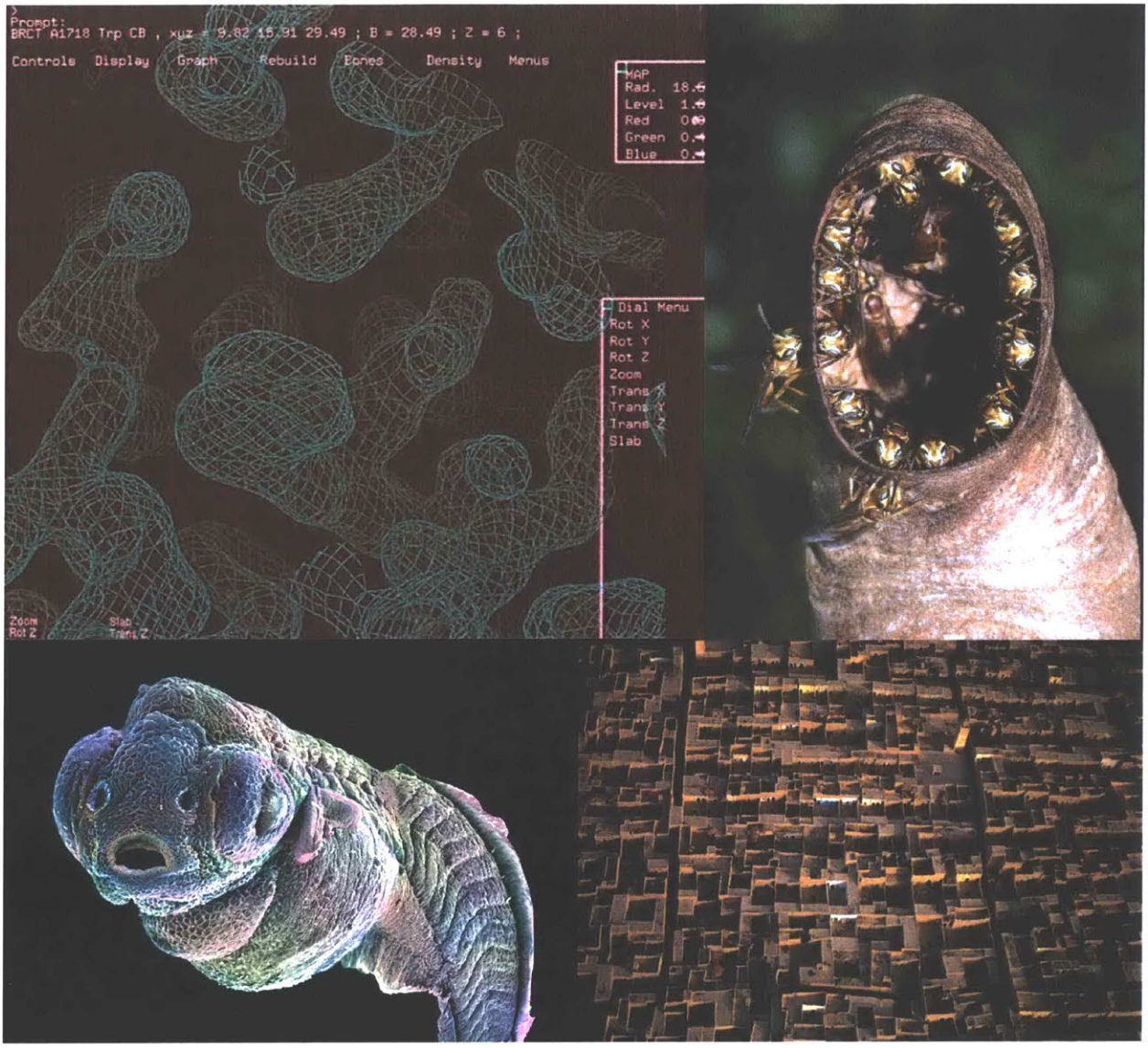


Figure 2-1: (a)An electron density map of a protein showing its constituent aminoacids. (b)A wasp's nest, collectively built by the colony. (c)An SEM image of a two days old zebra fish embryo developed out of a single cell. (d)An aerial photograph of a self-organizing indigenous city. *Image credits:Michael Yaffe, unknown, Annie Cavanagh, Yann Arthus-Bertrand*

2.1 Five Principles of Self-organizing Structures in Biological Systems

Here I introduce the five principles of self-organizing structures in biological systems and discuss them in the context of four systems across scales and contexts: proteins, wasp nests, embryos, and indigenous settlements. Then, in the rest of this chapter, I introduce each one of these systems in more detail to provide a comprehensive background for the generative language by which such biological systems construct themselves.

I. Self-Organizing structures in nature are built by numerous interacting builders.

Self-organizing structures are built by a family of builders that are in continuous interaction and co-operation with one another. These builders are often structurally similar and genetically related to one another.

> *Proteins are formed by hundreds of amino acids interacting and making bonds with one another; wasps' nests by a colony working cooperatively via stigmergy; embryos by thousands of constituent cells sending signals and turning each others' genes ON or OFF; and indigenous cities by the collective efforts of their inhabitants.*

II. Builders are ruled by distributed governance, rather than by a single leader.

Despite the high number of these builders, none of them is the leader or the master builder —they are all master builders.

> *There is no "amino acid-in-chief" commanding the rest of the amino acids to the final protein configuration; no single wasp giving building instructions to the*

colony[2]*; no embryonic cells forming a "developmental headquarters"*[3]*; and no master architect designing settlements in indigenous communities.*

III. Building activity is driven by local responsiveness, not global awareness.
Builders construct "bottom-up" by solely responding to local cues, which may be physical, chemical, mechanical, or visual; often are not even aware of what the emerging global structure.

Rather than having an agenda for building a complex protein, constituent amino acids simply respond to the local physical forces acting on them; rather than worrying about what the nest will end up looking like, wasps remain narrowly focused on organizing building blocks into their appropriate local configurations; rather than following a global blueprint, embryonic cells develop as a result of local chemical or mechanical signals they receive from one another; rather than relying on, or even having access to, city planning tools, indigenous builders are influenced by the positioning of the neighboring dwellings.

IV. Builders follow a set of generative rules, rather than descriptive blueprints.
To process these local cues, each builder uses a set of systematic rules, which enables them to make a decision about their next move towards the completion of the structure. Rather than strict building instructions, these rules make up a versatile generative code, which can be formulated through a formal language such as if-then statements or finite state-machine

[2]There is the queen wasp but she does not partake in the building activities, her main job is to ensure reproduction.

[3]Later in this chapter we will discuss how a group of embryonic cells can form a cluster called "the Spemann Organizer" in the developing embryo. Contrary to what its name may evoke, the organizer cluster is not a *control base* organizing the rest of the embryo; instead, it solely organizes its neighbors to collectively construct the central nervous system. Similarly, the central nervous system itself, especially the brain, is often colloquially referred as the control center governing the rest of the organism. However, with the recent studies on microbiome and gut-brain axis, we now know that other parts of the body controls the brain as much as the brain controls them.

diagrams[4].

> *Thousands of different proteins are built by the same set of 20 amino acids coming together in different ways; each amino acid determines what position it should assume by attending the physical forces surrounding it and by following a set of generative rules. Different construction techniques are required to build different parts of the wasp nest; each wasp determines which construction technique to use next by probing the local configuration of the existing building blocks at the emerging front line of the structure and by following a set of generative rules. Vastly different parts of the embryo are built by the cells that carry the same DNA; each cell determines which genes to activate for the construction of a specific body part by processing the local signals and by following a set of generative rules. A primitive settlement is made up of standard and repetitive dwelling units; the builder of each unit determines how to articulate the adjacent unit by examining the local orientation of its neighbors' entrance ways and by following a set of generative rules.*

V. The entire building process is governed by *spatial computation* rooted in bodily interactions, rather than symbolic thinking.

Generative rules for building are physically encoded within the builders themselves, and are thus executed bodily, bringing about a computational activity that takes place at the spatial interface between the structure and the builders constructing it. Unlike computation as we know it, which relies heavily on symbolic processes unfolding in an abstract realm (i.e., silicon-based computing), in nature computation is a corporeal activity unfolding in a

[4]For the purposes of this Background Chapter, generative rules introduced here will be those that can be formulated via if-then statements. Later in Chapters 3 and 4, I will introduce the more dynamic and complex generative rules that can be best represented with a state-machine diagrams

material world.

> *During the construction of a protein, generative rules followed by the constituent amino acids originate in and are enforced by the electron cloud surrounding them. During the building of a wasp nest, although it is not known whether generative rules followed by a wasp are cognitively entrenched or bodily intuited, we do know that these rules are internal to each wasp. During the development of an embryo, generative rules followed by embryonic cells are encoded in each cell's DNA stored in its nucleus. During the formation of a primitive settlement, generative rules followed by the inhabitants are not being drawn from a kind of construction manual, but rather from mechanical logic, aesthetic customs and local resources.*

Understanding these five complex but fundamental principles is essential to understanding how nature constructs itself. Another way to explain these five principles, besides examining them in their natural context as we have already started to do above, is to look at them in a simpler and more abstract setting, such as in a virtual cellular automata. One example of such a system is the *Game of Life*, created by mathematician John Conway in 1970 in the form of an autonomously evolving multi-agent system [12]. A closer look at this system below will show that even in such a virtual, insentient context, these five principles alone are enough to generate self-organizing structures, patterns and behaviors that are very similar to those we observe in biological systems.

Game of Life is a cellular automata unfolding on an infinite 2D grid, where every pixel is an agent annotated as a "cell" (i.e., an automaton) whose "purpose" is to come alive and survive by interacting with its eight neighboring cells based on the set of generative rules shown below. Even though each cell is solely motivated by its survival and is entirely operating at the local level, its interactions with neighboring cells start to generate highly

complex and orderly spatial organizations at the global level [Fig2-2], of which none of the cells are aware.

rule I

If a blank cell has exactly 3 alive neighbors, **then** it can come alive (i.e., the three neighbors "give birth" to the cell).

rule II

If an alive cell has 2 or 3 alive neighbors, **then** it can survive.

rule III

If an alive cell has less than 2 alive neighbors, **then** it dies due to underpopulation.

rule IV

If an alive cell has more than 3 alive neighbors, **then** it dies due to overpopulation.

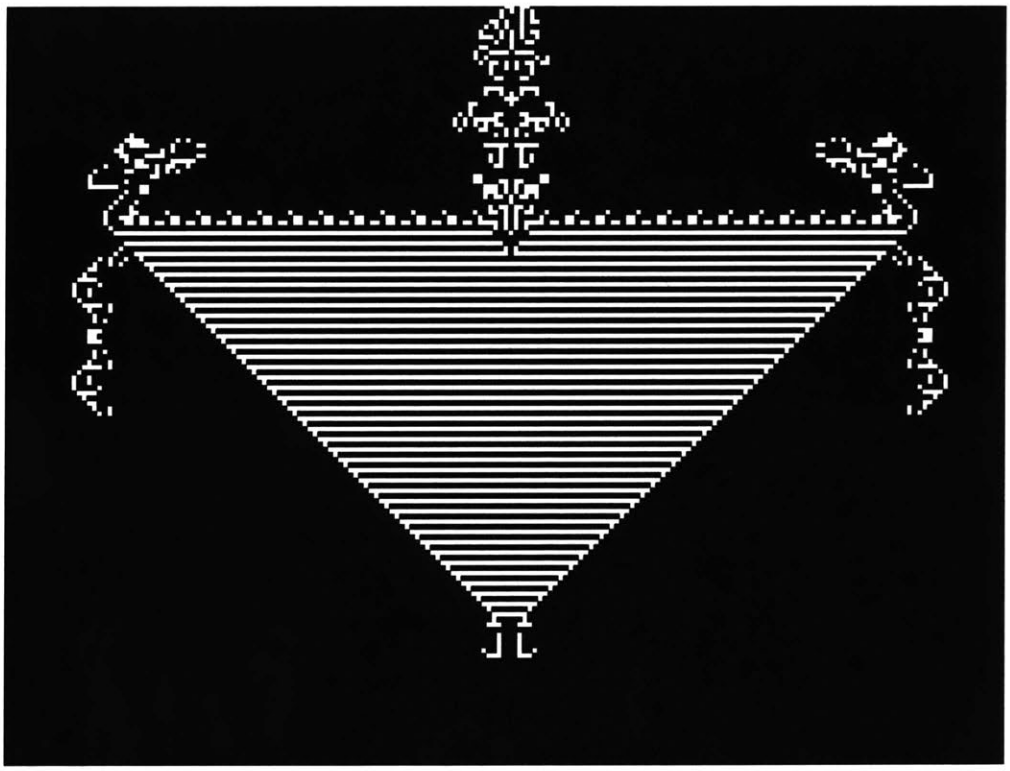


Figure 2-2: Game of Life: An example for a higher-order shape created by individual cells following local rules. Cells are solely motivated by their survival and are unaware of this orderly global shape they are generating.

Evidently, these rules are simplified reappropriations of the complex laws of population dynamics governing actual ecosystems in nature. It is precisely this abstract nature of Game of Life that makes it a great example for us to readily see how the five principles enabling nature to construct itself can generate genuine self-organization even in a rather insentient system. Here, similar to biological systems, the builders (i.e., cells) are identical (*principle I*), hierarchically equal (*principle II*), and unaware of the global layout; instead, they solely process local cues (i.e., the location of their immediate neighbors) using four simple rules of interaction (*principle III*). These rules are straightforward if-then statements (*principle IV*) allowing builders to determine their following move (i.e., whether they need to come alive, survive or die) in the next point in time. In this way, cells spatially compute their subsequent position (*principle V*) and thereby develop into higher-order spatial constructs.

Conway observed that the deployment of such a simple virtual system not only gives rise to complex spatial organizations, but also to a set of recurring patterns that consistently emerge irrespective of how the system was initialized. Upon surveying these convergent patterns, Conway annotated them as different "life forms," [Fig2-3] and categorized them based on their unique spatiotemporal behavior: *static and immobile, dynamic and immobile, dynamic and mobile* —reminiscent of naturalists' surveying of wildlife and cataloging them into various taxa based on their common traits.

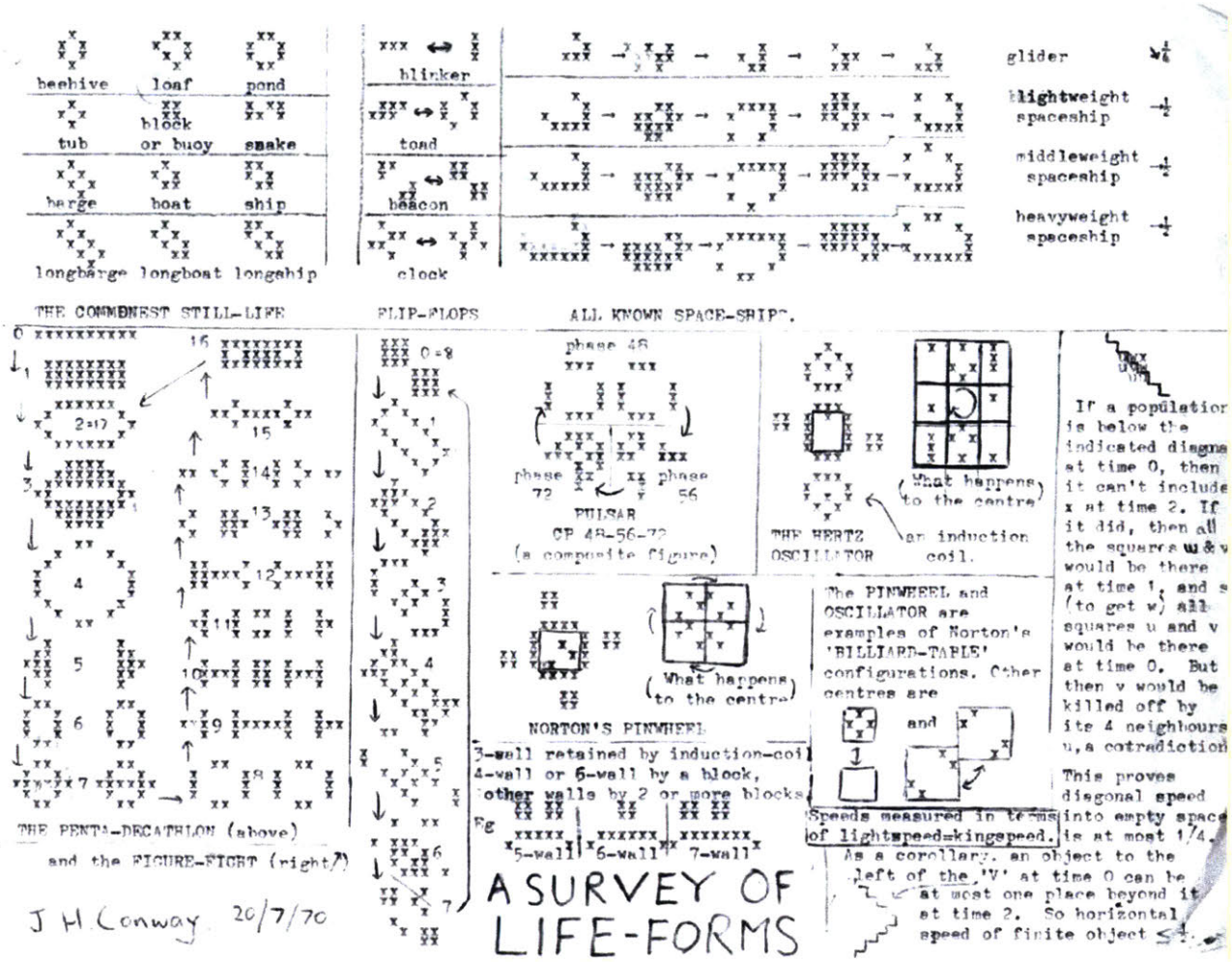


Figure 2-3: Game of Life: A sketch from Conway showing his discovery of different life forms within the universe of Game of Life. [13]

These life forms identified by Conway are analogous to the living organisms in nature —not only are they made of simple and repetitive building blocks, such as cells, and not only do they exhibit traits unique to their kind, but also, each life form seems to owe its characteristic shape to its *drive to reach the steady-state.* What generates momentum in biological systems is believed to be this constant search for steady-state, which is the state of *chemical equilibrium* where there is no net change. In other words, the search for steadiness appears to be the driving force behind the generation of orderly structures in nature; and a similar logic is at play in the spontaneous generation of spatial order in Game of Life.

As stated, life forms in the universe of Game of Life include both the static ones, who have already attained steady state, and the dynamic ones, who are in search of the steady state. The most primitive life forms among these are the *static and immobile*s, examples of which are the 'Beehive,''Block,' 'Boat,' and 'Loaf' [Fig2-4]. Each constituent cell in these life forms has exactly 2 or 3 neighbors; thus, per rule II, it can neither give birth, nor die, but simply survive. As a result, these static and immobile life forms exhibit no change in space and time; they are said to have reached the steady-state. In Game of Life, similar to what we observe in biological systems, once a life form arrives at the steady state, it loses its internal drive to transform itself into a more energetically favorable position; instead, it remains static and immobile.

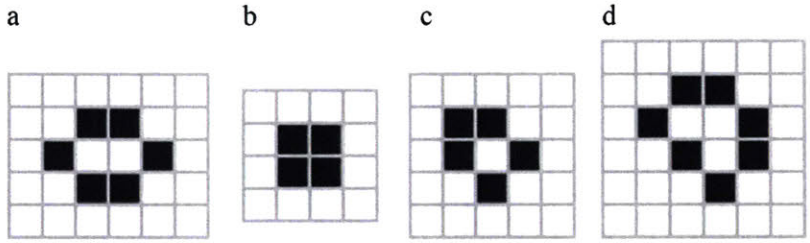


Figure 2-4: Game of Life: Examples of static and immobile life forms [13]. (a)"Beehive." (b)"Block." (c)"Boat." (d)"Loaf." These life forms do not change their spatial position across time since they have already reached the steady state.

Contrary to the static and immobile life forms above, the dynamic life forms of Game of Life always contain some cells that are in need of switching to a more "energetically favorable" state at the next point in time; hence, they constantly adjust their spatial position. Such cells include ones that had been blank but now are ready to come alive, since they are finally neighbored by exactly 3 other cells (per rule I); and some others that had been alive but now must die, since they are presently neighbored by more than 3 cells (per rule IV). As these cells are born and die, they morph the overall the life form, granting it dynamism. Examples for such dynamic life forms include both the *immobile* ones, which only change their spatial configuration in time and do not travel in space, namely 'Beacon,' 'Toad,' and

'Pulsar' [Fig2-5], as well as the *mobile* ones that not only change their spatial configuration in time but also travel in space, such as the 'Glider,' and 'Lightweight Spaceship' [Fig2-6].

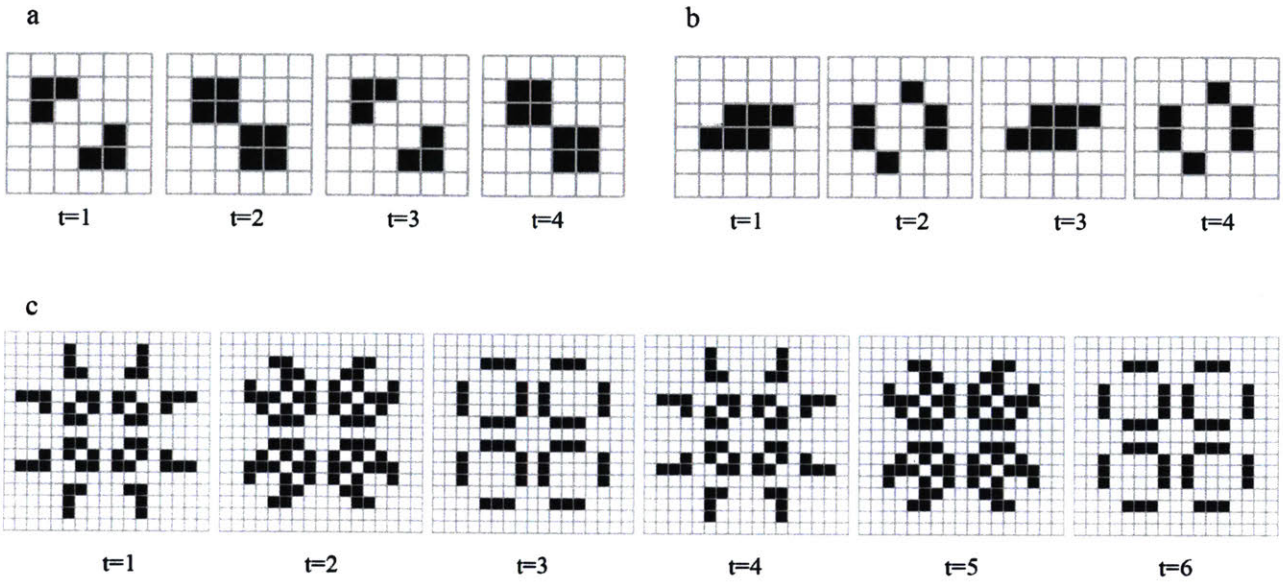


Figure 2-5: Game of Life: Examples of dynamic and immobile life forms across many time points [13]. (a)"Beacon." (b)"Toad." (c)"Pulsar." These life forms change their spatial configuration in time and do not travel in space.

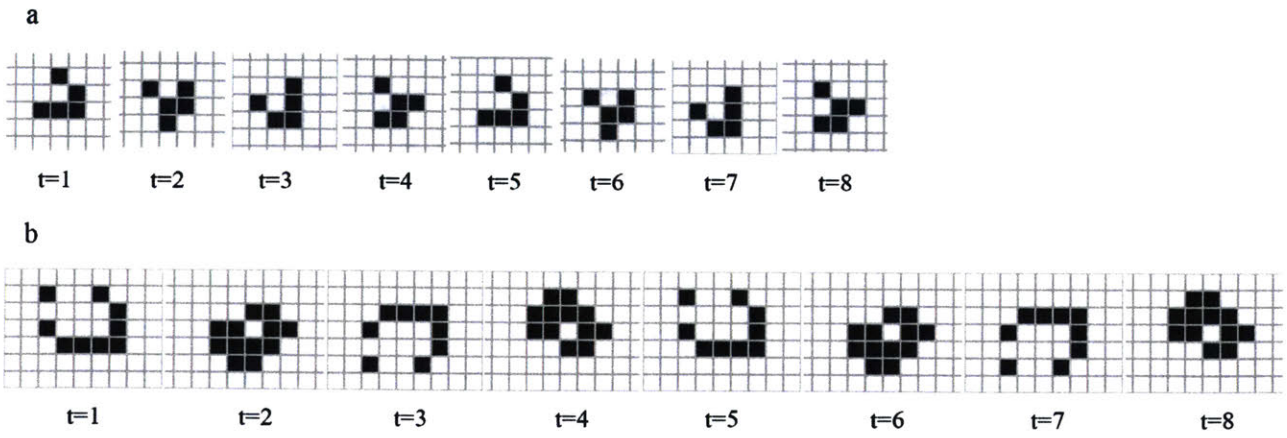


Figure 2-6: Game of Life: Examples of dynamic and mobile life forms across many time points [13]. (a)"Glider." (b)"Lightweight Spaceship." These life forms not only change their spatial configuration in time but also travel in space.

As the Game proceeds, interactions between these life forms begin to form even higher-order, vibrant societies, such as the "Puffer Train" [Fig2-7]. This dynamic, mobile and complex spatial organization emerges out of collisions between different life forms from all three taxa (including static and immobile Beehives, Blocks, Boats and Loafs; dynamic and immobile Beacons, Toads and Pulsars; dynamic and mobile Gliders and Lightweight Spaceships), reminiscent of nature's complex ecosystems composed of many different life forms.

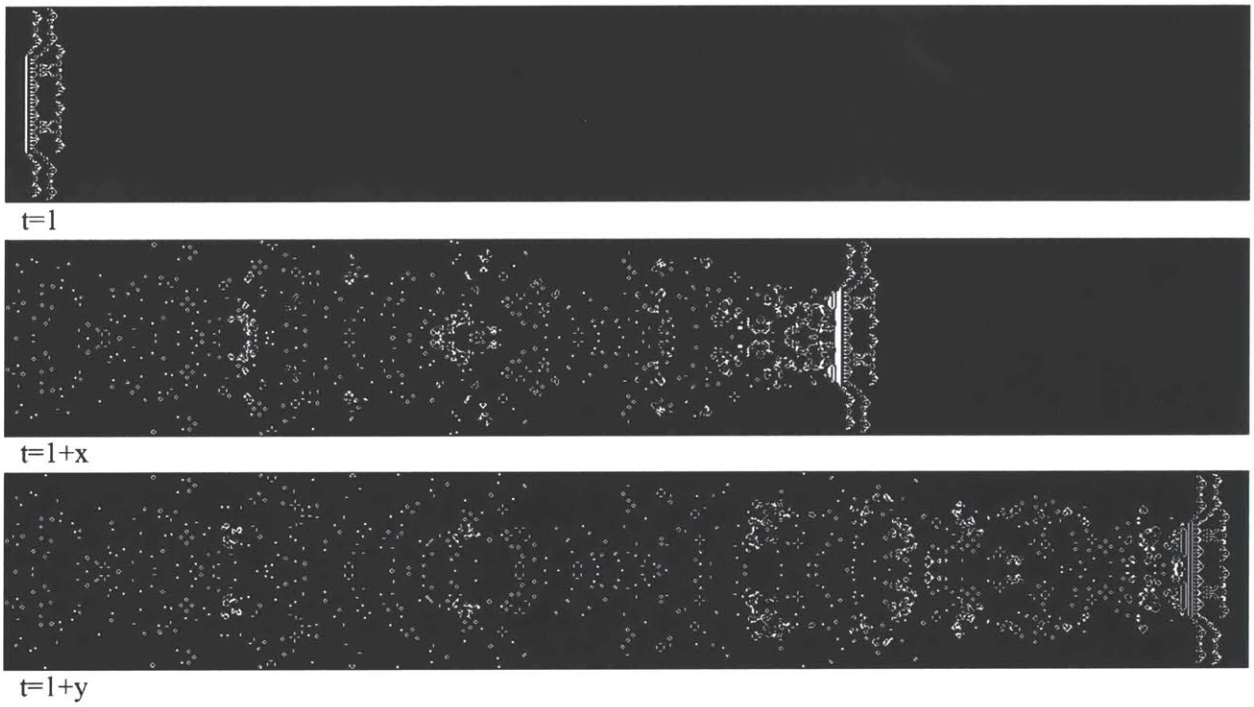


Figure 2-7: "Puffer Train," Example for a vibrant ecosystem in Game of Life

With these results, Game of Life shows that in a self-organizing system, even with extremely simple builders and straightforward rules by which they construct, highly complex spatial constructs can come about. In this way, Game of Life is a clear illustration of how *complexity in biological construction arises from the interactions between simple agents executing straightforward rules, rather than the sophistication of the individual agents.* One interesting implication of this axiom is that even if we kept the agents the same, changing

the rules by which they interact may result in immensely different spatial organizations. And this is exactly what we see when we look at other virtual systems where the the builders are still simple cells, but the rules by which they interact are less abstract; they are fine-tuned to generate spatial constructs that intimately resemble those occurring in nature. Examples of such systems include *Turing Patterns*, introduced by the English mathematician Alan Turing [14], and *L-Grammars*, introduced by the Hungarian Biologist Aristid Lindenmayer and Polish Computer Scientist Przemyslaw Prusinkiewicz [15][16].

These more complex self-organizing virtual systems generate spatial constructs that are more familiar to us than the abstract life forms of Game of Life. For example, a virtual system where cells are executing Turing's reaction-diffusion rules[5] [17][18][19] generate the labyrinthine spot and stripe patterns we observe on animals' skin [Fig2-8a]. Similarly, a virtual system where cells are executing Lindenmayer's grammars[6] [15] generate the mesmerizing branch bifurcation we see in plants [Fig2-8b].

[5]Rules governing a turing patterning system will be discussed in more depth in 3.2 and 4.2

[6]I will not be discussing L-Grammars in this thesis, for a further understanding, see bibliography.

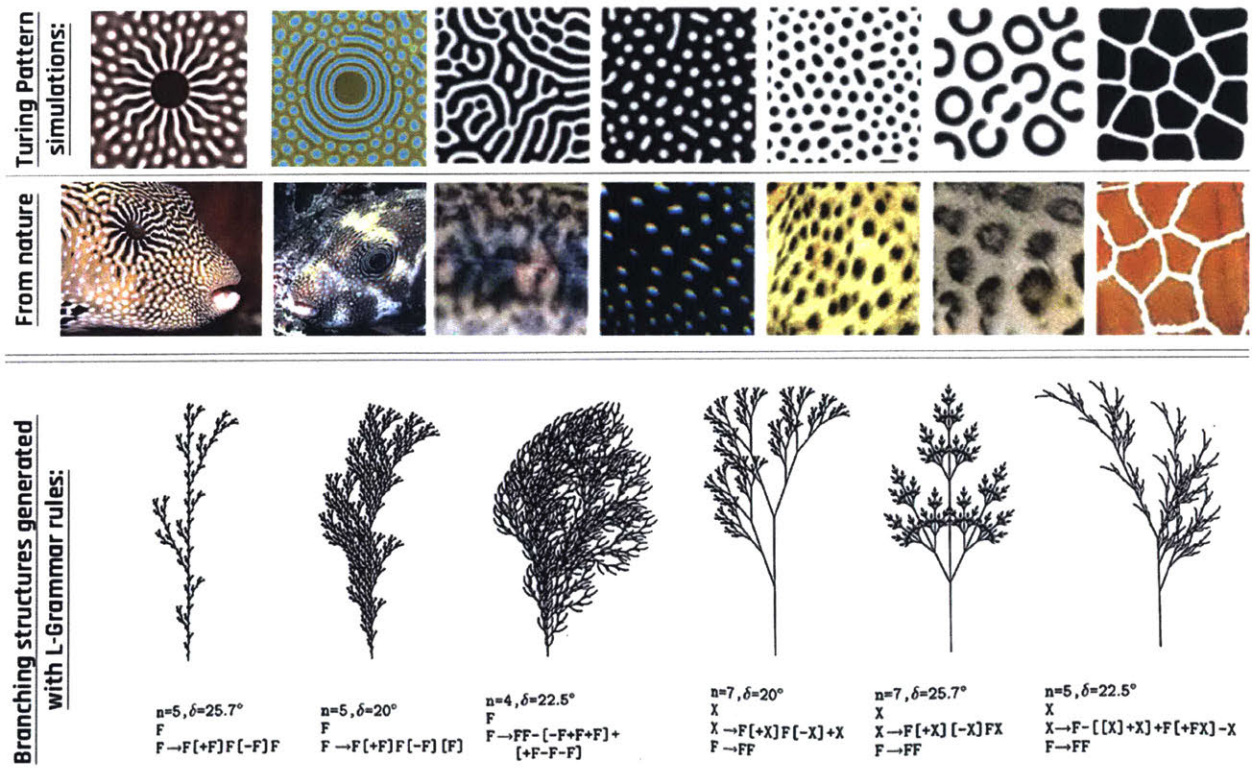


Figure 2-8: Turing patterns and L-grammars: (a)Turing pattern simulations generate stripe and spot patterns very similar to those found on animal skin, (b)as shown through the photographs. (c)L-grammar simulations: Elaborate branch structures similar to those observed in plants emerge when we employ Lindenmayer's recursive grammar rules.

It is important to emphasize that neither Turing Patterns nor L-Grammars necessarily elucidate the exact biochemical mechanisms by which such patterns on animals and plants are generated. Nevertheless, the power of these virtual systems comes from their clear demonstration of how a set of generative rules can build spatial constructs that are extremely similar to those that astound us in nature [Fig2-9], implying a rational, systematic, and even computational basis for natural pattern formation and structural development.

Figure 2-9: (a)Photograph of a fish with intricate patterns similar to those that can be virtually generated with Turing's reaction-diffusion rules. (b)Photograph of a tree with a bifurcating branch structure similar to those that can be generated with L-grammars. *Image credits: Poseidon Divers, Dahab, Egypt via D. Jain; unknown*

These virtual examples employ, in the more limited and primitive setting of a 2D plane, what earlier postulated to be the five principles of spatial self-organization in biological systems. These examples show that, even in this more abstract context, a group of short-sighted builders, who are unaware of the global scheme, can generate complex high-order spatial constructs by solely interacting with one another based on a set of local, generative rules —a designer, leader, or a blueprint is not needed.

To better understand how such a generative, bottom-up process can manifest itself in an actual biological context, in the next section, we will have a look at four case studies.

2.2 Case Studies: How Generative Rules Allow Nature's Self-organizing Builders to Construct Complex Structures Across Orders of Mangnitude

Understanding and appreciating the fundamental difference between how nature encodes and builds its evolved architectures using a *generative language*, versus how human architects and

engineers represent and build their purposeful designs using a *descriptive language* should be an absolute priority for any human designer intending to employ nature as a design medium. The main difference between these two approaches is that a generative language formulates the behavior of a system, whereas a descriptive language directly depicts what its final outcome should be. Accordingly, a generative language employs a more performative and process-focused syntax, such as if-then statements or state machine diagrams, whereas a descriptive language operates through rather static and product-focused representation, such as plans, sections, sketches, renderings, and blueprints.

I see one of the main roles of a human "biodesigner" to be a translator between these two languages, and thus I suggest that s/he first becomes fluent in both. Therefore, in the following sections of this chapter, I offer a deeper exploration of the generative language enabling nature to construct itself via four case studies across nature's different scales and contexts, from proteins to cities, insect nests to embryos. In each one of these systems, I focus on a different aspect of this generative construction process. With proteins and cities, I discuss i) how the if-then syntax of a generative rule works, ii) how such a rule arises in the first place based on builders' local motivations, and iii) how it can be used as a pragmatic tool by designers who are interested in getting these self-organizing builders to create artificial structures. Then, to illustrate this, with the insect nests and embryos, I introduce two scientific experiments that demonstrate how we can get wasps and embryonic cells to create artificial structures, such as double-headed mutants, by making rough "manual" alterations in these systems. It is later in Chapters 3 and 4, I show how more finely we can alter such systems by editing their generative rules and how in this way we can get their builders construct artificial structures *by design.*

Finally, the main objective of the case studies introduced in this Chapter 2 is to prepare the readers for the following chapters by showing *just how generative rules power the computational way in which nature constructs itself.* My goal with these case studies is not to convey all

there is about self-organization in proteins, cities, insect nests and embryos; but rather, to use these four systems to demonstrate how nature operates through a generative language, rather than a descriptive one; and most importantly, how this can be useful to us architects in developing a novel fabrication method.

2.2.1 Amino Acids Execute a Set of Generative Rules to Build Proteins

Amino acids execute a set of generative rules to process the local cues, and thereby, self-organize into the higher-order structures of proteins. These local cues are the four main physical forces acting on amino acids in solution: electrostatic forces, Hydrogen bonds, Sulfur-Sulfur interactions, and Van der Waals forces [20]. Based on these forces, autonomous construction of a protein's complex three dimensional structure occurs in several consecutive stages[7], throughout each of which these four forces impact amino acids at different intensities. Hence, the same set of amino acids at different stages of construction employ different generative rules to get to the next stage. In other words, the dominant physical force in each stage of construction determines which generative rules amino acids must use to get the overall structure into the next stage of construction. For example, while transitioning from stage one to stage two, the generative rules amino acids use are informed by the physical force that is dominant in stage one: Hydrogen bonding (H-bonding) force.

All amino acids in stage one, which are as yet only connected to one another in a simple, chain-like structure called the *primary structure of a protein*, contain atoms that want to to form H-bonds with similar atoms in the neighboring amino acids, causing this chain to either fold onto itself and form a *helix* structure, or to combine with other chain-like primary structures and form a *sheet* structure. These helix or sheet assemblies are called the *secondary structure of a protein*. Below, let us have a look at the generative rules governing this process

[7] In three stages for most amino acids, in four stages for some.

in their if-then syntax:

Primary structure of a protein is made up of individual aminoacids (each shown with a dashed box on Fig2-10) that are linked to one another via peptide bonds. In this long "polypeptide chain":

> ***if*** *the "carbonyl" Oxygen atoms (shown in red) and the "amino" H atoms*[8] *(shown in gray) of all amino acids are aligned on opposite sides of the chain (i.e., if they are in "cis position" with respect to the peptide bonds connecting them) [Fig2-10],*
>
> > ***then*** *the polypeptide chain will wind around itself and form a helical structure, called an "alpha helix" [Fig2-11].*

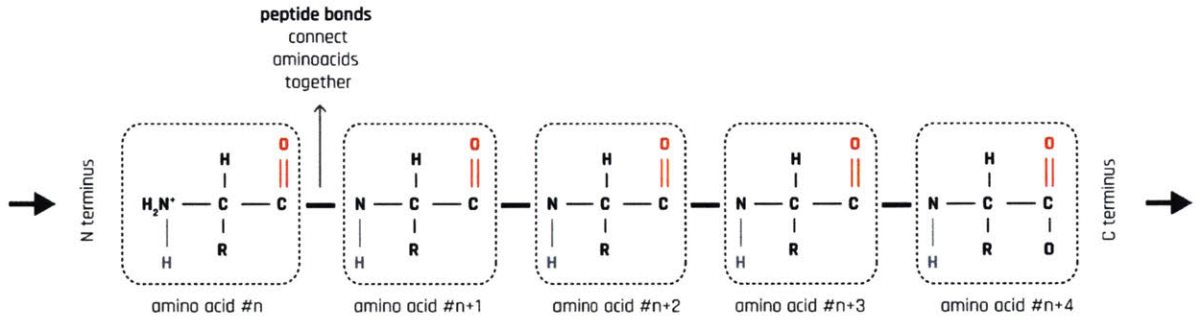


Figure 2-10: Primary structure of a protein, where the individual amino acids are shown with a dashed box. Each amino acid's "carbonyl" Oxygen atom is marked in red and the "amino" H atom is marked in gray. These O and H atoms align on the opposing sides of the chain.

[8]These O and H atoms are the H-bond forming atoms, whose orientation determines into which secondary structure these amino acids will self-organize. However, it is important to note that this process is in fact more complex: what determines the orientation of these O and H atoms are the each amino acid's "phi" and "psi" angles, which are determined by its functional group (R) constituents. The orientation of the H-bond forming atoms O and H discussed inline results from this complex web of physical forces.

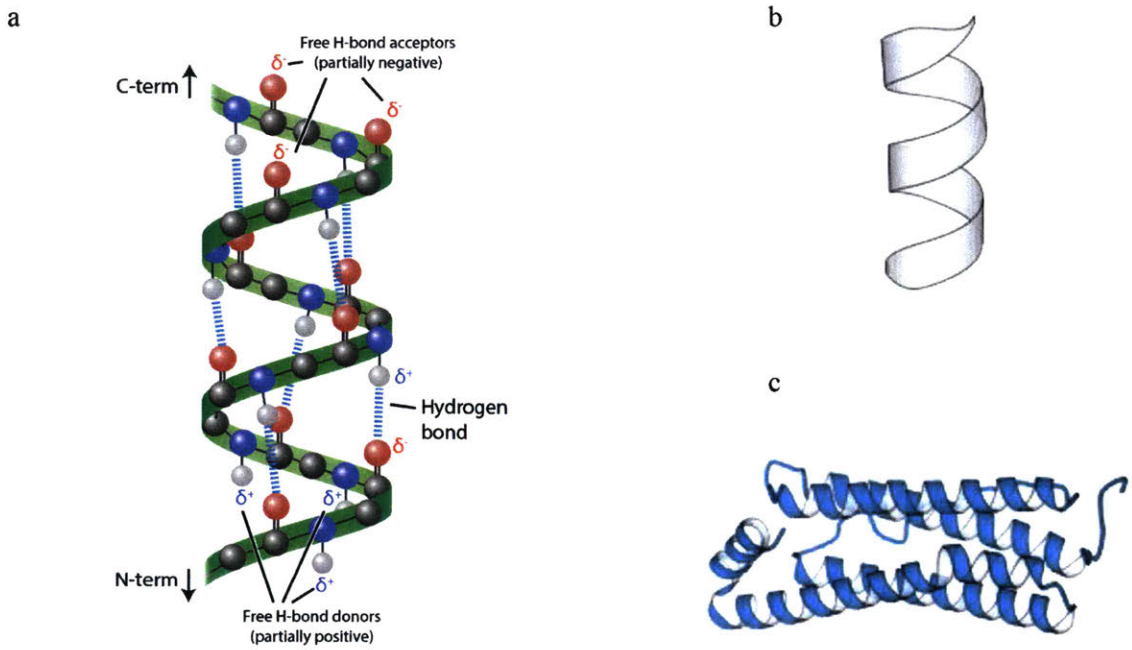


Figure 2-11: (a)A protein secondary structure called "alpha helix." The same carbony Oxygen and amino H atoms we saw in the primary structure are still marked in red and gray, respectively. Dashed lines show them participating in H-bonding with each other, which is what enables this chain to self-organize into such a higher-order helical structure. (b)Conventional "ribbon" depiction of an alpha helix. (c)Example of an actual protein (Ferritin, an iron-storage protein) that is entirely made up of alpha helices, also shown in ribbon representation.

> ***else**, if the carbonyl Oxygen atoms (shown in red) and the amino H atoms (shown in gray) of all amino acids are on altering sides of the chain (i.e., if they are in "trans position" with respect to the peptide bonds connecting them) [Fig2-12],*

> ***then** the polypeptide chain will start laterally linking to other similar chains to form a sheet-like structure, called a "beta sheet" [Fig2-13].*

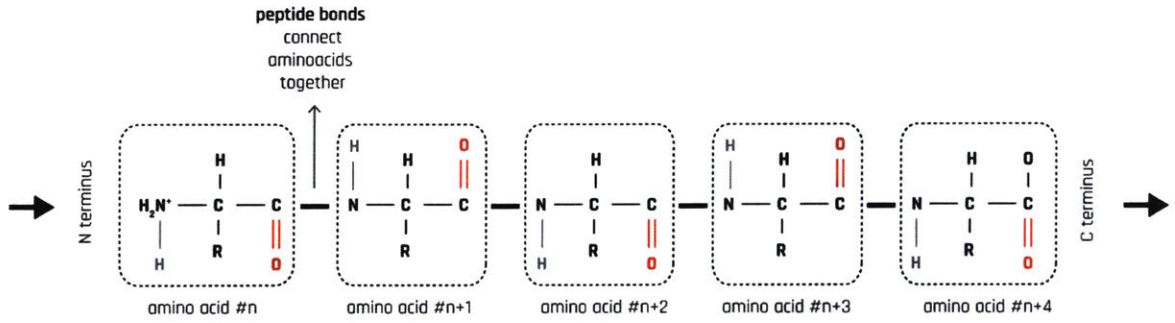


Figure 2-12: Primary structure of a protein, where the individual amino acids are shown with a dashed box. Each amino acid's carbonyl Oxygen atom is marked in red and the amino H atom is marked in gray. Notice that these O and H atoms are on the altering sides of the chain.

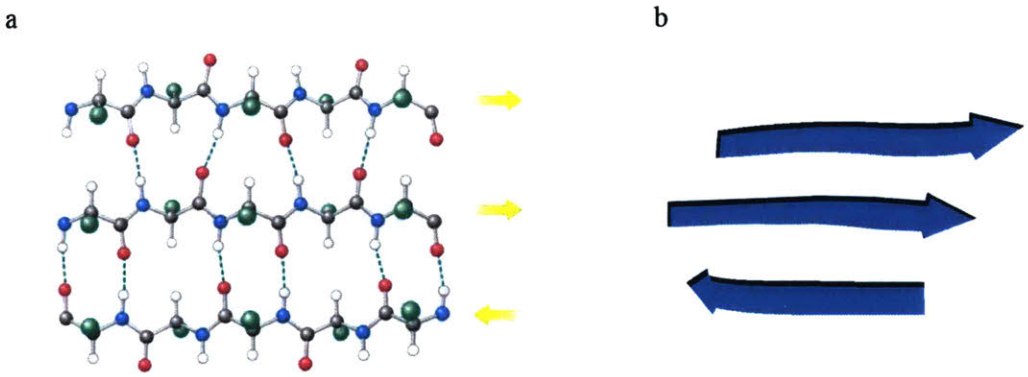


Figure 2-13: (a)H bonds (shown with blue dashed lines) are formed between the carbony Oxygen atoms and the amino H atoms (marked in red and gray, respectively) of different polypeptide chains, bring about a protein secondary structure known as the "beta sheet." (b)The same beta sheet shown in the conventional "ribbon" depiction.

In these beta sheets:

if *the H bonds between the carbonyl Oxygen atoms (still shown in red) and the amino H atoms (still shown in gray) are forming at right angles [Fig2-14a],*

then *the beta-sheet will have an "anti-parallel" pattern [Fig2-14b].*

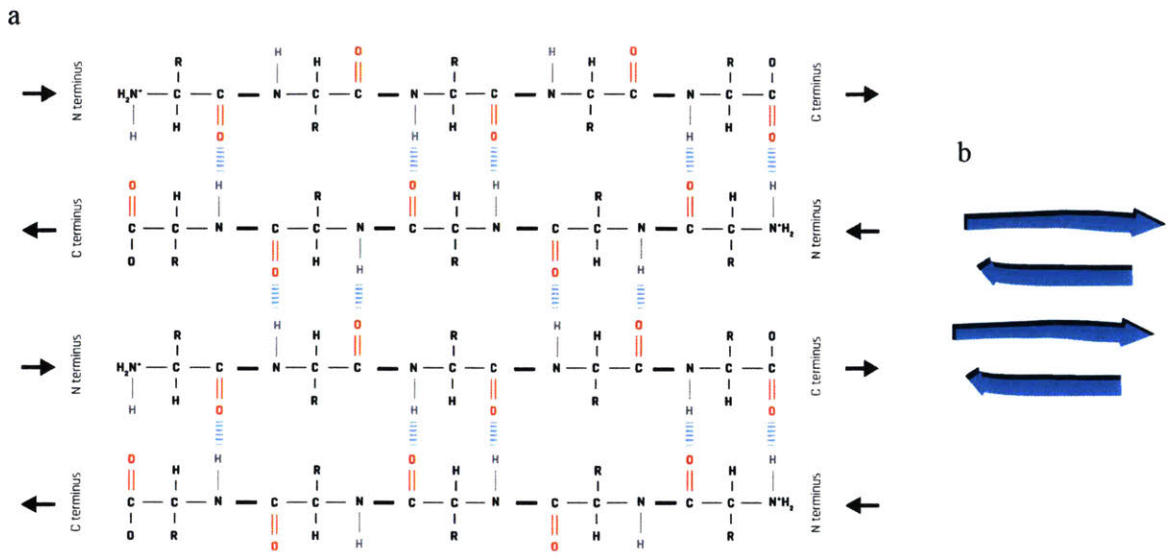


Figure 2-14: (a)An anti-parallel beta sheet secondary structure. Blue dashed lines indicate the H bonds forming between the carbonyl Oxygen atoms (in red) and the amino H atoms (in gray). Because these H bonds form at right angles, the chains link to one another in an anti-parallel fashion (each chain has a directionality from its "N terminus" to "C terminus"). (b)Conventional "ribbon" representation of the same anti-parallel beta sheet. Notice that the arrows are showing the N to C termini directionality of each chain.

else *the H bonds between the carbonyl Oxygen atoms (in red) and the amino H atoms (in gray) are forming at 45 degree angles [Fig2-15a],*

then *the beta-sheet will have a "parallel" pattern [Fig2-15b].*

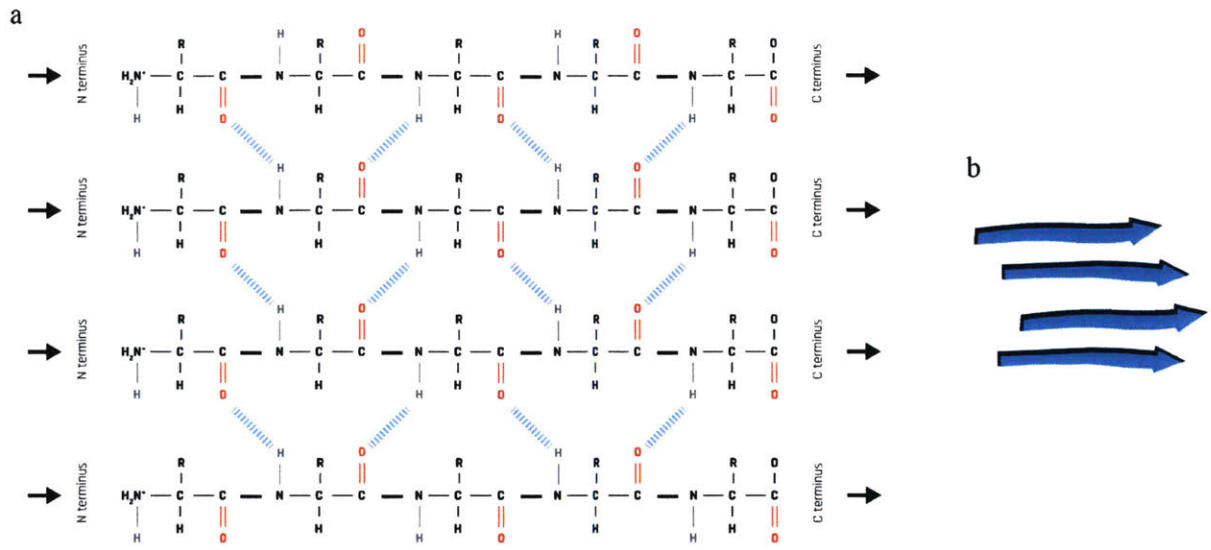


Figure 2-15: (a)A parallel beta sheet secondary structure. Blue dashed lines indicate the H bonds forming between the carbonyl Oxygen atoms (in red) and the amino H atoms (in gray). Because these H bonds form at 45 degree angles, the chains link to one another in a parallel fashion (each chain has a directionality from its "N terminus" to "C terminus"). (b)Conventional "ribbon" representation of the same parallel beta sheet. Notice that the arrows are showing the N to C termini directionality of each chain.

In this way, generative rules specific to stage one enable amino acids to process local cues present at this stage (i.e., Hydrogen bonding forces), and thereby spatially compute whether this protein's secondary structure should be an alpha helix, or a parallel / anti-parallel beta sheet. These secondary structures can further interact, this time using a different set of generative rules, and form the even higher-order tertiary structure of a protein [Fig2-16].

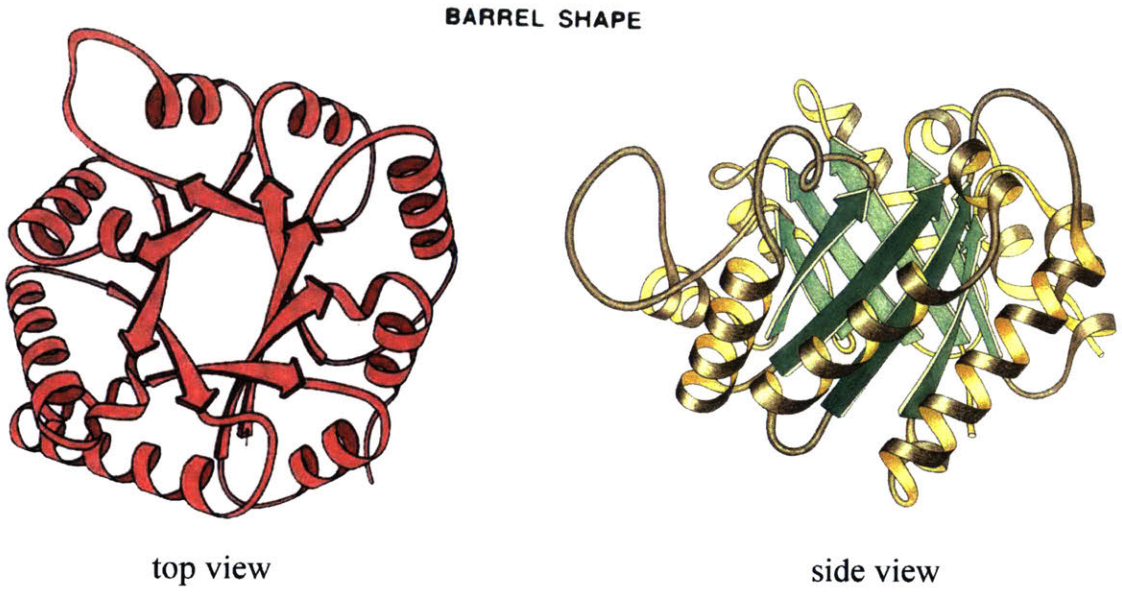


Figure 2-16: The final 3D structure of the protein *Triose Phosphate Isomerase*, which has a pivotal role in our glucose metabolism, is composed of beta sheets and alpha helices forming a barrel-like shape.

One important thing to note about these if-then statements is that they are high-level descriptions of *what* the builders do, and not necessarily *how* they reason about doing it. Amino acids are rather passive entities; they lack the agency to reason about creating higher order structures; instead, they are solely concerned about meeting their immediate needs, which is to arrive at a more "energetically favorable" state. To do this, the atoms making up an amino acid constantly work towards making the most of the immediate forces exerted on them; they are completely oblivious to anything beyond this local interest, such as building higher order molecules shaped like helices [Fig2-16].

Therefore, looking back to the generative rule introduced above, we can appreciate that, in fact, an individual carbonyl Oxygen atom within an amino acid does not, nor can it, register whether or not the other carbonyl Oxygen atoms are on the same side of the chain as itself —let alone accordingly assist the chain to form a helix shape. Instead, the only effort a

carbonyl Oxygen expends is to calculate what would be the most convenient position for it to avoid collisions with other atoms in this polypeptide chain, so that it can stay stable. Yet, this is exactly the most astonishing point about the way nature constructs itself: such pure local responsiveness not only creates a higher-level structure with an orderly shape, such as a helix, but it also creates a *formulaic pattern* by which this structure is generated, *a generative rule.* In other words, generative rules are not exact descriptions of how builders locally reason about creating global orderly structures; instead, they are higher-level abstractions from our global point of view of that reasoning. Yet another way to understand this subtle but important point regarding the essence of generative rules is to look at self-organizing construct built by humans, such as an indigenous city.

2.2.2 Humans Execute a Set of Generative Rules to Build Cities

As humans, we like to think of ourselves as distinct from the rest of the nature since we are self-aware creatures with a capacity to be cognizant of the bigger picture. Our desire and ability to think at the global level has had major effects on how we live, and especially how we construct. Our cities are shaped more and more by our ever-developing top-down planning tools, spanning from the ancients' hand-drawn city maps to today's satellite-enabled technologies, such as the Global Positioning System (GPS) or the Geographical Information System (GIS). Moreover, development of human civilization brought about specialization among individuals in a society, resulting in development of professionals with expert knowledge on the construction of dwellings and cities, such as architects, civil engineers and city planners.

However, there exists cities built by humans who had none of these global planning tools or any specialized knowledge at hand. These cities were built either at a time when such knowledge and tools did not exist, such as a neolithic settlement, or in a place where they were not readily accessible, such as a slum. Hence, humans building these cities had no reference point other than local cues and their immediate needs. These needs often included

having a living unit that is i) sheltered from environmental sources of discomfort, such as the weather, noise, or predators; ii) private, yet still connected to the other units in the settlement for the purposes of efficient sharing of resources, collective defense against external threats, or social interaction; and iii) able to make the maximal use of the available land and building materials to construct with. This is too much to account for all at once for a group of people who have limited resources and no access to top-down, rational planning tools. Nevertheless, by locally reasoning about these needs in the following bottom-up way, these human groups were able to create settlements that not only met the indicated needs[9], but also brought about an orderly global layout such as the "beady ring" layout [21].

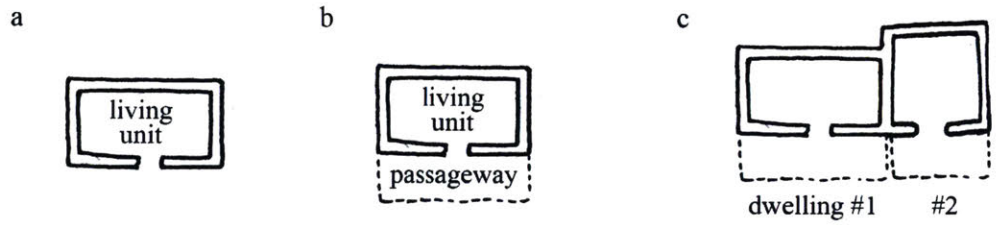


Figure 2-17

1 Shape of the each living unit had to be a rectangle, so that the inhabitants could fit the most number of dwellings into their limited land[10] [Fig2-17a].

2 Despite the need for dense space usage, each unit had to reserve an equally sized passageway facing its entrance. Passageways of two separate units would rarely overlap to ensure an unobstructed access to each dwelling [Fig2-17b].

[9]It is important to keep in mind that each land imposes different constraints on their inhabitants, hence not all builders of self-organizing cities have the same needs. Needs presented here apply generally to the environments where the main limitation is the availability of the land to construct on such neolithic cities, where dwellings often had to be constructed underground for better protection from wild animals; some medieval villages, especially ones that had to fit within city walls or were constructed on a narrow hilltop; or, slums, which have only so much space they can reclaim before they encounter gated communities.

[10]In places where the land is not the bottleneck, we see formation of circular or elliptical dwelling units, for example in Moundang compound in Cameroon.

3 Each new dwelling could articulate its rectangular living unit into an existing one only face-wise, so that there would be one less wall to construct and hence building material would be saved. Living units rarely articulated to one another vertex-to-vertex because this would cause an unnecessary waste of land [Fig2-17c].

4 Each new living unit articulated its new passageway into an existing one, so that a continuous circulation path could form, connecting all the dwellings in the settlement to one another [Fig2-17c].

How these local considerations can give rise to a functioning city layout can be seen in the neolithic settlement *Hacilar*, which dates back to 7040 BC and is found in present day Turkey [21] [Fig2-18].

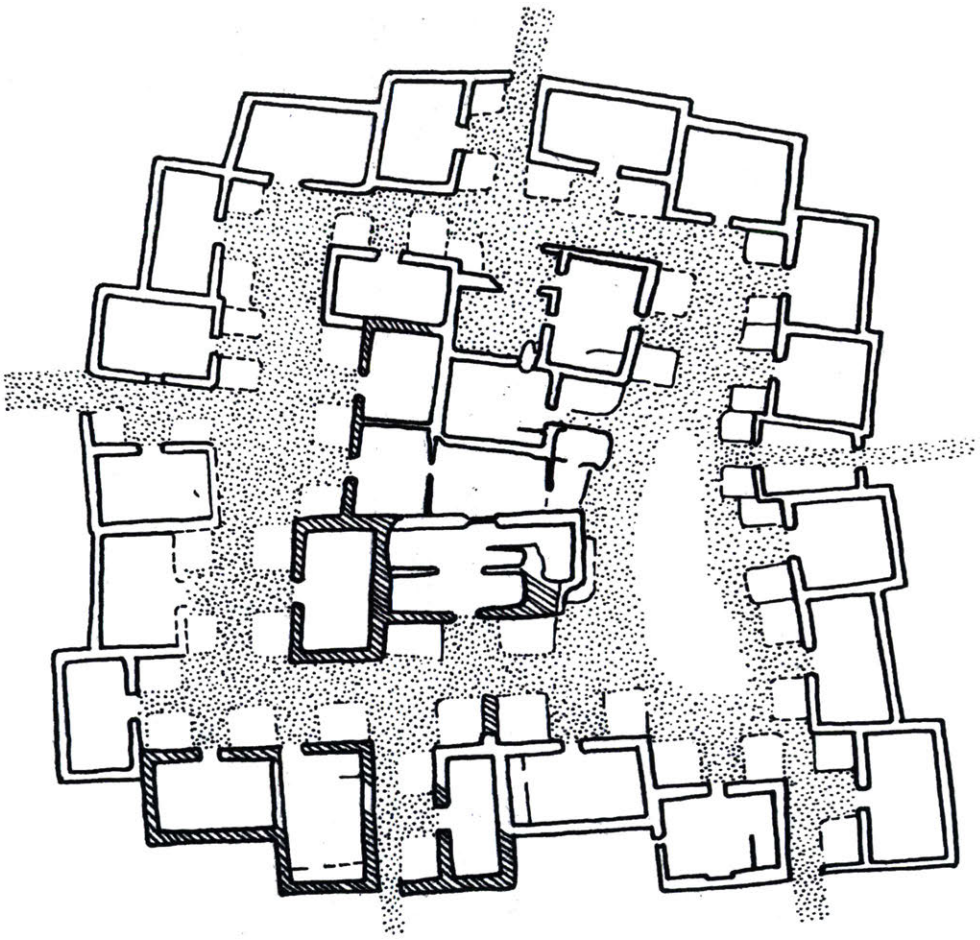


Figure 2-18: Neolithic settlement *Hacilar*: An example of a self-organized city (only a small portion of it shown here) with the beady ring layout.

Below, let us have a closer look at how such an orderly and functional settlement can grow in a self-organizing manner, only based on the local considerations stated above. In this next example, each dwelling is shown with an abstract representation: each covered living unit is indicated with a filled square, marked with a dash on its one side representing its entrance [Fig2-19]. The entrance connects the unit to its passageway, which is shown with a cross to represent its emptiness. In this way, one covered living unit and its passageway make up a dwelling; dwellings come together and form a settlement.

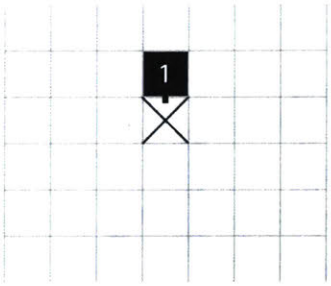


Figure 2-19: First dwelling of our hypothetical settlement, shown in an abstract representation.

Below, by using the four major local considerations, or rules of growth, explained earlier, we can see how a second dwelling could articulate to this arbitrarily placed first one:

If the builder of the dwelling #2 is interested in saving material over having full privacy,

then, it can share a wall with the existing dwelling #1, in which case, its covered living unit has to be adjacent to the covered unit of the dwelling #1. Furthermore, since each new passageway has to articulate into an existing passageway, the new dwelling can only assume one of the two positions:

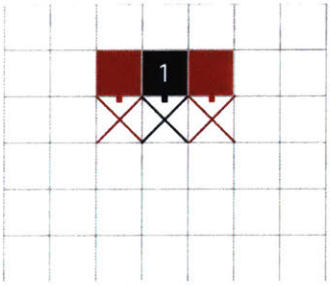


Figure 2-20: Two of the five plausible positions that the second dwelling can assume, per local rules of growth ,based on the first dwelling's position.

Else, if the settlement #2 prefers more privacy over saving material, then it does not have to share a wall with the existing dwelling #1 at all. In this case, the dwelling #2 can assume only one of the three positions below, because its living unit cannot connect vertex-to-vertex to the living unit of #1 and its passageway still has to articulate face-wise into an existing passageway:

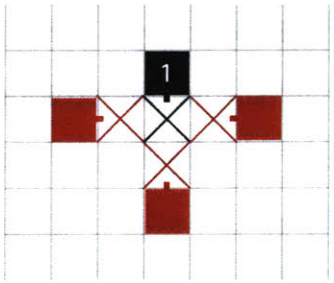


Figure 2-21: Other three of the five plausible positions that the second dwelling can assume, per local rules of growth ,based on the first dwelling's position.

Once dwelling #2 picks one of the red marked slots, the process starts again to determine the location of dwelling #3. In a settlement growing in this way, each dwelling can have many different fates, each of which is contingent on the choices made by the previous ones. If we continue this process until we have 24 dwellings in our settlement, the below "beady ring" layout emerges (the order in which settlements added is shown with increasing numbers) [21]:

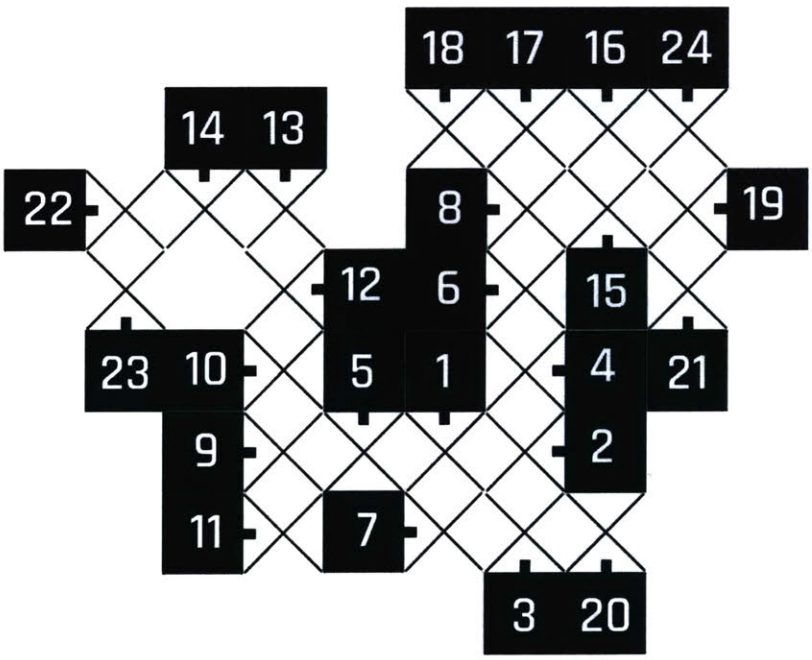


Figure 2-22: Emerging beady ring settlement layout.

In order to see more easily how such a local building logic could grow into a global city layout, we can simulate this process in a computer by randomizing the decision made by each builder regarding whether to save building material, or to have more privacy. In fact, for the purposes of a computer simulation, such local human motivations that gave rise to this logic in the first place can be further abstracted as a simple if-then-based generative rule shown below. After all, the computer running this simulation does not have to know *why* humans developed such a construction logic; local concerns regarding bodily comfort, social interaction and material economy are irrelevant to a computer. The computer only needs to know *what* it is that humans are doing when constructing a self-organizing settlement with a beady ring layout, which can be summarized as:

function ("select next dwelling"):
<random toggle (m , p)>[11]
if *m*
    then *mirror about the longitudinal axis on <random toggle (left, right)>*
    select next dwelling
else
    rotate <random select (90, 180, 270)> degrees about the cross midpoint
    ofset 1 slot outwards
    select next dwelling

Based on this generative ruleset, if we let the computer grow us a city, we see it proposing the layout on Fig.2-23a [22]. Striking similarity of this layout into an actual settlement built by humans [Fig2-23b] shows the power of generative rules in creating self-organizing structures *de novo*. It is enough for a computer to know the abstract generative rule by

[11] "m" for material, "p" for privacy

which humans construct; it is only concerned with *what* the builders do; *how* or *why* they are doing this is irrelevant.

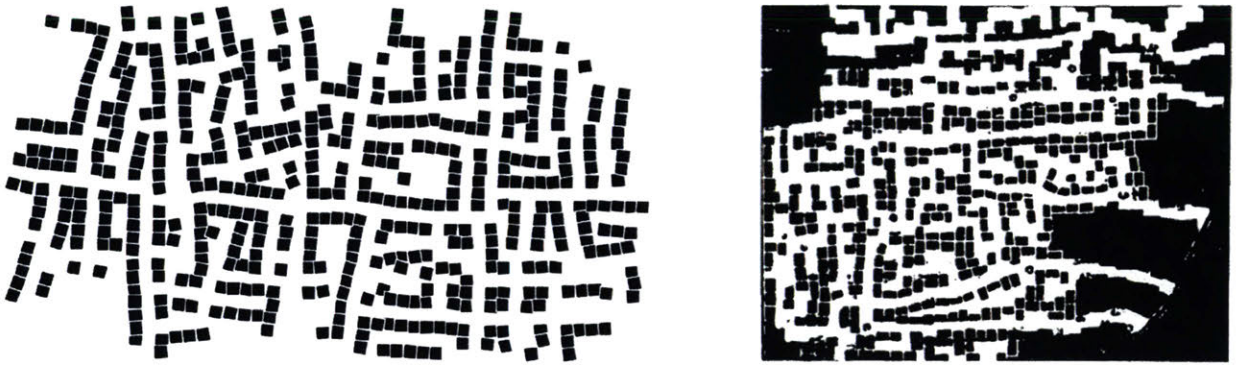


Figure 2-23: (a)Computer simulation of a beady ring layout. (a)An actual human-built city in the *Jhuggi* district in Delhi, India.

When we look at this process from the builders' local perspective [Fig2-24], neither the "beady ring" layout, nor the rule generating it, is visible. At this low-level, the only thing that matters is whether the highest possible number of dwellings could fit into this limited field; whether each dwelling has a clear passageway and an unobstructed access to the field-wide circulation path; and finally, whether the material used in this process was minimized.

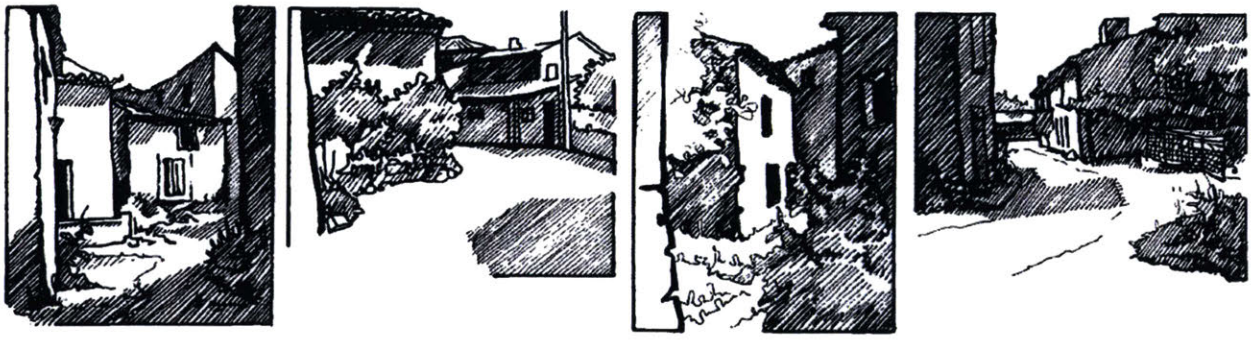


Figure 2-24: Street level sketches from the medieval villages of France, Vaucluse district, yet another self-organized city with the beady ring layout

Just as neither the helix shape, nor the if-then statement by which it was created is visible to a carbonyl atom within a protein, neither the beady ring layout, nor the generative rule

by which it was created is evident to the human builders and occupants of these cities. Yet, the fact that humans do not explicitly use this generative rule to build such cities does not invalidate this rule's capacity for generating them, as was shown by the computer simulation [Fig2-23].

Using generative rules in this way provides us a practical means by which to recreate self-organizing structures in-vitro. More specifically, it constitutes a rather top-down method for designers to *communicate with* nature's builders and get them to construct structures by design. Hence, even if these builders operate entirely bottom up, the tool of generative rules creates a middle ground. The emerging biodesigner does not need to learn the generative language of nature so that s/he can entirely abandon her/his top-down approach; but rather so that s/he can develop a new design heuristic that integrates this approach with the bottom-up way in which nature builds. Using generative rules in this way constitutes such a design heuristic.

The next case study, on wasp's nests, demonstrates how such a pragmatic understanding of the generative rules governing a self-organizing system can be useful for a human scientist to get wasps construct him *mutant structures on demand.*

2.2.3 Wasps Execute a Set of Generative Rules to Build Their Nests

Wasps construct funnel-shaped structures, composed of "bricks" made of mud, to serve as a gate to their nest, which is a narrow, deep hole in the ground [Fig2-25].

Figure 2-25: (a)Cross section of a wasp's nest. (b)The nest belongs to the potter wasp species *Paralastor sp.*

Each wasp is "hardwired" with a building program that consists of a set of generative rules, each rule guiding the construction of a different part of this funnel-shaped structure [23] [Fig2-26]. For instance, rule set 1 (R1) is used to build the stem of the funnel; R2 for the curved neck; R3, R4 and R5 for different sections of the bell. Just as amino acids use different rule sets in different construction stages of a protein, wasps use different rule sets in different construction stages of their nest.

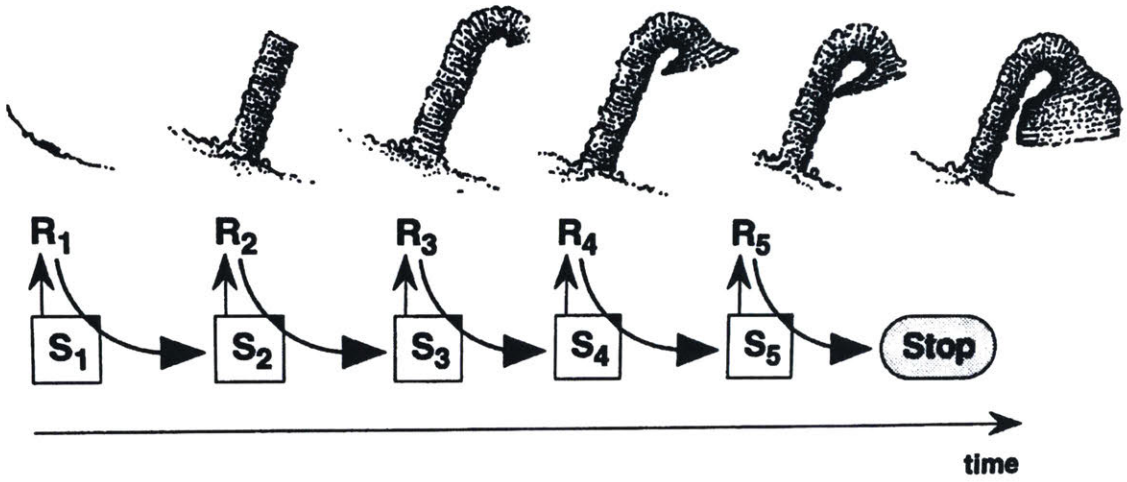


Figure 2-26: Building algorithm of a wasp's nest, made up of many generative rules

Yet, if wasps do not have a shared mental blueprint of the overall structure, how do they know which rule set to employ at different stages of the construction? In fact, how do they even decide which part of the funnel-gate they are supposed to be constructing at a given time? For this purpose, like any other builder of a self-organizing structure, a wasp, too, solely relies on local cues for this purpose. For wasps, these local cues are the geometric configurations emerging at the front line of the structure at different stages of the construction process. Each local configuration strikes the wasps as a unique stimulus (i.e., S1, S2, S3...), which then enables them to employ the respective rule set leading the construction of the next stage [Fig2-26]. For example, a circular hole strikes the wasps as the start stimulus (S1), which then enables them to employ the R1 rule set leading to the construction of a rim encircling the cavity [Fig2-26] [Fig2-27].

Figure 2-27: A wasp constructing the rim encircling the cavity on the ground.

As this rim rises and reaches to a 3cm-tall stem, the newly emerging local geometric configuration (S2) then stimulates wasps to commence with rule set R2. In this second stage of the construction, the R2 rule set guides the construction of the curved neck [Fig2-28], the completion of which then generates the local geometric configuration (S3) that stimulates wasps to begin R3. In this way, completion of one stage provides the stimulus for wasps to switch to a new rule set, which then guides the construction of the consecutive stage [Fig2-26]. This sequential building activity goes on until the bell-like opening of the funnel approaches

completion, at which point the emerging local geometric configuration finally signals wasps to stop.

Figure 2-28: A wasp constructing the neck of the nest using the generative rule R2.

Zoologist Andrew Smith, having observed how wasps organize mud bricks in this way when constructing their funnel-gate, wanted to test whether the wasps simply obey a mental blueprint, or solely rely on local stimuli [24]. For this purpose, he performed a set of controlled experiments with a wasp colony in their wild environment. In one of these experiments, Smith approached an already completed funnel-gate and on top of it carved a circular cavity, one that is very similar in its local geometric configuration to S1 [Fig2-29].

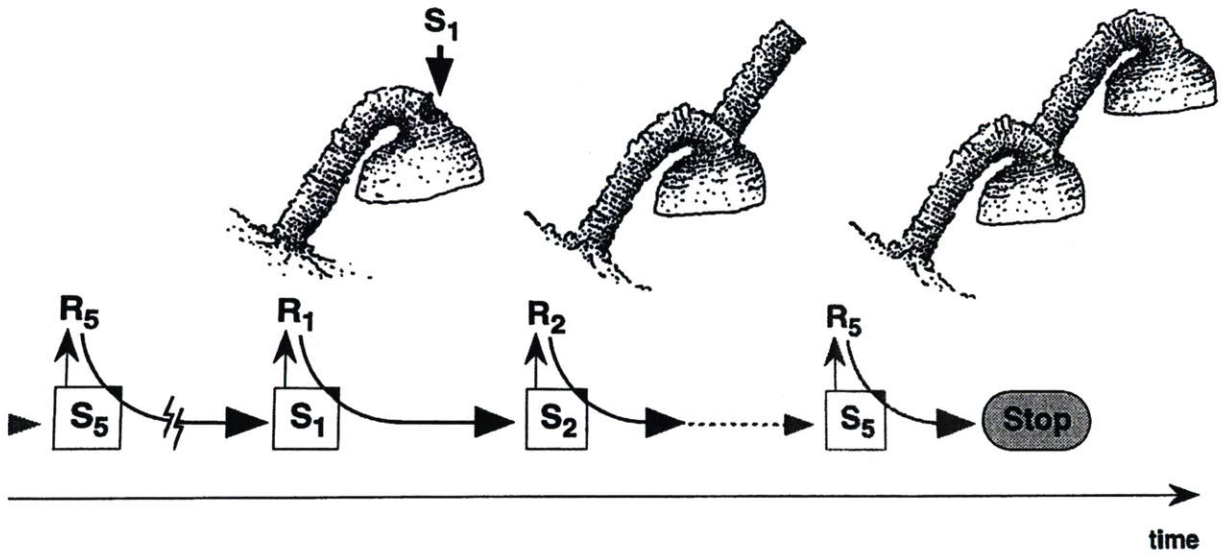


Figure 2-29: Smith's experimental layout and the result

Smith's hypothesis was that, if a wasp constructs in a top-down fashion with a descriptive mental blueprint of its nest, then it would see this cavity as damage to its nest and repair it, reverting the structure back to its original state. On the other hand, if wasps construct in a bottom-up fashion by solely relying on local stimuli, they would instead take this circular cavity as a start signal, due to its local resemblance to S1, and construct a second nest on top of the already completed one!

Smith's experiment repeatedly resulted in wasps' constructing a mutant, double-headed gate. This result proved[12] which proved that wasps do not rely on a pre-made blueprint but rather generate spatial order by taking in, processing and producing information in the form of the local organization of the building blocks (mud bricks). In doing so, wasps do not require any central governance, rational design, or top-down inspection; the structure they build is pure embodiment of distributed intelligence.

[12]See the associated negative controls of this experiment in the same paper, where when the damage had no particular resemblance to a local stimuli, wasps did repair it properly.

To get wasps to build such mutant structures, we do not need to know why these particular local geometric organizations stimulate the wasps the way they do; we just need to know what they are and what is their effect on wasps (i.e., which part of the structure they stimulate a wasp to construct). In other words, what the if-then based generative rules are in this system. Smith knew that S1 was a circular hole on the ground and it would stimulate a wasp to start the construction of a new nest. This was enough for him to get wasps systematically build him a mutant structure. Smith did not need to know *why* wasps take such a circular cavity as a start stimulus, which is that the resulting gate acts as a parasite deterrent protecting the caterpillar stock found in a wasp's nest [Fig2-30]. In this way, this case study yet again shows the fundamental role of generative rules in governing a self-organized construction process.

Figure 2-30: A wasp carrying a caterpillar into its nest, which is under construction.

How a wasp systematically constructs its nest in this way is a prime example of the computational way in which nature constructs itself. Here, the rules of computation are executed by, and embodied in, the builders of the space —the wasps— and the information wasps compute with —local cues— is transmitted through the space via the local configuration of the mud bricks. Accordingly, each time a wasp "executes computation," it generates information by articulating a new set of building blocks into the existing structure. The result is a series of informational statements forming a continuous, inhabitable structure.

Still, this structure itself does not have any computational capacity. Unlike its builders, which are living computational agents, the structure is made of inert building blocks that are unable to process information, sense and respond to their environment, or self-replicate. Therefore, despite the structure encoding all the spatial information needed for its construction, by itself it cannot process this information; it always depends on its builders for transforming information into action via computation. Even though this way the nest becomes a figurative extension of the wasp, the structure and its builders are still physically separate; the subject and object of generative spatial computation in this case are distinct entities.

Yet, there is a whole other class of self-organizing structures in nature where the builders holding the computational power and the building blocks making up the structure are one and the same. One example where the subject and object of generative spatial computation are merged in this way is the development of an embryo, by which a single cell develops into an extremely complex architecture made up of trillions of cells. In the next case study, I will introduce the development of an embryo as another prime example for how nature constructs itself in a computational fashion, wherein, once again, the rules of computation are executed by, and embodied in, the builders of the space, the cells; and the information cells compute with, i.e., local cues, is transmitted through the space via chemical or mechanical signals.

2.2.4 Embryonic Cells Execute a Set of Generative Rules to Build Organisms

Each embryonic cell encodes in its DNA a series of generative rules necessary to construct different parts of the embryo, reminiscent of how wasps use different rule sets to construct different parts of their nests. The major difference is that, while wasps use inert building blocks (mud bricks) to build the structure, biological cells use themselves. Cells self-assemble

into complex architectures wherein structural building blocks are living computational agents. In this way, biological cells form living structures that compute.

Yet how do cell construct these complex architectures in the first place? We have already discussed how wasps build complex structures with no shared blueprint in mind but rather by employing rule sets unique to different stages of construction based on local cues. As shown by Smith, wasps are so unaware of the global layout of their nest that one can "trick" them into building a mutant nest by editing the local cues. What about embryonic cells? Are they aware of the global layout of the mature embryo such that they know whether to become a part of an external appendage or an internal organ based on where they happened to be located on the "map"? No. Similar to the wasps building a nest, cells building an embryo are unaware of how the mature embryo is supposed to look. Rather, they encode in their DNA a series of generative rules enabling them to respond to local stimuli.

Even though all embryonic cells have the same DNA, as they all descend from one initial cell, what is encoded in this DNA is not a descriptive blueprint shared globally across the embryo. Each embryonic cell encodes in its DNA a complete set of rules to build an entire embryo, yet not each and every cell in the embryo executes all these rules simultaneously. Instead they collectively build the embryo with a self-organized division of labor. As a result, which specific rule set(s) a cell will pick from its DNA to execute is determined by which local signals it receives from its neighboring cells at a given time.

One of the fields to which we owe this understanding is experimental embryology, where a foundational experiment was undertaken in 1924, by Hilde Mangold and Hans Spemann, called *the Organizer Experiment* [25]. Even today, this experiment still is one of the best illustrations of how the early embryo's initially uniform cells differentiate into distinct body parts —not based on an elaborate global blueprint, but rather based on local signals coming from neighboring cells.

The Organizer Experiment involved two identical healthy frog embryos early in their developmental process [Fig2-31a, b], wherein each embryo could become a healthy tadpole [Fig2-32] if allowed to grow normally. The cells in these early embryos had not yet differentiated into distinct body parts; nevertheless, they were beginning to sort into somewhat noticeable clusters. One of these clusters, the one forming slightly above the "blastopore lip," which is a "landmark" in the early embryo, was called the "organizer cluster" because it was hypothesized to be composed of cells that send local signals to their neighbors to solicit them into collectively constructing the central nervous system. In other words, which body part these neighboring cells are going to become was hypothesized to be contingent upon the local signals they get from the "organizer" cells, and not their knowledge of the whole. Mangold wondered, if this is the case, what happens then if an embryo ends up with two organizer clusters?

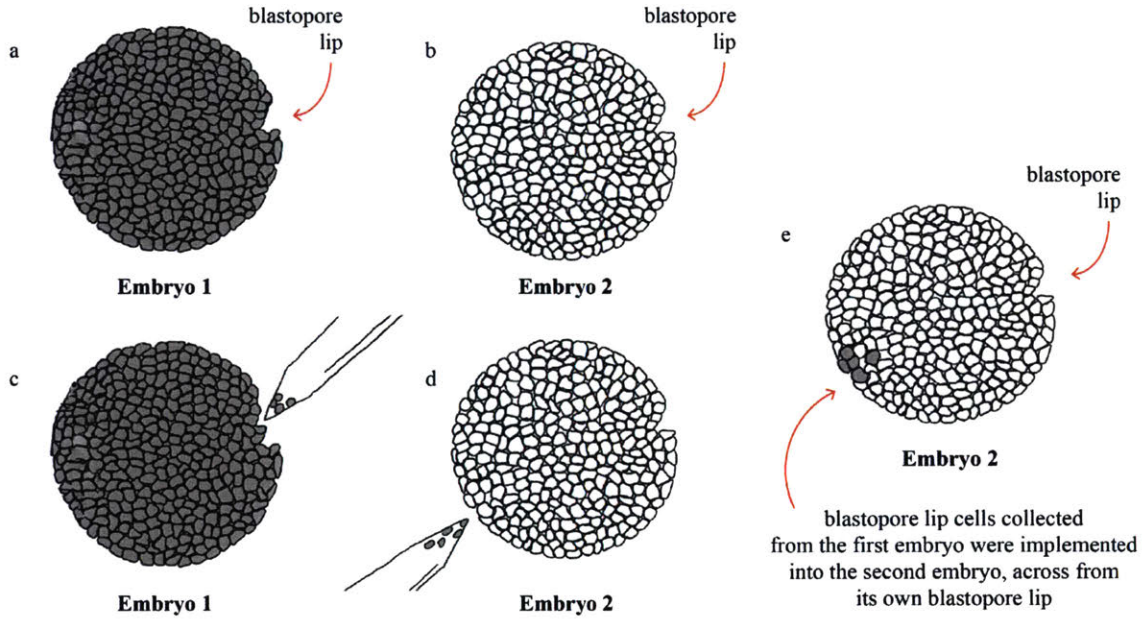


Figure 2-31: Experimental setup of the Mangold and Spemann Organizer Experiment. (a)Healthy embryo #1, blastopore lip shown with an arrow. (b)Healthy embryo #2, blastopore lip shown with an arrow. (c)Cells are being collected from the first embryo's organizer cluster. (d)Collected cells are being implanted into the second embryo, across from this embryo's original organizer cluster. (e)Mutant embryo with two organizer clusters across from one another.

Figure 2-32: A healthy tadpole.

To answer this question and test her hypothesis, Mangold collected some cells from the first embryo's organizer cluster [Fig2-31c], and implanted them into the second embryo [Fig2-31d], resulting in the second embryo having two organizer clusters on opposite sides. In this experimental layout, if the cells constructing the embryo indeed relied solely on local signals, instead of a global blue print, the now mutant second mutant embryo would develop two central nervous systems. This is because, in addition to this embryo's original organizer cluster, now there was a second, implanted organizer cluster. Cells now surrounding this implanted cluster were originally going to form the digestive tract of the tadpole, but they were now instead receiving strong local signals to build a central nervous system from the implanted organizer cluster. If cells followed a global blueprint, these cells in the second embryo would nevertheless build the digestive tract, remaining unaffected by the local signals they are receiving from the implanted organizer cluster. If this were the case, the experiment would result in the development of a regular tadpole. Otherwise, it would yield in a mutant embryo with two central nervous systems, a double-headed tadpole, which is exactly what Mangold observed at the end [Fig2-33].

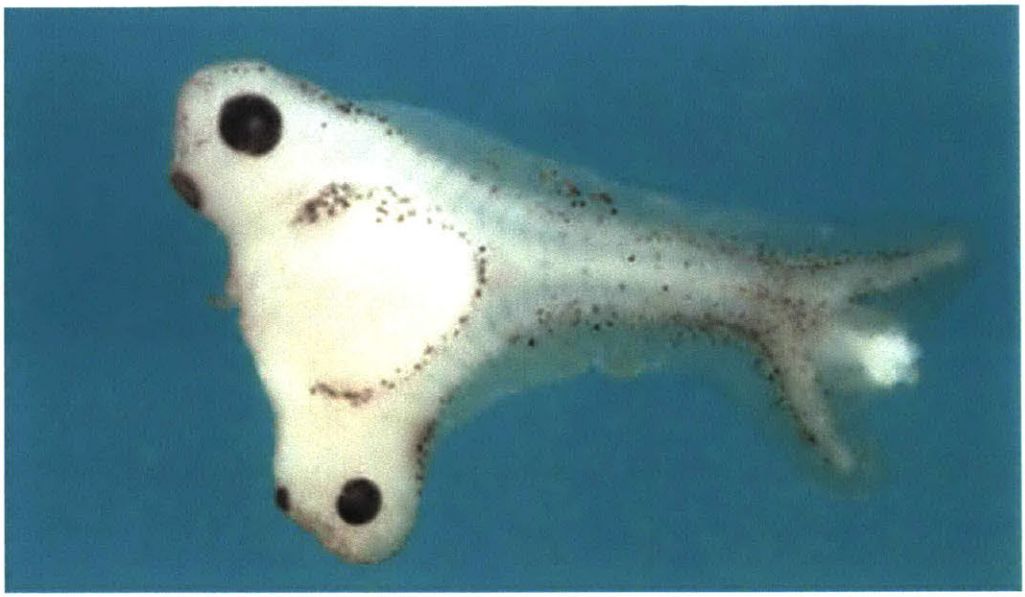


Figure 2-33: Double headed mutant embryo

This result shows that an embryonic cell is not predestined for any specific structure; it encodes a variety of rule sets, each leading to the construction of a different body part. Thus, whether a cell will become part of a digestive tract, or a central nervous system, or another structure is not dictated top-down by a blueprint, but is rather contingent on which generative rule set the cell will draw from its DNA based on what local cues it encounters.

For unveiling this exciting understanding of how embryonic cells follow a generative program, not a descriptive blueprint, the Organizer Experiment was awarded a *Nobel Prize in Physiology or Medicine in 1935.* What is even more exciting is to realize that at the time DNA was not known; hence, Mangold and Speman did not many things about the mechanistic details of this process. Yet, they were still able to get a group of embryonic cells to grow a mutant structure on demand, only by the knowledge of which rules result in what type of structures in that particular system. This is yet another example showing the fundamental role of generative rules in governing a self-organized construction process.

In this chapter, through various examples and case studies, I demonstrated that biological systems construct themselves in a computational fashion, using a generative language. Such

an understanding is critical to employing a biological system as a design medium, as is the premise of *synthetic morphogenesis* powering the work presented in the thesis as a whole. To this extent, in the next chapter, I show how an understanding of the generative language by which multi-cellular systems operate can enable us to edit these processes and fabricate artificial morphogenetic building elements, with which to create self-constructing structures in architectural scale.

Chapter 3

Methodology

In Chapter 2, I introduced nature's self-organizing construction framework, wherein numerous similar builders generate complex spatial constructs by processing local cues using a set of generative rules, rather than following instructions from a blueprint or a central governor. In this Chapter 3, I introduce the term *Generative Spatial Computation* to formally describe this computational process by which nature constructs itself. The generative rules required for generative spatial computation are always physically encoded in the builders themselves, which in some cases remain external to the emerging structure in some cases (e.g., wasp nest introduced in 2.2.3); yet, in some other cases, they become a part of it (e.g., frog embryo introduced in 2.2.4). It is this latter approach that constitutes a rather astonishing and rare fabrication method where the generative spatial computation is fully embodied; the builder, the building material and the specifications for building are all merged in a single computational building block making up the final structure: the cell.

3.1 Generative Spatial Computation in Multi-cellular Systems

The process of generative spatial computation starts in a cell when it obtains information from the environment in the form of local cues. These cues —whether they are the presence of certain chemicals in the environment, changes in pressure or temperature, mechanical impacts, or cell-to-cell communication signals coming directly from the neighboring cells— are essential to providing the cell with the information it needs to compute. Such diverse information is received and selectively imported into the cell via its hundreds of different specialized signal receptors. Once inside the cell, the signal activates a specific generative rule, guiding the cell through the steps necessary to devise a functional outcome. Different generative rules are activated inside the cell in response to the reception of different local signals. In other words, once a local signal is received by the cell, its respective signal transduction pathway gets switched on, wherein a series of intermediate proteins process this signal and relay it into the DNA. These intermediary proteins that make up a signal processing pathway are analogous to the series of if-then statements that make up a generative rule.

Once a signal is processed and relayed into the DNA, it can turn certain genes ON or OFF, enabling the cell to formulate a tremendously wide variety of outputs. For example, as an output, the cell can generate a communication signal and export it to the "extracellular milieu," where this signal becomes a local cue for other cells to then import, process and carry out the generative spatial computation further. Or, the cell can undergo morphological change, initiating the next stage of embryogenesis, as well as it can start producing a new biomaterial, which may alter the functional characteristics of the emerging structure. Whether the output is to foster exchange of signals between cells, initiate a structural transformation, develop a new material, or any combinations thereof, it always has a *spatial* consequence. In this way, executing generative spatial computation keeps the cells in an

active state of building throughout the morphogenetic development: cells synchronously migrate, differentially proliferate, collectively fold, chemotaxically locomate, adhere, sort, or fuse and thereby form sophisticated multicellular structures [Fig3-1].

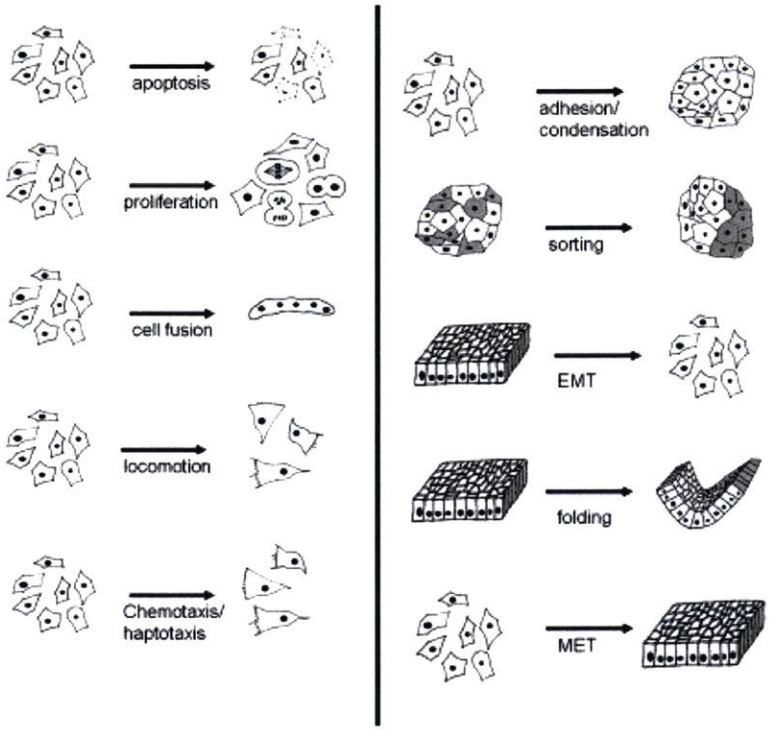


Figure 3-1: Different morphogenetic functions cells execute during embryogenesis. *Image credit: Jamie Davies*

Beyond such a tremendous capacity for embodied computation, as well as structural and material development, the ultimate power of a biological cell is its ability to self-replicate. A single cell can self-replicate and exponentially grow into an army of cells. In doing so, it also faithfully copies its DNA, ensuring an accurate distribution of the generative building rules to this emerging army of builders. Thus, by continuously self-replicating and concurrently executing generative spatial computation, biological cells in nature collectively construct architectures whose structural and material specifications are encoded in their constituent cells' DNA, in the form of a set of generative rules.

3.2 Editing the Generative Rules of Spatial Computation

Such an understanding of construction in multicellular systems raises the question: can we edit the rules? DNA is nothing but an exhaustive *rule book* that contains all the rules necessary for a cell to determine which output it should formulate in response to which cues are present in the environment. The rule book is tightly packed into a cell's 10um-wide nucleus, yet in fact it is a two meters-long molecule; being able to physically edit this molecule provides us a tangible interface for controlling the complex computational processes carried out in a cell. By using the synthetic biology theory and techniques, we can introduce new generative rules into this 2 meters-long rule book, and thereby redesign how a cell processes the local cues. In this way, *we can repurpose construction in multicellular systems to serve as an innovative, novel fabrication method, wherein, based on the new generative rules in their DNA, builders generate artificial architectures by design.* Since these builders, which can perpetually self-replicate, are also the building blocks of these architectures, such a novel fabrication method would be supported by an infinite supply of building material.

To demonstrate this, in this section, I introduce *Fabrication of Autonomously Constructed Engineered Three-dimensional Shapes (FACETS)* [Fig3-2]. Here, we introduce *a set of artificial generative rules* into a cell [Fig3-2a] for it to thereby self-replicate and develop into a specific artificial multicellular architecture [Fig3-2e]. In this way, we harness biological cells as an infinite supply of building material with embodied building instructions.

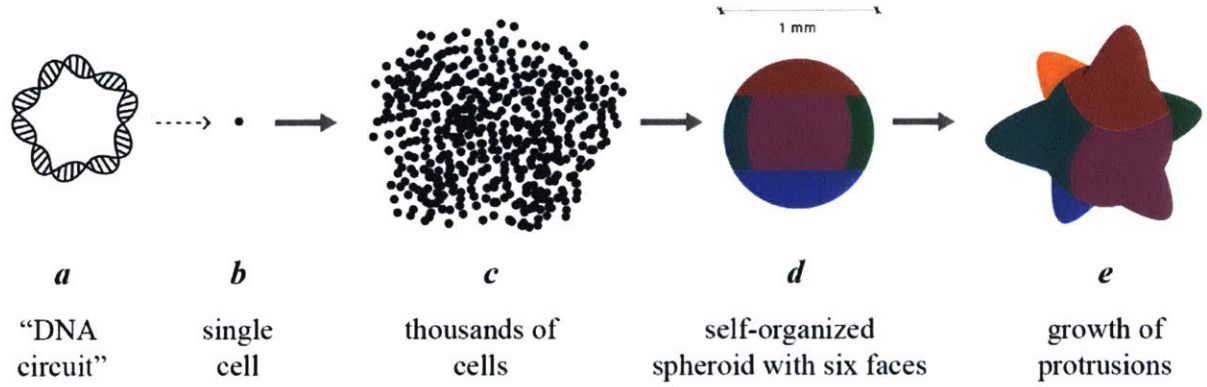


Figure 3-2: Schematic showing the sequential formation of a morphogenetic building element, FACETS, out of a single cell. (a) DNA molecule encoding the artificial generative rules that enable cells to process local cues such that they can develop into a FACETS structure. (b) This DNA molecule is then inserted into a single cell, which self-replicates into an army of cells, all carrying the same genetic material (c). Through these artificial generative rules, cells self-organize into a spheroid with six faces (d), from any subset of which limb-like protrusion(s) can develop (e).

Each FACETS develops out of a single engineered cell [Fig3-2b], into which we insert a DNA circuit encoding a set of artificial generative rules [Fig3-2a]. As this one cell self-replicates [Fig3-2c], it starts to use these rules to spatially compute and grow into a multi-cellular spheroid with 6 distinct faces [Fig3-2d]. This six-faceted spheroid spans half a millimeter in diameter and consists of approximately 10,000 cells [Fig3-3a]. Each one of the faces on this spheroid can then generate different structures, such as a limb-like protrusion [Fig3-3b].

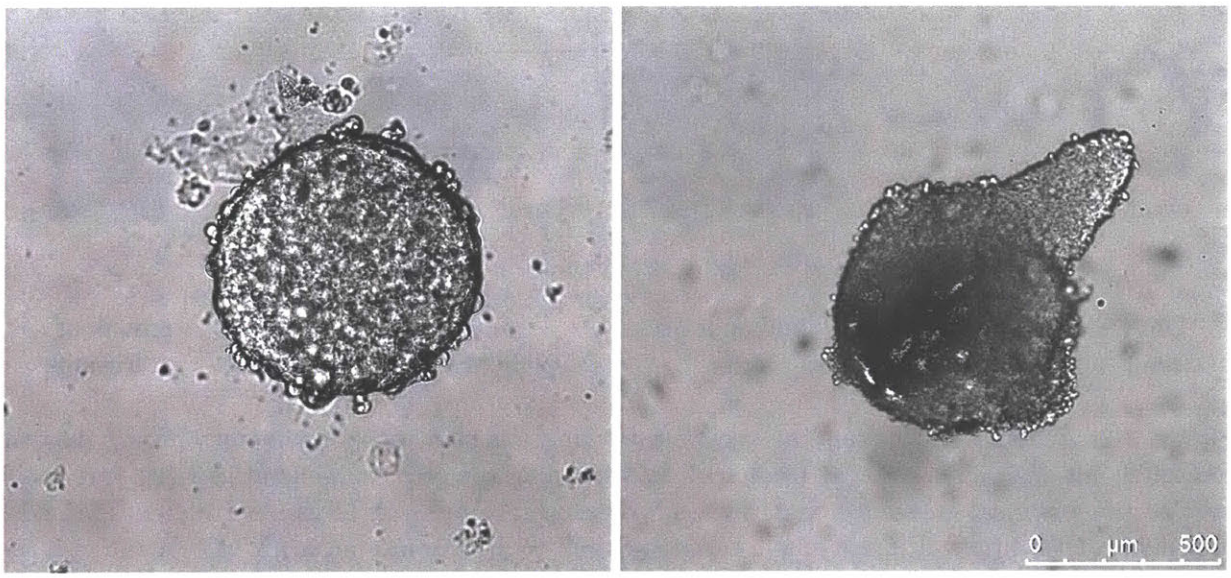


Figure 3-3: (a) A FACETS precursor spheroid made up of approximately 10,000 cells. (b) A spheroid with a single protrusion, image courtesy of Jesse Tordoff.

. Fix [Fig3-4] shows a step-wise illustration of the artificial generative ruleset enabling these cells to execute generative spatial computation and grow into the artificial multi-cellular structure of a FACETS. This artificial generative ruleset is encoded in a *synthetic genetic regulatory network* made up of DNA, which we have been concisely referring to as a "DNA circuit" [Fig3-2a]. Once this DNA circuit is inserted into a cell, it enables the cells to execute a series of artificial rules, each facilitating a different morphogenetic function shown below, including differential adhesion, Turing patterning, Band-pass filter, and a genetic clock. Only one of these morphogenetic functions are purely temporal: the "clock," which is an oscillator that runs at the cellular background and temporally organizes the remaining rather spatial morphogenetic functions.

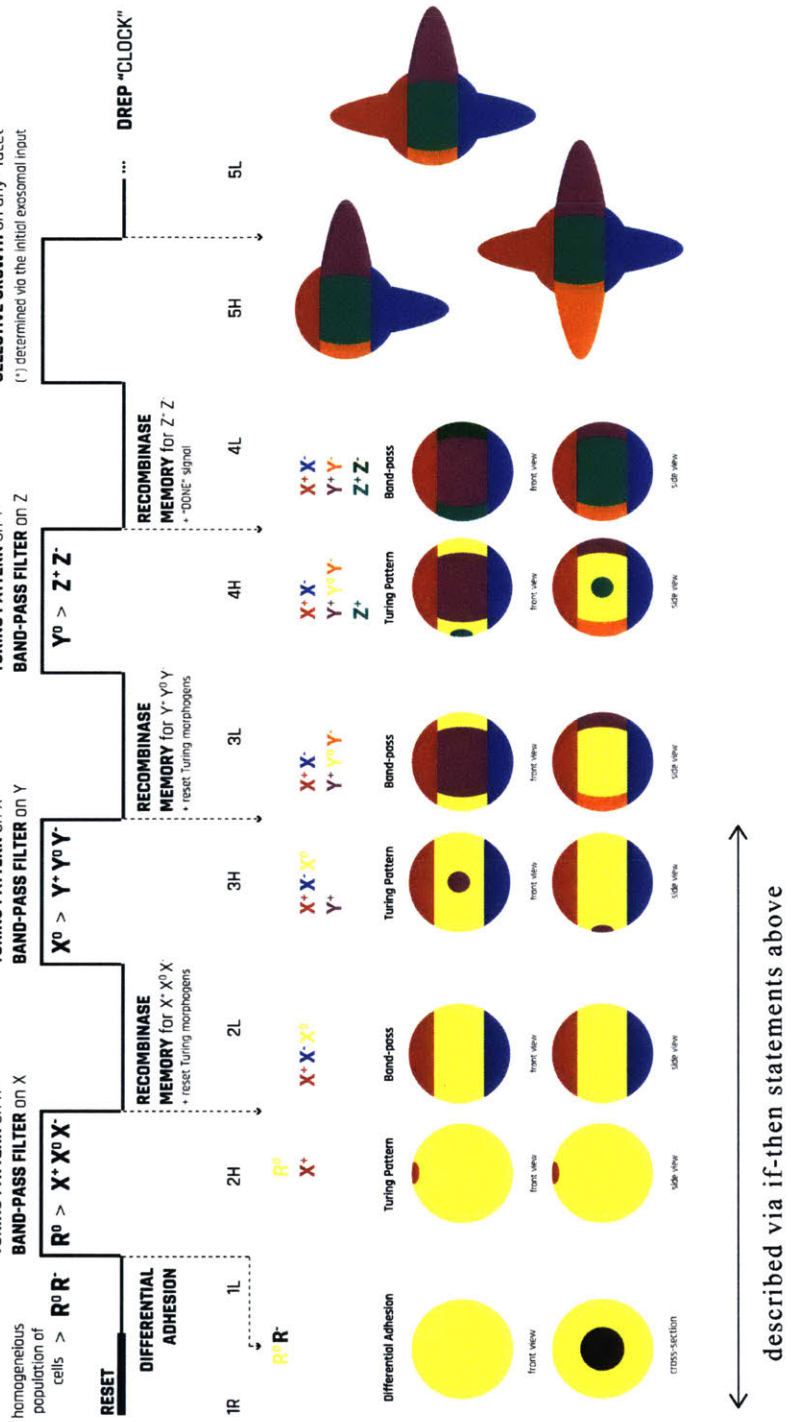


Figure 3-4: Artificial generative rules inserted into a single cell constitutes a morphogenetic algorithm governing the development of an arbitrary "FACETS" shape out of a single cell. Diagram above shows the procession of these events and the resulting phenotypic level changes at every step. The indicated portion of these artificial rules are also described via the if-then syntax below.

Below, I introduce this morphogenetic algorithm's first half (indicated with an arrow on Fig3-4) in its if-then syntax[1]:

If *clock = reset*

random toggle (s , w)[2]

If *toggle = s*

Express strong cadherin

Commit "R-" cell fate into genetic memory

If *toggle = w*

Express weak cadherin

Commit "R0" cell fate into memory

If *clock = high*

Turing patterning

If *in the active zone*

Commit "X+" cell fate into genetic memory

Establish morphogen gradient

Else *preserve cell fate until morphogen*

If *morphogen = high*

Commit to "X+" cell fate

If *morphogen = medium*

Commit to "X0" cell fate

If *morphogen = low*

Commit to "X-" cell fate

[1]Even though some of the consecutive rules are best represented via state machine diagrams, such as the clock and the Turing patterning functions, the overall artificial generative ruleset encoded by the master morphogenetic algorithm is still best represented via a set of if-then statements

[2]"s" for strong cadherin, "w" for weak Cadherin

Morphogenetic functions introduced in this first section of the synthetic genetic regulatory network included the differential adhesion, Turing patterning, and Band-pass filter, which are further discussed below. The remainder of the morphogenetic algorithm simply consists of the recursive employment of this first section twice, followed by the differential growth function at the end. Differential Growth function enables a particular subpopulation to start growing much faster than the remaining population, and thereby form a protruding limb bud-like structure. However, to be able to get to this final function, cells first need to execute the functions below, in the context of the artificial if-then based generative rules above, also illustrated on Fig3-4.

- **Differential Adhesion** function creates two subpopulations expressing different amounts and types of *Cadherin* (an intercellular adhesion protein), which causes cells to spatially segregate into two concentric layers within the spheroid, forming its core and the crust layers (for a deeper discussion, see Chapter4). The generative rule necessary to have a group of cells undergo such a self-organized stratification behavior can be represented with the simple if-then syntax below. This function is also the first spatial function called by the complete set of generative rules encoding for a FACETS shape, which we will examine shortly.

<random toggle (s , w) [3] *>*

If *toggle = s*

Express strong Cadherin

Else

Express weak Cadherin ***If*** *strong Cadherin*

Bind to other strong Cadherins

[3] "s" for strong cadherin, "w" for weak Cadherin

Else

Bind to either Cadherins

- **Turing Patterning** function can be employed by having each cell in a group of homogeneous cells to simultaneously create two orthogonal chemical species: a fast diffusing repressor molecule and a slow diffusing activator molecule. A dynamic balance between these two molecules fosters a "symmetry break" in this population, and depending on the relative diffusion rates between them, it thereby generates either spot or stripe patterns. Contrary to most generative rules we have discussed so far, the most appropriate formal representation of a Turing Pattern function is a finite-state machine diagram [Fig3-5]. In Chapter 4, I demonstrate how such a finite-state diagram can be converted into a DNA circuit for cells to execute the generative rule for Turing patterning.

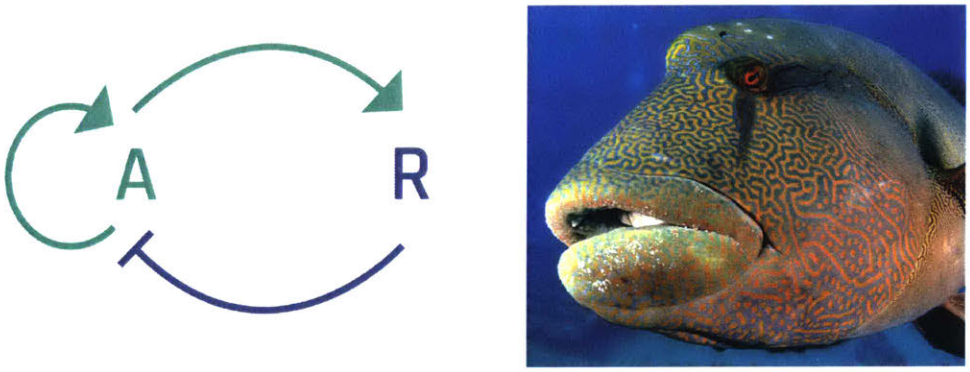


Figure 3-5: The generative rule needed for cells to locally compute when creating a Turing Pattern. Represented as a finite-state machine diagram.

- **Band-pass Filter** enables a group of cells surrounding an activation spot to self-organize into concentric layers based on how far they are from this spot. This is another function that is best represented as a generative rule via an if-then syntax, which is shown below. The DNA circuit diagram for this positional value acquisition behavior can be found in Basu et al.'s *A synthetic multicellular system for programmed pattern formation.*

If *morphogen = high*

Output #1

If *morphogen = medium*

Output #2

If *morphogen = low*

Output #3

In this way, FACETS constitute an example of an artificial multi-cellular structure that can be created with the synthetic morphogenesis approach. Currently, the biggest programmable structures we can develop in this way is only at the scale of millimeters. This is mainly due to the limitations of cellular viability; as the multicellular structure gets bigger, the cells deep inside start to lose their access to surrounding oxygen and nutrients, resulting in cell death and ultimate collapse of the entire structure. Therefore, a novel approach for scaling them into bigger structures is needed. For this purpose, next I introduce ASSEMBLE (Architectures from Staged Self-assembly of Morphogenetic Building Elements), which facilitates the self-assembly of millimeter-scale morphogenetic building elements into centimeter-scale structures.

3.3 ASSEMBLE: Architectures from Staged Self-assembly of Morphogenetic Building Elements

ASSEMBLE is a novel application of synthetic morphogenesis, where the *morphogenetic building elements* are employed as distinct building blocks that can attach to one another via *physical assembly cues* that they are programmed to grow on their surfaces. A morphogenetic building elements is a high-order structure formed via the process of synthetic morphogenesis, such as a FACETS, with structural properties that can serve as joints with specificity. For

example, the limb-like protrusions of a FACETS can function as such joints connecting one FACETS to another. However, for these protruding joints to serve as assembly cues, they must have *specificity*. We can achieve this by adding an extra morphogenetic step into a FACETS' synthetic morphogenesis algorithm: distal end adhesive protein expression [Fig3-6f]. Through this novel morphogenetic function, a FACETS can start expressing a specific type of adhesive protein at the distal end of each one of its protrusions. Hereafter I refer to these adhesive proteins as *glues* for concision.

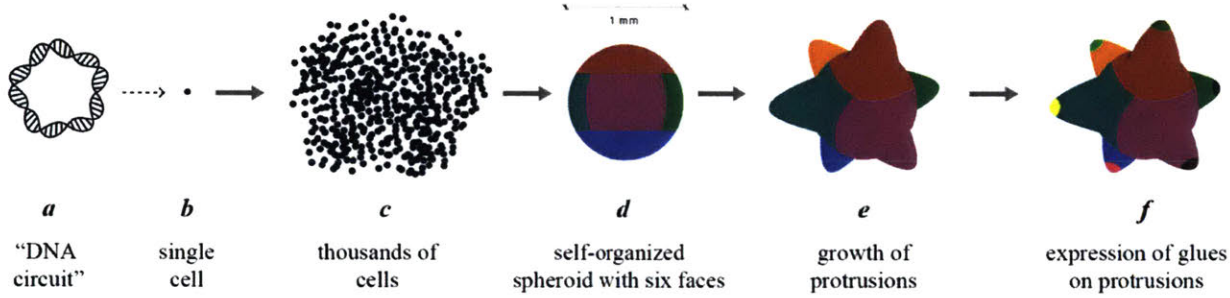


Figure 3-6: The same schematic showing the sequential formation of a FACETS as earlier, except this time, (f) adhesive proteins at the end of protrusions are also shown.

Each such glue has a strong preference for its own kind, for example glue type 'a' only connects with type 'a' and repels types 'b', 'c', or 'd.' Such a specific binding allows a group of FACETS with different number of appendages [Fig3-7] and type of glue to assemble in unique ways into higher-order 3D lattice structures [Fig3-8].

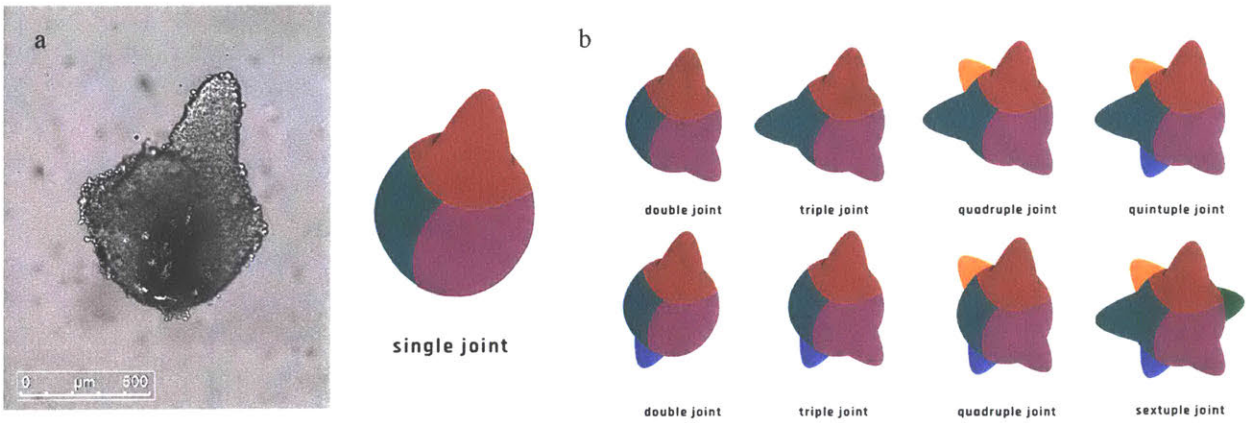


Figure 3-7: Protrusions that can grow from any subset of a FACETS' six faces can give rise to a library of different FACETS.

.

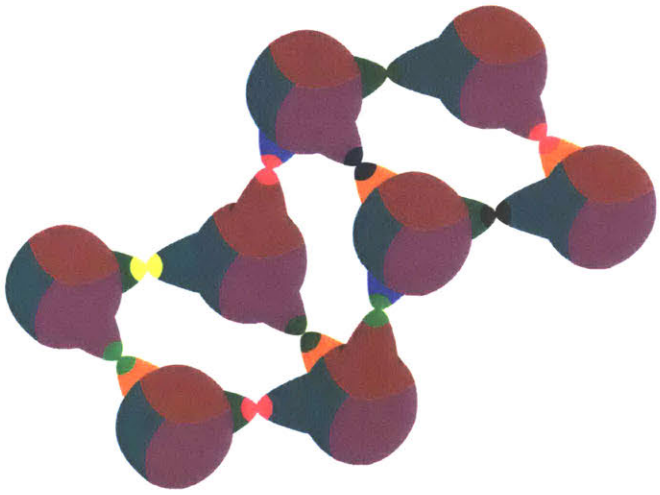


Figure 3-8: A higher-order 3D lattice structure self-assembling based on the complementary glues expressed at the distal end of FACETS' protrusions.

To better understand how these adhesive proteins enable individual building elements to selectively assemble together, one can think of these glues as joints in a timber structure [Fig3-9].

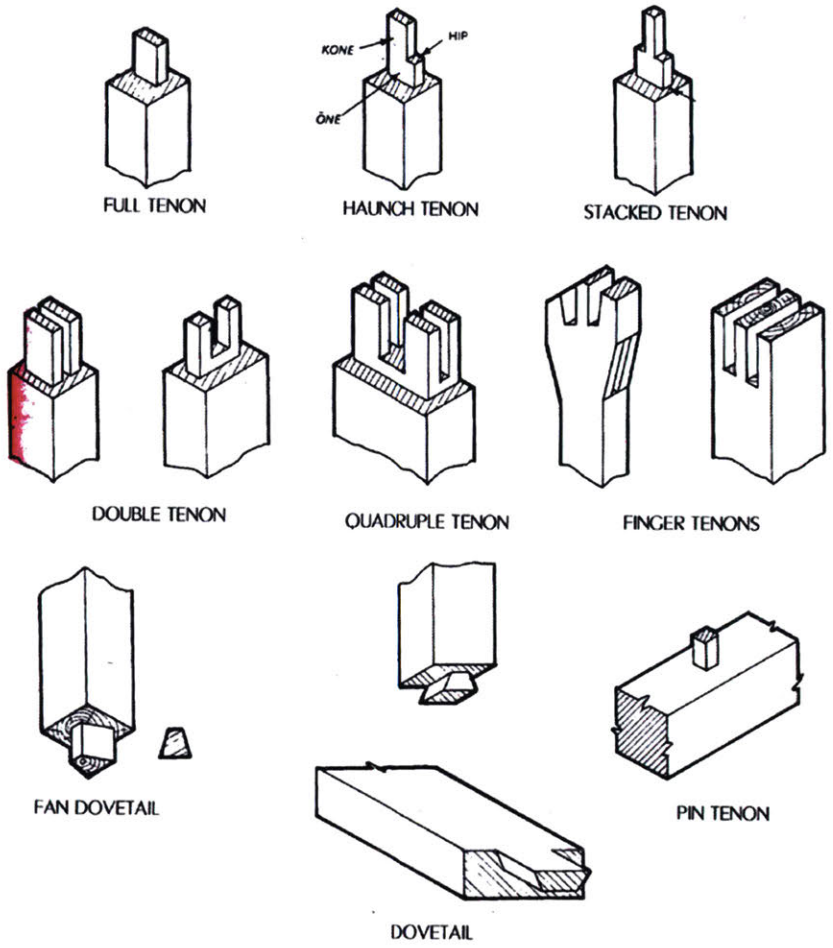


Figure 3-9: Various wood joints with specificity, analogous to the adhesive proteins I use to connect different morphogenetic elements with specificity.

A similar specific binding would allow FACETS with different numbers of protrusions and types of glue to assemble in unique ways into higher-order 3D lattice structures. In this way, FACETS become distinct building blocks that can attach to one another via embodied assembly cues that they are programmed to grow on their surfaces. The ultimate question then becomes: how can we identify the kind of FACETS required, in terms of number of protrusions and types of glue each need to have, for them to reliably self-assemble into a *specific target structure predefined by the user*? Here, ASSEMBLE's global-to-local structural compiler backbone *LACE (Lattice Adhesion Compiler Environment)* comes into play.

Preceding the fabrication process in the wetlab, user uploads a 3D model[4] of the target architecture into LACE, which first discretizes this structure into one made up of FACETS [Fig3-10a], and then assigns the number of protrusions and types of glue to each constitutive FACETS such that they will reliably self-assemble to this target structure when mixed together. For each distinct FACETS (with a unique protrusion count and type of glue on them), LACE specifies the generative rules necessary for it to develop out of a single cell via synthetic morphogenesis. LACE outputs these generative rules both in a human-readable syntax, such as a set of if-then statements, as well as in the form of DNA sequences to be used in the wetlab [Fig3-10b]. Upon identifying the generative rules a cell needs to use to grow each type of FACETS needed in this target structure, the user then proceeds from the CAD tool to the wetlab, where s/he constructs the DNA circuits encoding these generative rules [Fig3-10c]. Once a DNA circuit is inserted into a cell, the cell can execute the generative rules encoded in this piece of DNA molecule and use these rules to self-replicate and grow into the respective FACETS [Fig3-10d]. Thanks to the cell's ability to exponentially self-replicate, one cell and one DNA circuit encoding the rules for a given morphogenetic building element such as a FACETS is sufficient to grow millions of this building element. Finally, when mixed together in the order specified by LACE, these FACETS then autonomously self-assemble into the target structure [Fig3-10e].

[4]prepared in any mainstream modeling software which exports voxel maps

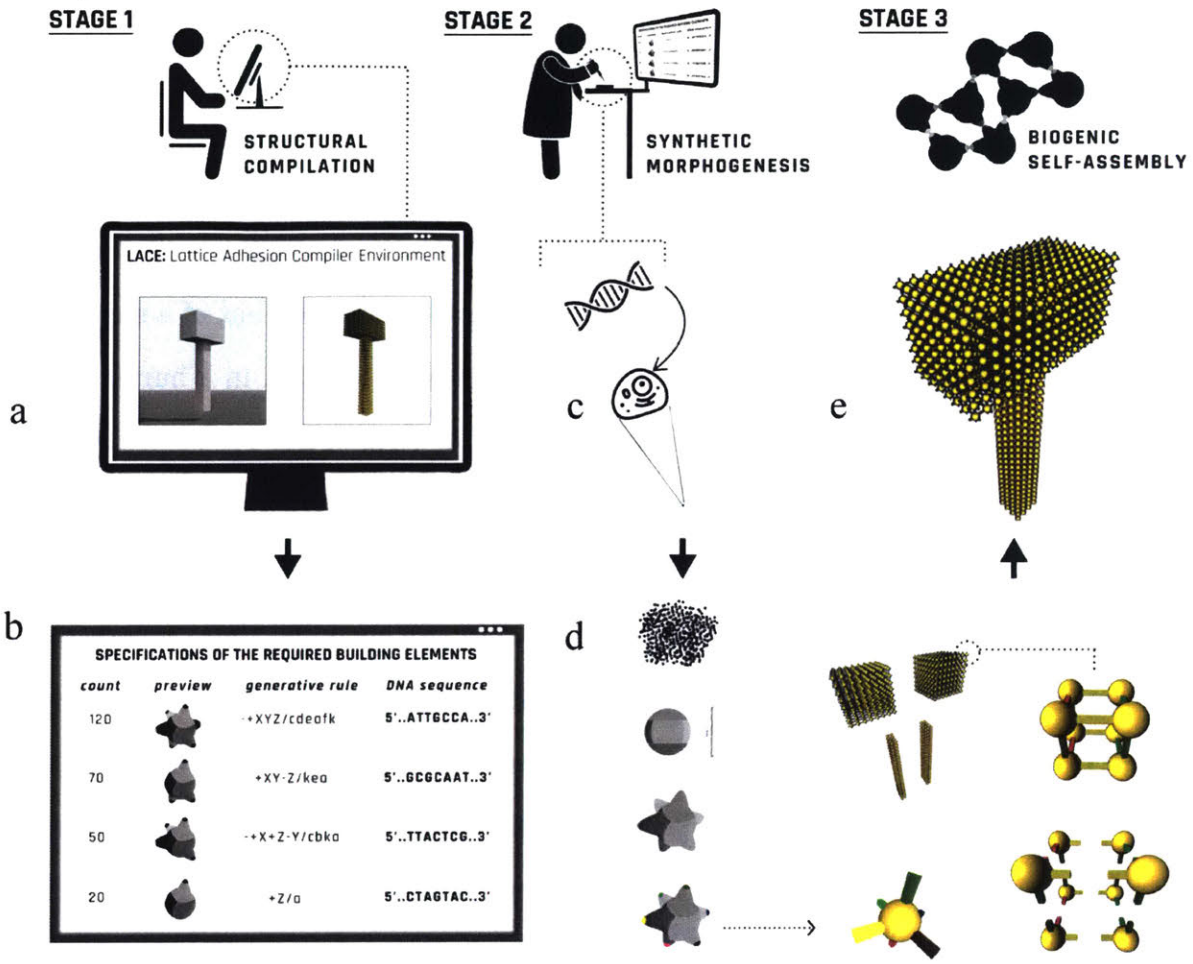


Figure 3-10: In stage 1, centimeter scale arbitrary target architecture, e.g., a sledgehammer, gets discretized into a lattice structure made up of "morphogenetic building elements," such as FACETS, on which the assembly cues are embodied as structural features (a). The global-to-local compiler's compilation output is then a specification sheet detailing the number and type of morphogenetic building elements as well as their respective generative rules (b). In stage 2, the fabricator then proceeds to the wetlab with this information. Here, the digitally identified DNA circuits are physically constructed and inserted into a series of single cells (c), thereby enabling them to grow into a series of morphogenetic building elements with the respective structural features. In this way, in Stage 3, these morphogenetic building elements are mixed in the sequence initially identified by the CAD software and self-assemble into the target structure.

In the next section, I introduce LACE's compilation algorithm, which is critical for not only identifying what kind of FACETS required, but also for specifying the order in which they should be mixed.

3.3.1 A 3D Global-to-local Structural Compiler

In any kind of assembly, structures need their building blocks to stay together. For self-assembly in particular, the assignment of glues to building elements is crucial not just for this final structural integrity, but also for limiting *how* the elements self-organize. Building elements which connect without specificity with each other may self-assemble into unintended structures. We mitigate this risk by using glues that adhere *only* to their own type when linking building elements —in other words, glues that are *orthogonal*.

At first, this orthogonality of glues suggests an intuitive solution for how to assign the assembly cues: simply use a different glue at each linkage in the structure. However, engineering a set of orthogonal adhesive proteins actually presents a significant biochemical research challenge. Therefore it is essential that we assign *as few distinct glues as possible* to ensure the feasibility of ASSEMBLE. For this purpose, we employ the staged self-assembly approach, initially proposed by Demaine et al. [26], where we spatially constrain which building elements can interact such that they first self-assemble into substructures, and then these substructures come together to form the target structure. To facilitate such a workflow, LACE outputs a *mixing graph* detailing which building elements should be mixed in what order so that the number of distinct glues needed for a particular target structure is kept at the minimum. To better understand this workflow, let us look at the mixing graph below [3-12] that facilitates the assembly of a 2x2 square made up of FACETS. In this graph, each FACETS is represented with a line abstraction shown on Fig3-11d.

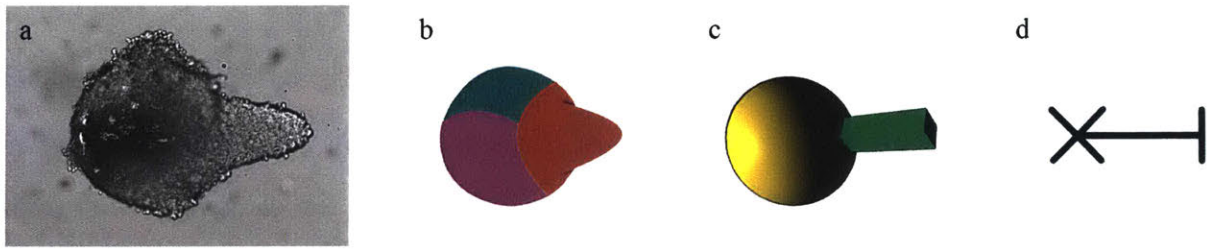


Figure 3-11: (a) An actual single protrusion FACETS in solution. (b) Diagrammatic representation of a single-protusion FACETS. (c) 3D compiler representation of a single-protusion FACETS. (d) 2D mixing graph representation of a single-protusion FACETS.

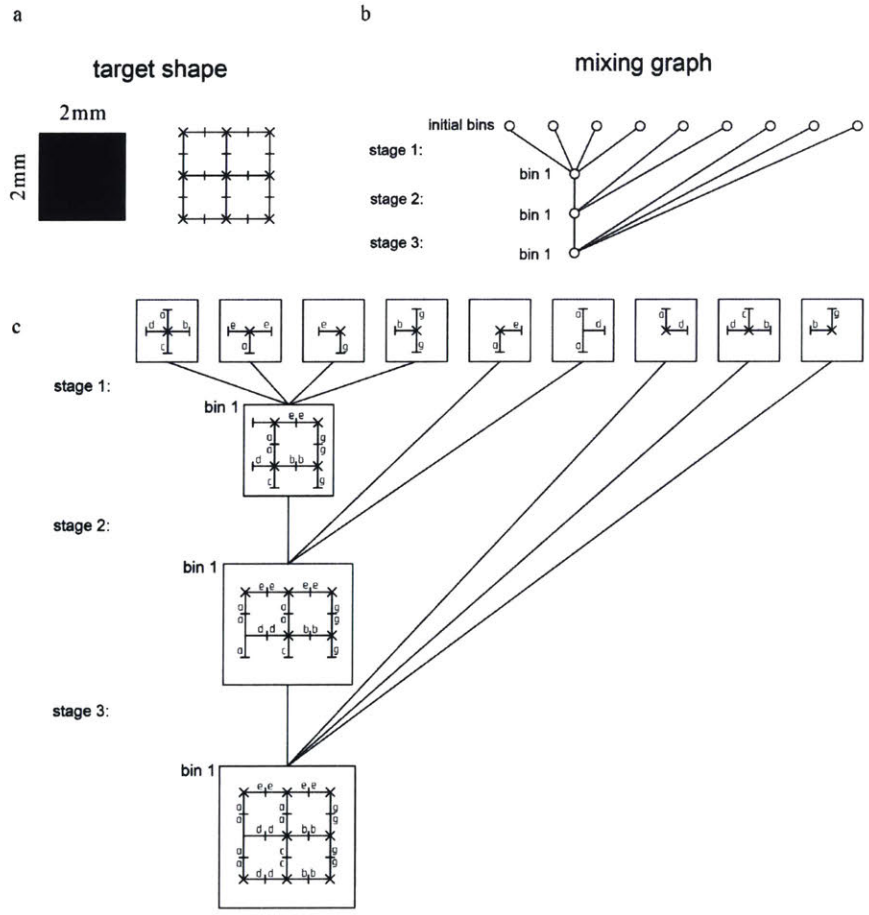


Figure 3-12

If we were to assemble this 2x2 square without staging, we would need to use 12 distinct glues instead of the 6 we were able to use above. Furthermore, as we increase the length of

this square, the number of glues needed stays the same. This is because by increasing the number of stages, we can reuse glues that are now "sealed" within a particular subassembly. In this way, addition of stages allows the scaling of structure with only a constant number of glues. These trade-offs are attractive for practical realization, given standardized bins are so much easier to acquire than developing orthogonal adhesive molecules that function as glues.

Cost and Benefit Analysis of Staged-self Assembly

More specifically, the cost of constructing a target structure using such a staged self-assembly approach may be defined through the following parameters:

- **Element complexity** describes the number of *distinct types* of FACETS needed.

  This value does not constitute a bottleneck for ASSEMBLE's fabrication process because not only the number of distinct types of FACETS can be used is limited by design (see FACETS library on Fig3-7), but also, even if a given target structure will need all the distinct types, switching between them is easy. An engineered cell with the master FACETS algorithm can be used to create any FACETS structure in our library without requiring any alterations at the genetic level. This is because how many protrusions a FACETS need to grow and which glue type should be expressed on it is determined by an initial "exosomal" input administered into the FACET's growth media. In this way, selecting between distinct types of FACET can be determined on the fly, without being have to change the DNA circuit each time.

- **Bin complexity** describes the number of *standardized containers* needed.

  This value also does not constitute a bottleneck for ASSEMBLE's fabrication process because, thanks to our bottom-up approach, we can use the same type of bins irrespective of the shape of the structure growing in it. This is an important achievement because, for many other current architectural biofabrication methods, bin complexity *does* constitute a significant bottleneck. This is because in these methods form does not come from

within, but from the container in which cells are grown. For instance, molding and casting-based biofabrication approaches require fabrication of a custom container for each distinct shape to be fabricated. Since in ASSEMBLE, we develop form from within, instead of form from imposition, we can always use standardized tissue culture wells as our containers. This standardization drastically reduces the cost of infrastructure needed for fabrication.

- **Glue complexity** describes the number of *distinct glues* needed.

  As discussed before, this is the value that could also constitute a significant bottleneck for ASSEMBLE's fabrication process since we have only so many orthogonal adhesive molecules that can function as glues. Therefore, we built the global to local compiler with minimizing this value as the single most important goal. Thanks to our staged self-assembly approach, we are able to keep this value constant as we scale up the structure.

- **Stage complexity** describes the number of *stages (i.e., amount of time)* needed.

  This value could also constitute a somewhat important bottleneck for ASSEMBLE's fabrication process. We envision a standard liquid handling robot to do the mixing of bins across stages, which does create some on an external equipment. Yet, this is a dependency for the shape acquisition, which is something we deliberately avoid with our morphogenetic approach, but one for the automation of mixing-graph execution and hence speed up the overall fabrication. Moreover, unlike other fabrication methods that depend *constantly* on the external equipment, such as 3D printing, we rather *intermittently* depend on this liquid handling robot. I other words, given the substructures in each bin take some time to assemble, *a single robot can be used to assemble many structures in parallel.* Therefore, even if the value of stage complexity ends up being relatively higher for some structures, it does not proportionally increase

the robot turnover time. Again, this is unlike other biofabrication methods that depend on external machinery for form acquisition, such as 3D printing, where one machine can only be used to build one structure at a time.

We see these considerations still hold true when we move from the 2D to the 3D case. The main difference in the 3D case is that the subdivision of a target structure is made along x, y, and z axes, instead of only x and y axes. In this way, the 3D case "flattens" the target structure into an *octree*, instead of a *quadtree* as in the 2D case, and assigns the glues in a similar fashion[27]. Such a spatial partitioning prior to the glue assignment is essential for reducing the glue complexity of the target structure. As a trade-off, it increases the bin complexity and stage complexity; however, as discussed above, this is a profitable trade-off in terms of being able to scale up self-constructing biological architectures. Below is an application of this binary partitioning principle on a non-discretized 2x2x2 cube:

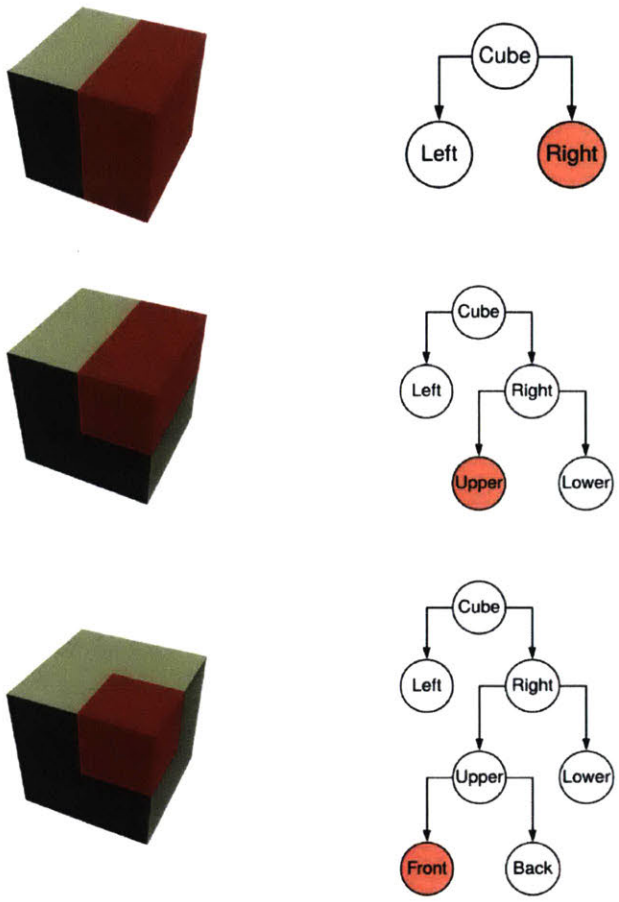


Figure 3-13: A target structure of a 2x2x2 cube is partitioned into an "octree," shown on the right. This tree defines the mixing graph for self-assembly, although specific glues are yet to be assigned to elements.

Below is an application of this binary partitioning principle on a discretized 2x2x2 cube made up of FACETS, shown in 3D compiler representation [Fig3-11c].

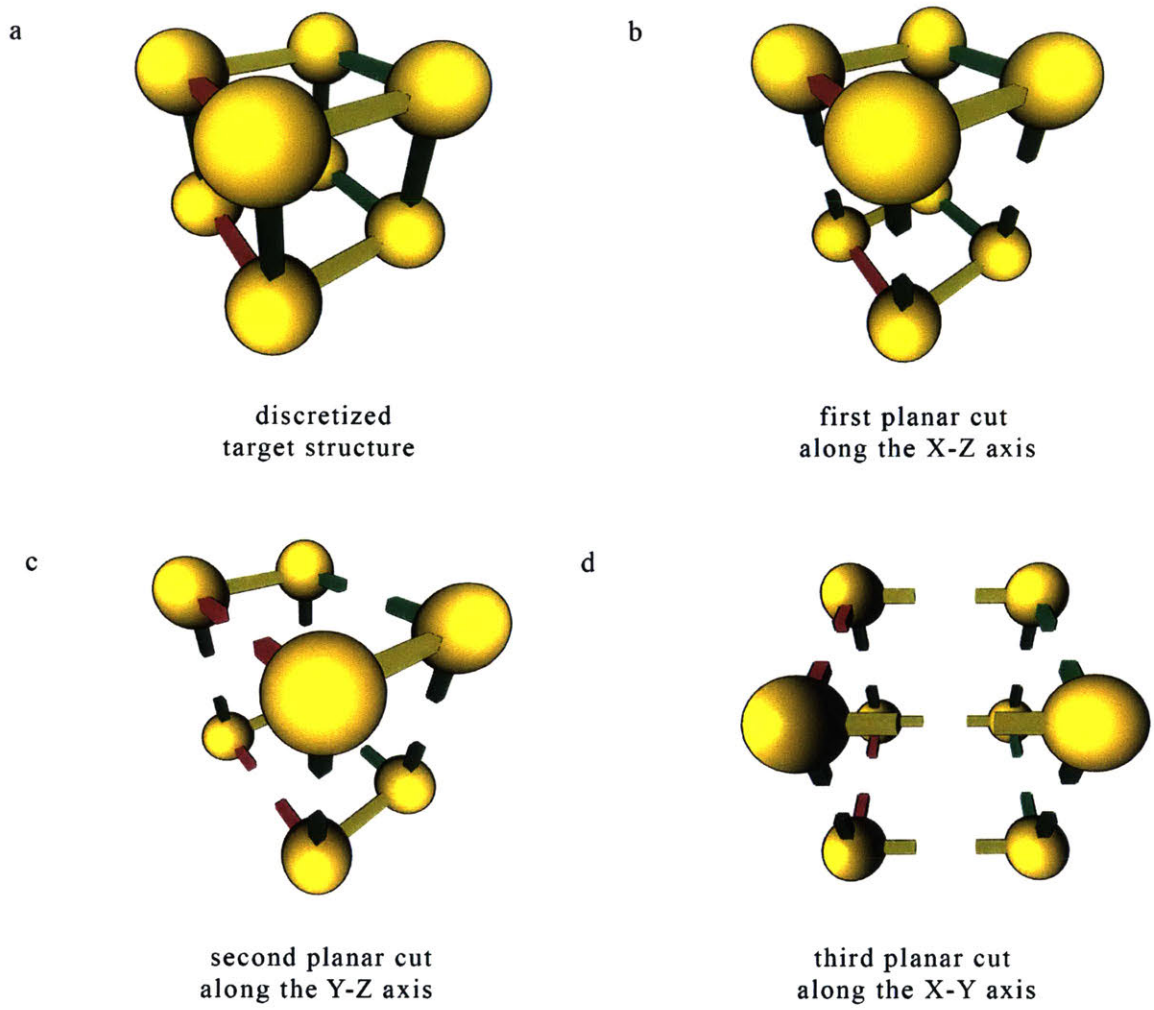


Figure 3-14: (a) Target structure discretized into a lattice made up of FACETS. (b) First planar cut on this structure along the X-Z axis. (c) Second planar cut on this structure along the Y-Z axis. (d) Third planar cut on this structure along the X-Y axis.

Finally, after having showed the basic binary space partitioning principle of our compiler in this simple cube example, we can now proceed to a more complex 3D shape, such as a sledgehammer [Fig3-17a]. (We picked the sledgehammer for an example as a playful tribute to it being one the first fabrication tools humans developed). In this example below, we apply the same octree cuts explained above, except that this time we employ it recursively until the high-level shape, the sledgehammer, is compiled into individual morphogenetic elements.

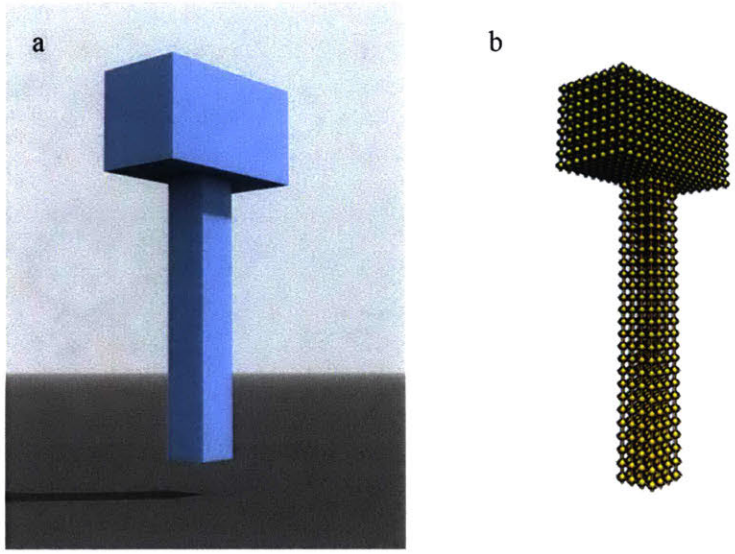


Figure 3-15: (a) A generic 3D model of an arbitrary target structure, a mini sledgehammer, with a height of approximately 3cm. (b) Discretized version of this 3D hammer composed of a series of FACETS.

After the target structure (i.e., the sledgehammer) is uploaded into LACE and got discretized into a lattice structure made up of FACETS [Fig3-17], the compiler then makes the first round of cuts along three axial planes, resulting in four substructures [Fig3-16]. Then, the compiler cuts these substructures once more in the same fashion, resulting in sub-substructures. The compiler continues to recursively cut each one of these smaller assemblies until each is compiled into individual morphogenetic elements. Once this is achieved, it assigns glues efficiently by using the staged self-assembly algorithm, which, as demonstrated in X and Y, uses the same internal glues in symmetric substructures since those glues will be "sealed in," hence will not interfere with one another, by the time different substructures make it to the same bin. In this way, these planar cuts not only efficiently generate sub-structures, but also serve as the axes of symmetry for mirroring internal glues.

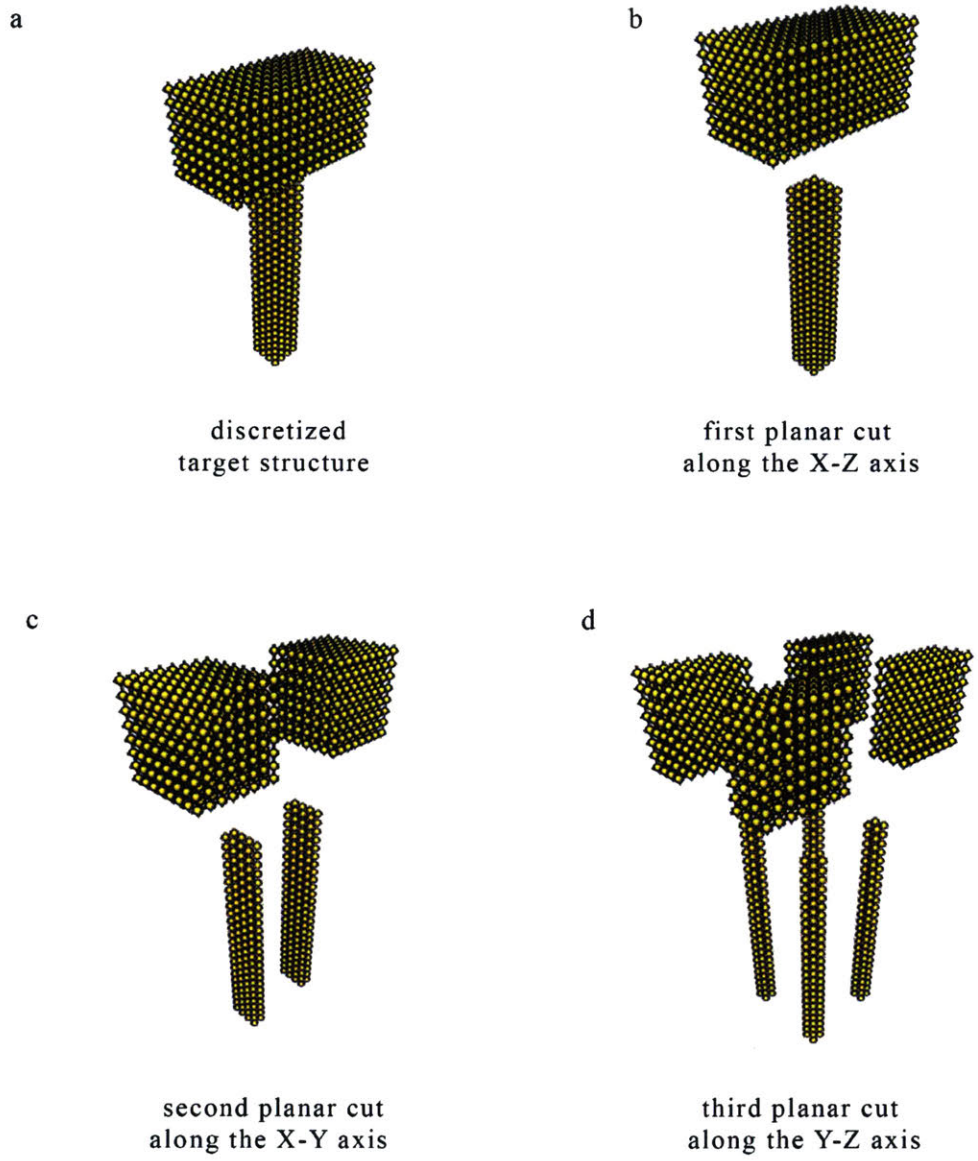


Figure 3-16: (a) Target structure sledgehammer is discretized into a lattice made up of FACETS. (b) First planar cut on this structure along the X-Y axis. (c) Second planar cut on this structure along the Y-Z axis. (d) Third planar cut on this structure along the Y-Z axis.

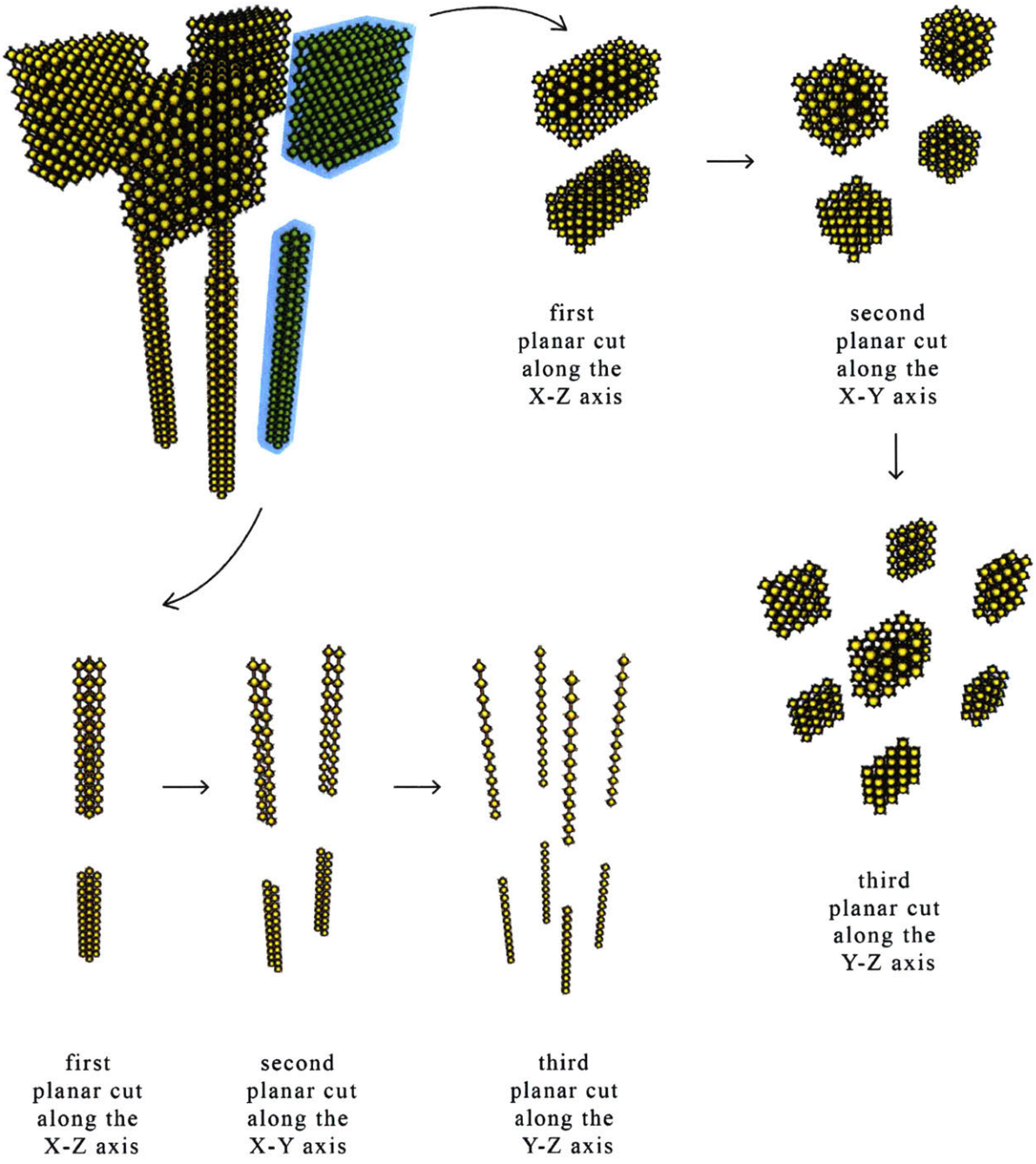


Figure 3-17: Second round of cuts

As a result of these 3D binary space partitioning and efficient glue assignment functions, for a given shape such as the sledgehammer, our staged self-assembly algorithm outputs:

1. a list of distinct kinds of morphogenetic building elements needed for self-assembly,

2. the generative rules necessary for each kind of morphogenetic building element in this list to be developed from a single a cell,

3. DNA sequences for these generative rules that the cell can understand and execute,

4. the order in which these morphogenetic building elements should be mixed (i.e., mixing graph) for them to reliably self-assemble into the sledgehammer.

In this way, our compiler LACE provides a profitable means by which to assemble morphogenetic building elements into centimeter-scale target structures. Our compiler can be found online at `http://www.research.xerarch.org/lace` [Fig3-18].

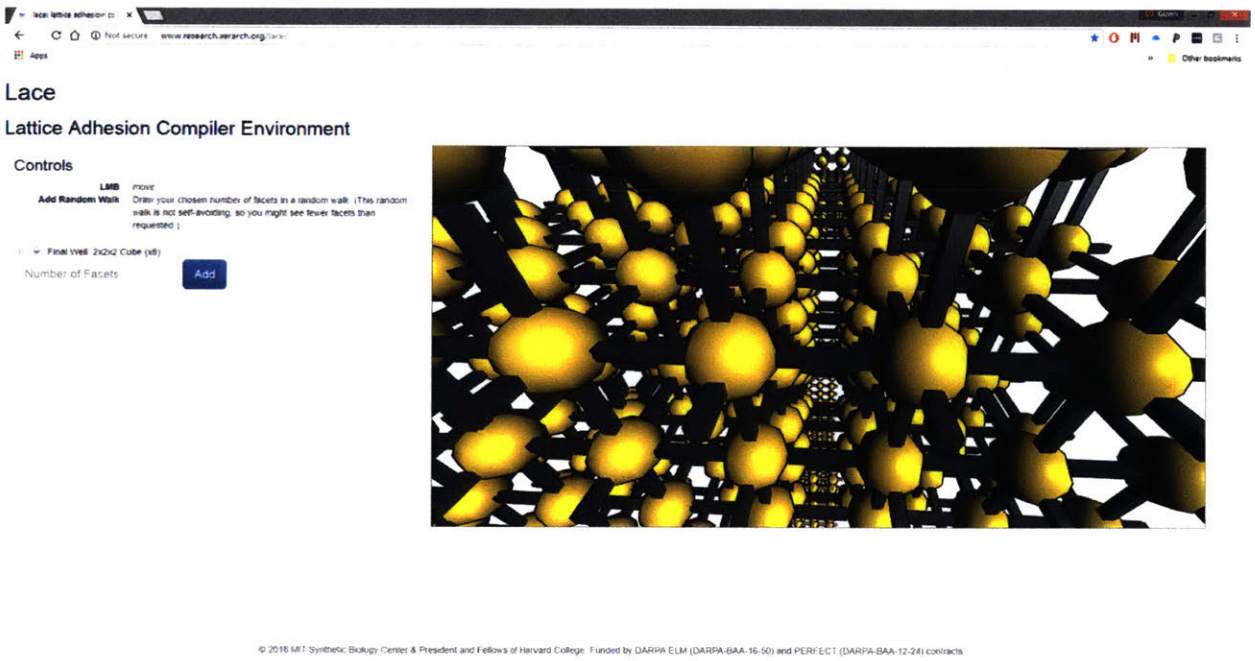


Figure 3-18

In this Chapter 3, I introduced ASSEMBLE's global-to-local compilation workflow. After the ASSEMBLE's initial step, in which a 3D model of the target structure is uploaded to LACE, LACE discretizes this structure into a lattice made up of FACETS and runs the staged

self-assembly algorithm on this lattice. This algorithm identifies and assigns the assembly cues required on each FACETS, such as the number of protrusions and types of glue on them, as well as the order in which these FACETS should be mixed (i.e., a mixing graph). When mixed together in this way, these FACETS autonomously self-assemble into the target structure initially specified by the user.

Chapter 4

Results and Discussion

So far we have been discussing the importance of generative rules as a tool for programming biological cells to construct structures by design. In 3.2, with FACETS, I showed how we can introduce a set of artificial generative rules into a cell for it to then self-replicate and build itself into a specific artificial multicellular architecture. In this chapter, I dive deeper into some of these artificial rules and demonstrate how we can convert them into the language of DNA and thereby get biological cells execute them. Work presented in this chapter primarily uses the Human Embryonic Kidney (HEK) cells shown below.

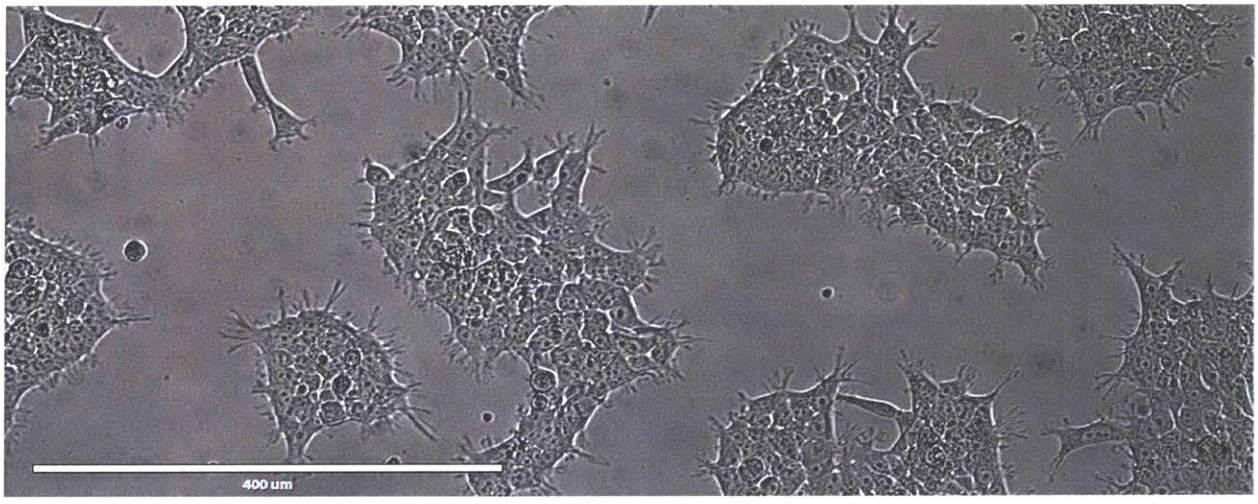


Figure 4-1: Human Embryonic Kidney (HEK) cells in 2D culture. I used HEK cells for the majority of the synthetic biology work presented in this chapter.

More specifically, in this chapter, I will primarily focus on the differential adhesion and Turing patterning functions. I will then present my results on improving the artificial cell-to-cell communication system *synNotch*, which is the short range activator morphogen playing a critical role in Turing patterning.

4.1 Differential Adhesion with Cadherins

In this section I demonstrate how the simple morphogenetic function of differential adhesion can enable the autonomous spatial segregation of an initially "well-mixed" group of cells consisting of two subpopulations with contrasting cell-to-cell adhesion strengths [28]. Such a contrasting intercellular adhesion between two groups is achieved by tuning both the expression levels and the innate binding strengths of surface proteins called Cadherins, which are analogous to "intercellular Velcro"s as they promote cell-to-cell adhesion. The control of the Cadherin expression levels are achieved at the DNA level, by using promoters with different strengths, whereas the innate binding strengths of Cadherins are modulated by using Cadherins with different conformations (i.e., different quaternary protein structures, see 2.2.1). Such contrasting adhesion affinities between two subpopulations give rise to a self-organized stratification behavior, where the strong adherents constitute the core of the sphere, and the remaining weak adherents surround them as a crust. This behavior can be captured in the following generative rule that the cells follow:

If *strong Cadherin*

Bind to other strong Cadherins

Else

Bind to either Cadherins

Before encoding this differential adhesion function in a DNA circuit for cells to execute, I first computationally simulated the expected self-segregation behavior both in 2D [Fig4-2a,b] and [Fig4-2c]. Such simulations enable us to encode the generative rule(s) for the desired morphogenetic function(s) in a set of mathematical equations for "virtual cells" to execute, instead of encoding them in a DNA circuit for biological cells to execute. In other words, ordinary differential equations (ODEs) below are analogous to the "virtual genetic material" of the cells.

$$\frac{d\,\text{eCadherin}}{dt} = \nu_{\max} - \text{eCadherin} \cdot \text{deg}$$

$$\frac{d\,\text{pCadherin}}{dt} = \nu_{\max} - \text{pCadherin} \cdot \text{deg}$$

For this set of ODEs, I encoded the differential Cadherin expression as hard-coded as binding energies into the terms "eCadherin," which represents the strongly adherent Cadherin, and "pCadherin," which represents the weakly adherent Cadherin. As a simulation platform, I chose to use *Morpheus* because it supports 2D and 3D modelling of dynamic phenomena in cellular populations by utilizing ODEs [29]. Below are the 2D and 3D simulations of this behavior using Morpheus [Fig4-2].

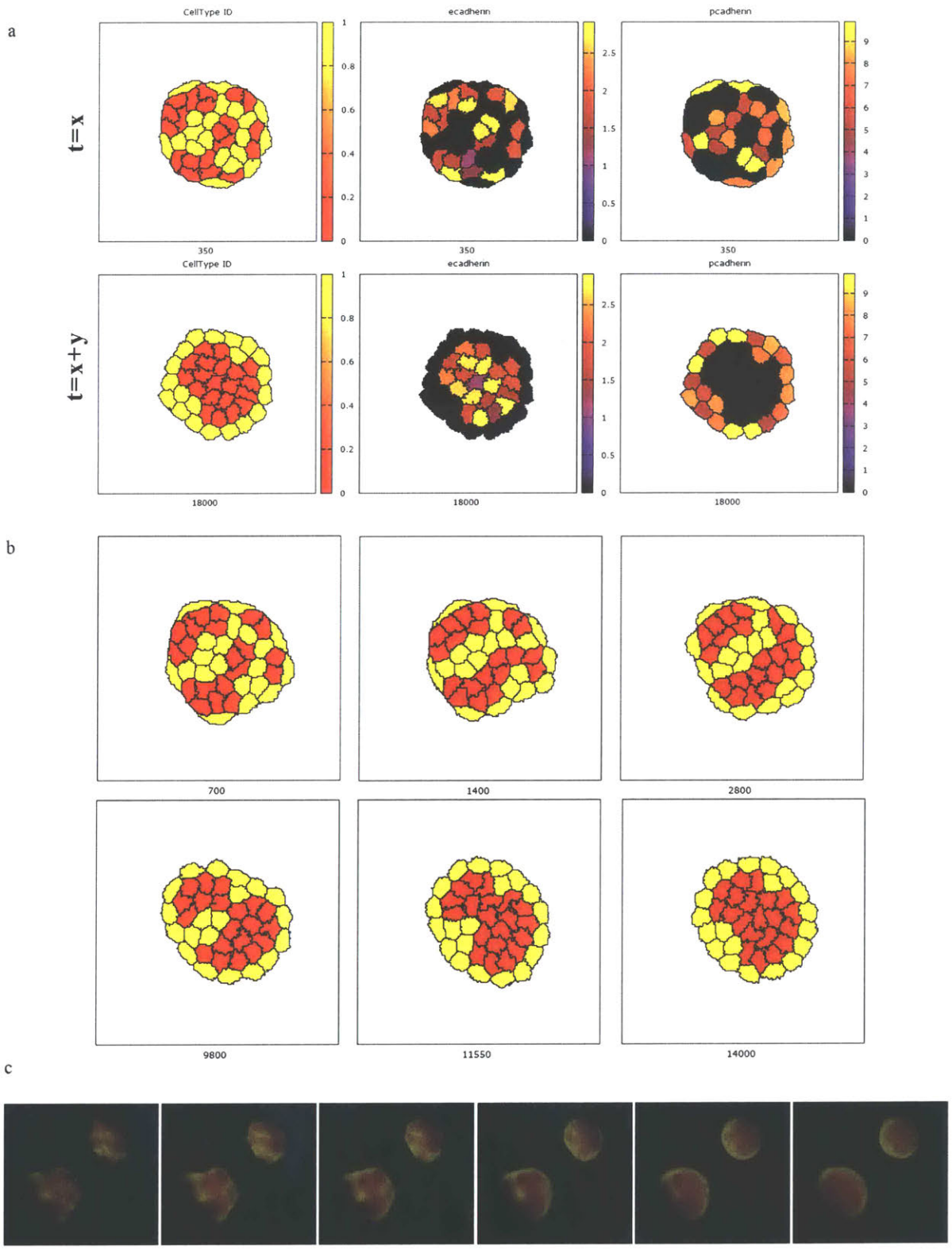


Figure 4-2: The differential cadherin expression enable a well-mixed spherical cellular aggregate to self-organize into two concentric layers that are the 'core' and the 'crust' of the spheroid.

Following the verification of the autonomous self-segregation behavior using an ODE-based simulation in this way, where the generative rules are encoded in a set of mathematical equations that the virtual cells compute with, I proceed to the wetlab to construct the DNA circuits, with which this time the actual biological cells can compute to generate this autonomous self-segregation behavior [Fig4-3]. Each DNA circuit has three major parts: i) *promoters* function like ON-OFF switches for the genes "downstream" of them and are often represented with arrows: ii) *genes* are stretches of DNA encoding for proteins that enable the cell to carry out its massively different functions, and are often represented with rectangular boxes following promoters; iii) *terminators* indicate where the gene ends to the cellular machinery that converts genes to proteins, and are often represented with a T-shaped cap following gene bodies.

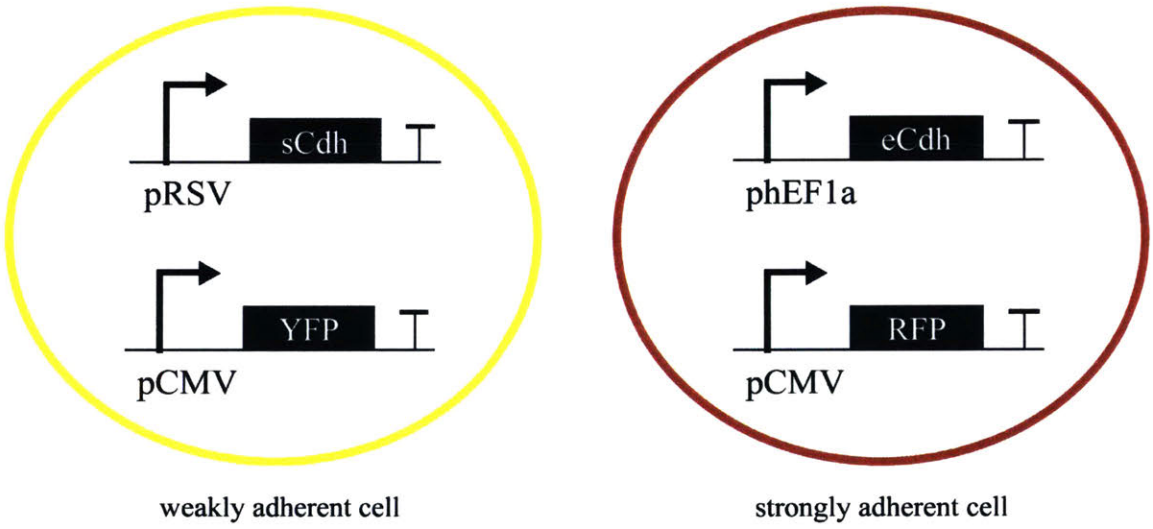


Figure 4-3: (a) Weakly adherent cell expresses the less adherent type of the Cadherin protein (pCadherin) from a weak promoter *RSV* that produces less copies of the gene downstream. This cell also produces the Yellow Fluorescent Protein (YFP) for us to be able to distinguish it under microscope. (b) Strongly adherent cell expresses the more adherent type of the Cadherin protein (pCadherin) from a strong promoter hEF1a that produces more copies of the gene downstream. This cell also produces the Red Fluorescent Protein (RFP) for us to be able to distinguish it under microscope.

Some promoters are always ON, some can be turned OFF in the presence of a "repressor" molecule, and some are always OFF, unless a certain "activator" molecule is present in the environment. Furthermore, different promoters have different strengths. For example, in the circuit diagram below, I use promoters with different strengths to promote contrasting adhesion affinities between two subpopulations. Weakly adherent population uses a weak promoter *RSV* to produce the less adherent type of Cadherin (pCadherin), whereas the strongly adherent population uses a strong promoter *hEF1a* to produce the more adherent type of Cadherin (eCadherin).

In this way, each one of these DNA circuits form a simple *feed-forward loop* within living cells, enabling them to gain new, synthetic identities of "weakly adherent" and "strongly adherent." Before I show next how these two new populations can recreate the self-segregation behavior seen in the Morpheus simulation, see the below representation of these DNA circuits. To give the abstract circuit diagrams above little more tangibility in the reader's mind, Fig4-4a shows the weakly adherent cell's DNA circuit as a *virtual circular plasmid* and Fig4-4b, c zooms into this plasmid until individual DNA bases A, C, T, G are shown. The actual DNA circuit in solution looks more similar to this circular plasmid than the representation on Fig4-3.

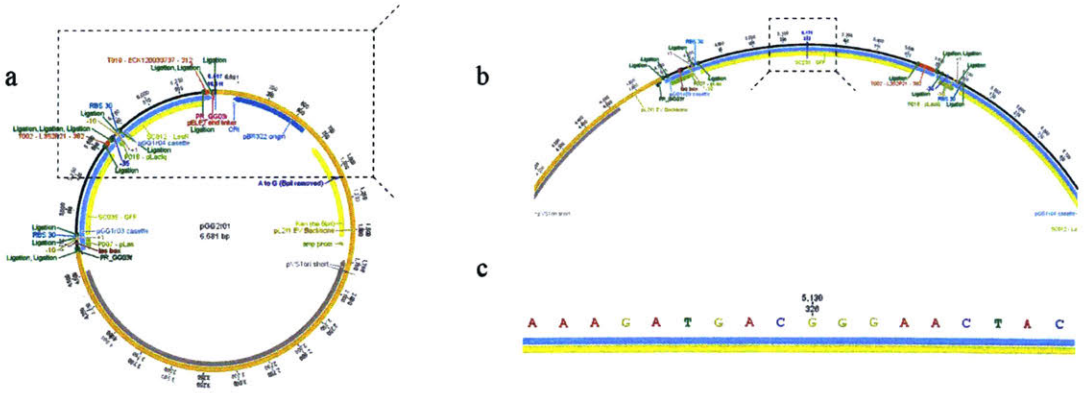


Figure 4-4: (a) Virtual DNA circuit of the weakly adherent cells prepared on *Geneious*. Dashed rectangle outline the region shown in more detail on (b), on which another dashed rectangle outlines the region shown in more detail on (c), where individual DNA bases A, C, T, G can be seen. *Prepared with Geneious 7.1.9*

These DNA circuits are then inserted into the biological cells shown below for them to execute the encoded generative rule and thereby self-organize into two concentric layers[1] [Fig4-5].

[1]The data above is from the mixing of two established cell lines HEK and CHO, mimicking the strongly adherent and weakly adherent cells, respectively, thanks to their native surface protein expression. This experiment is a proxy for the circuit diagrams above since the imaging of that original experiment has failed due to microscope hardware issues.

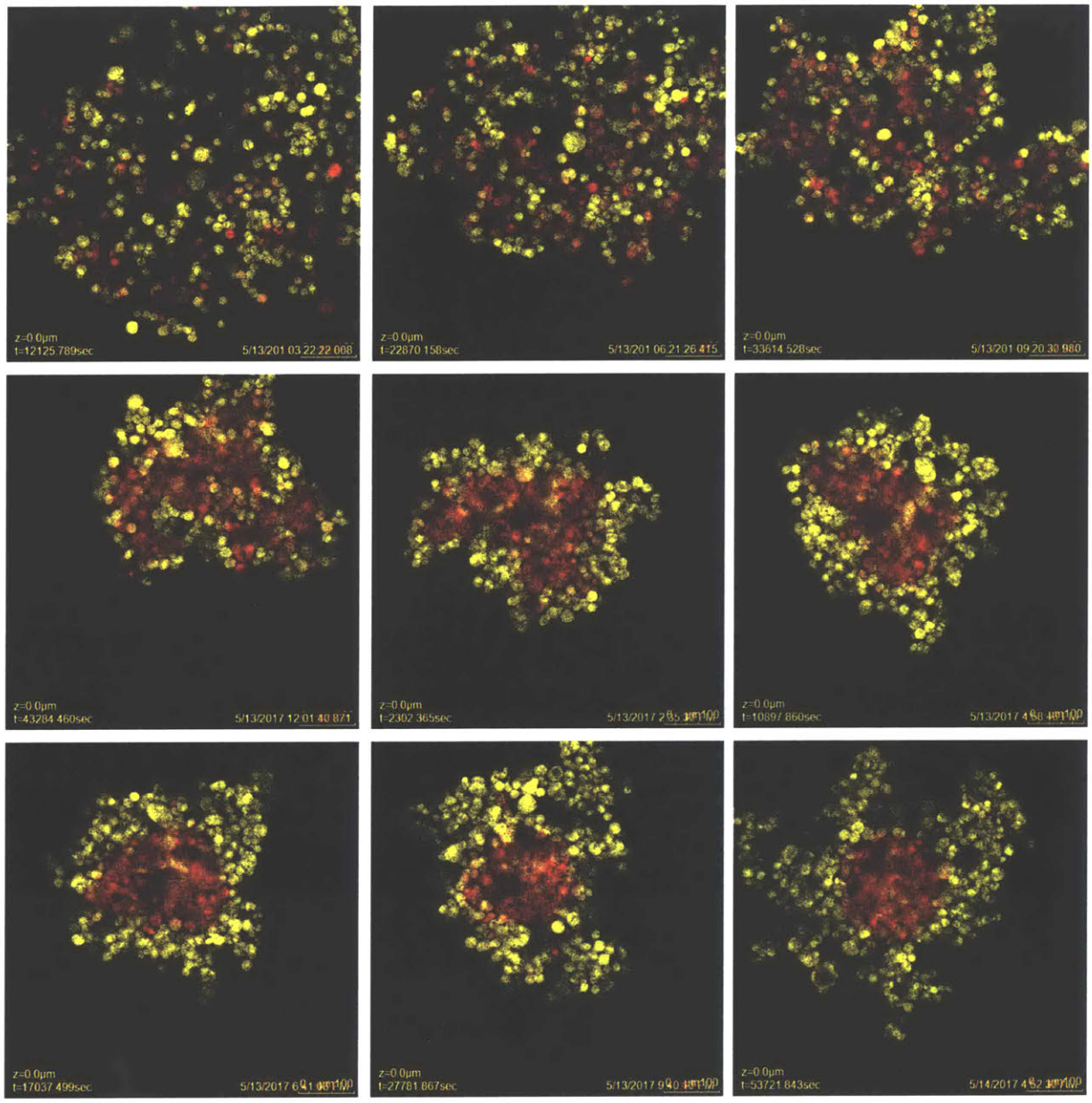


Figure 4-5: The differential cadherin expression enable a well-mixed spherical cellular aggregate to self-organize into two concentric layers that are the 'core' and the 'crust' of the spheroid. *Image credit: Jesse Tordoff*

In this way, differential adhesion function enables a spheroid of cells to develop the notion of a core and crust. Such a center-wise polarization is important because it enables the crust to further polarize and create a FACETS' 6 faces, which play an important role in developing the assembly cues for FACETS to further self-assemble. In the next section, let us dive

deeper into one of the two morphogenetic functions that is at the core of this patterning step: Turing patterns[2]. Turing patterns where one can convert a set of generative rules into a DNA circuit in the context of another morphogenetic function: Turing Patterning.

4.2 Turing Patterns with synNotch and Phloretin

The notion of Turing patterns were introduced by Alan turing in his foundational paper *The Chemical Basis of Morphogenesis* [14] to describe the development of intricate spot and stripe patterns found in nature. The main thesis of Turing was that such patterns are formed by the amplifications of the initial miniscule, local fluctions in an apparently homogeneous group of cells into global heterogeneity. More precisely, Turing describes this phenomenon as a dynamic balance in a system where a self-activating "short-range" morphogen also activating another "long-range" morphogen, which then acting as a repressor shutting down its own activator. Since the activator is a short-range molecule, meaning diffusing slow, and the repressor is a long-range molecule, meaning diffusing fast, such an interaction motif results in the initially strong activation points to get even more activated, whereas its periphery to get even more repressed. Therefore, depending on the relative diffusion rates between these two molecules, the system generates either spot or stripe patterns. Due to this dynamism, contrary to most generative rules we have discussed so far, the most appropriate formal representation of a Turing Pattern function is a finite-state machine diagram [Fig4-6].

[2]The other morphogenetic function that is at the core of this patterning step is the Band-Pass Filter, as discussed in 3.2

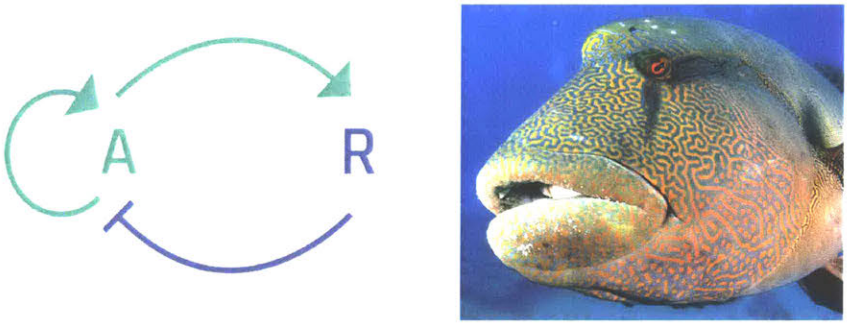


Figure 4-6: the best syntax to represent the generative rules for Turing patterning is a finite-state machine diagram, instead of if-then statements. The reason for this is that in a Turing Patterning system, there is an ever present dynamic balance between activation and repression. Therefore, a rigid if-then statement falls short in representing this dynamic behavior. Later in this section, I introduce how such a finite-state diagram can be converted into a DNA circuit for biological cells to execute the generative rule for Turing patterning.

Turing initially introduced this as a theoretical model and did not propose a specific biological or chemical experimental set up to back up his hypothesis. While having stayed dormant following the years it was initially proposed, in recent years, Turing's this hypothesis gained prominence with several scientists trying to recreate this system in actual chemical and biological systems. Whereas Turing pattern demonstrations were recently made in chemical [30] and bacterial systems [31], so far, nobody has been able to demonstrate this system in a mammalian system. Motivated by this challenge as well as the necessity for further polarization on FACETS' crust, I set out to build the parts one may need for creating Turing patterns in mammalian cells, in collaboration with a post-doctoral fellow in the Weiss Lab.

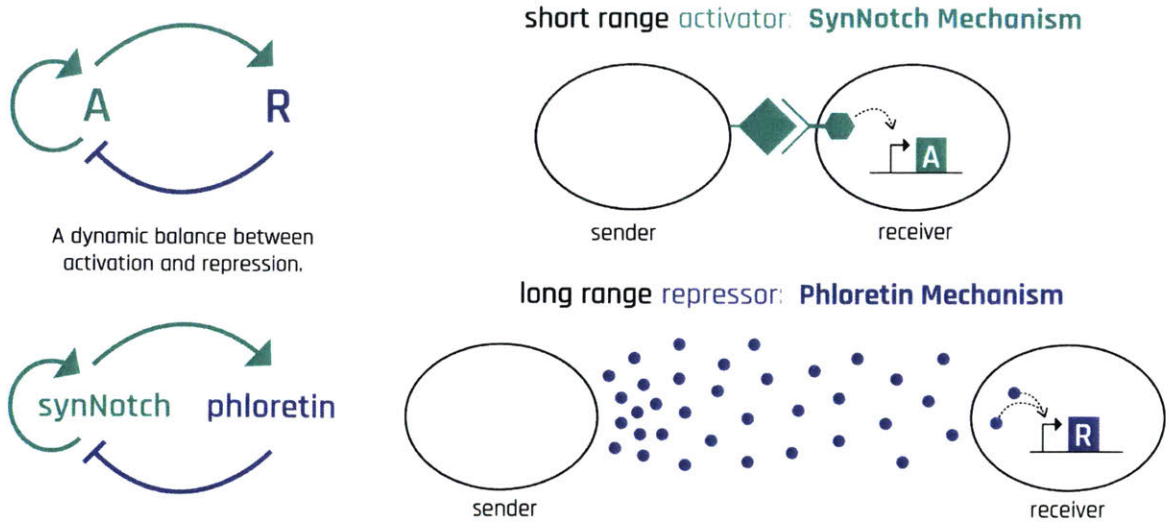


Figure 4-7

For this purpose, as our slow diffusing activator molecule, we decided to use the contact-based cell-to-cell communication system synNotch, and as our fast diffusing repressor molecule, we decided to use a diffusion based cell-to-cell communication system Phloretin [Fig4-7]. SynNotch is the synthetic analog of the native Delta-Notch signaling system [32], and Phloretin is a molecule isolated from the apple tree [33]. Using these two synthetic cell-to-cell communication systems, we designed a DNA circuit [Fig4-8] that captures the generative rules required for Turing patterning.

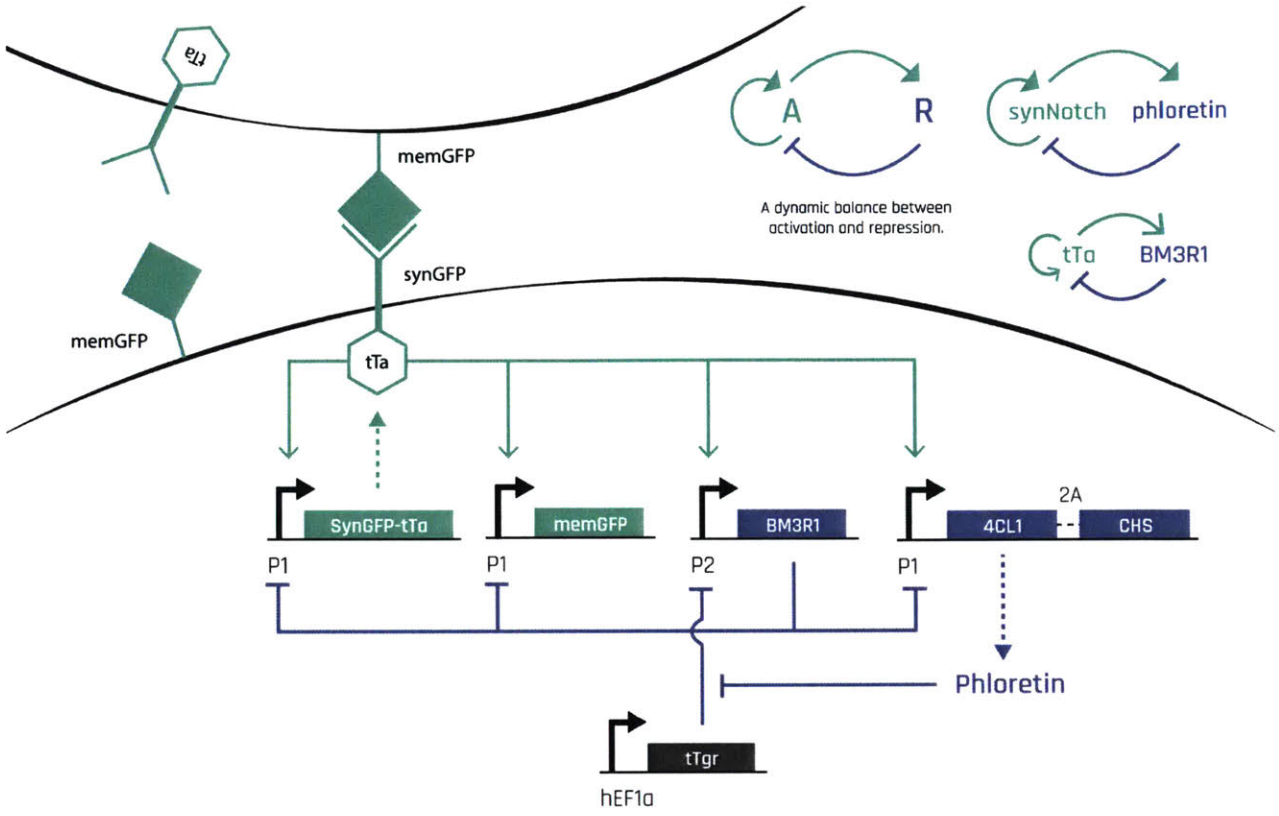


Figure 4-8

In this circuit, once the short-range synthetic cell-to-cell communication receptor *synGFP* gets in contact with its ligand *memGFP*, the receptor's intercellular signalling domain tTa gets cleaved and activates the promoters P1 and P2 (see 4.3 for more information on these proteins). Since P1 is responsible for making more of these synGFP and memGFP parts, such an activation constitutes a positive feedback loop. And since P2 is responsible for producing the repressor BM3R1 and the Phloretin molecule, such activation also constitutes a negative feedback. In this way, the short-range activator activating itself and activating its repressor requirements are thereby met. It is important to note that for the repressor BM3R1 to be produced, activation from tTa alone is not adequate; presence of Phloretin is also necessary (see the double repression path shown on the diagram). In this way, presence of Phloretin

activates the repressor, also fulfilling the long-range repressor requirement of a Turing pattern system.

After having sketched out the DNA circuit for a generative rule in this way, before constructing the circuit, we first needed to make sure that all the pieces are working. In the next section, I demonstrate individual parts of such a genetic circuit can be isolated and tested their functionality. More specifically, I will be focusing the artificial short-range cell-to-cell communication system synNotch.

4.3 A Synthetic Cell-to-cell Communication Framework: synNotch

synNotch is a synthetic cell-to-cell communication molecule that enable one cell to transmit a signal to the other one via mechanical contact. Such signaling occur between a "sender cell," which expresses a ligand protein on its cell surface such as the memGFP, and a "receiver cell," which expresses a receptor protein on its cell surface such as the synGFP.

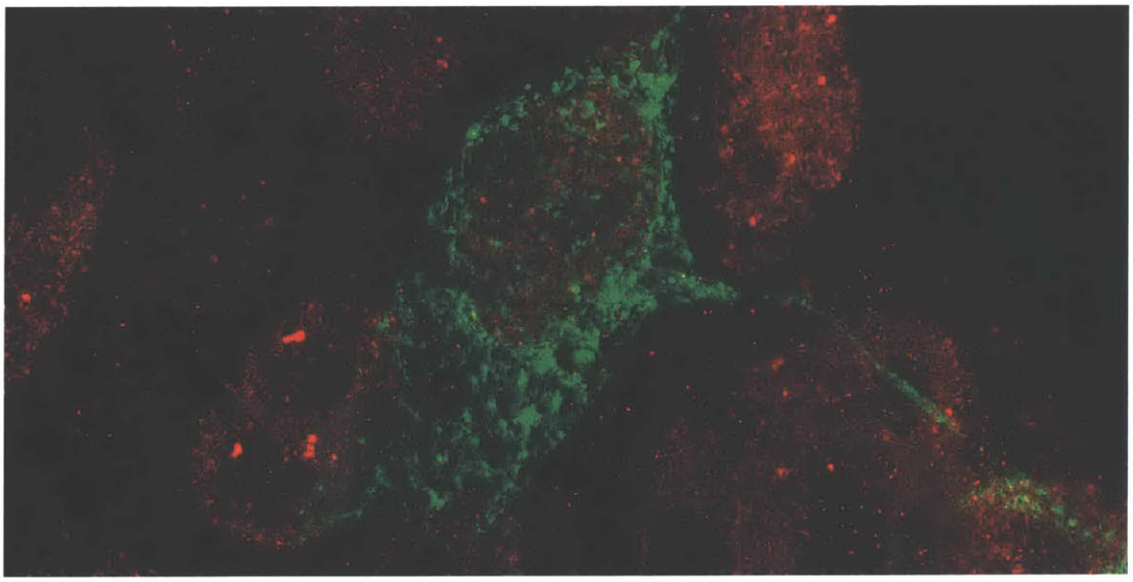


Figure 4-9: Green stain indicating the position of the signal receptors on a cell's surface. Image acquired in collaboration with Dan Oran, Boyden Lab, MIT.

A given cell can have both of these molecules on its cell surface, as is the case on the cells participating Turing patterning shown on Fig4-8. However, to test the functionality of synNotch communication, we need to create a more straightforward set up than the one in Turing patterning case, using the rather simple DNA circuit shown below on Fig4-10.

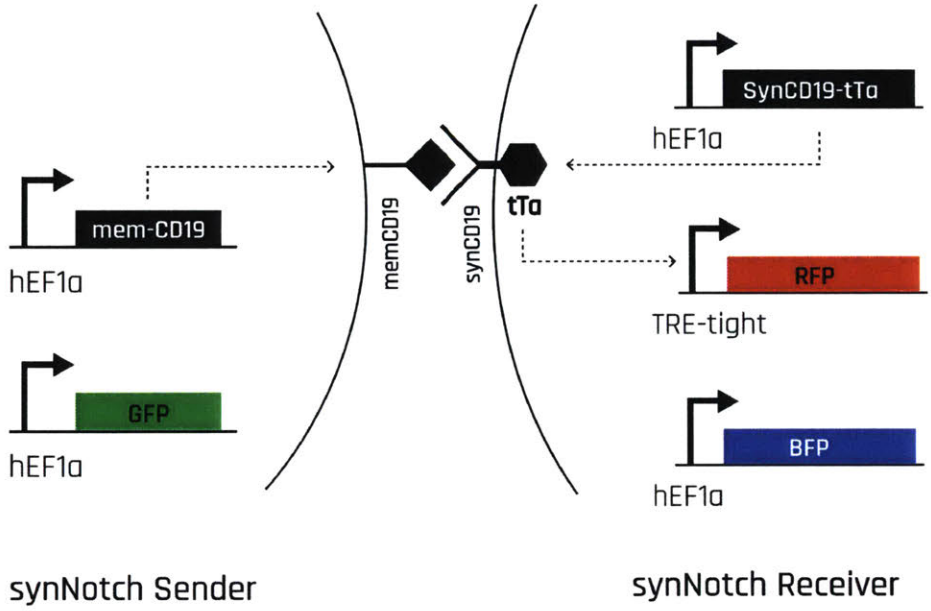


Figure 4-10: co-culture, native Notch domain is still used for the protein to travers the cellular membrane, the receptor module is switched with a GFP receptor and the intercellular domain is switch with a — Interactions between a sender and a receiver cell forms the simple if-then statement below, which can be encoded in a DNA circuit. synGFP synCD19 different

This synthetic DNA circuit enables both the sender and the receiver cell to constantly produce their respective communication molecules on the cell surface from a constitutive promoter such as the hEF1a, which I had also used in the Cadherin circuit [Fig4-3]. When these communication molecules, namely sender's ligand protein *memCD19* and the receiver's receptor protein *synCD19*, come in contact, the mechanical thrust they create cleaves the receptor protein's intercellular domain, which is called *tTA* (tetracycline-controlled transactivator). Such a communication is an example of the mechanical local cues one cell can receive from its environment to compute an output. The generative rule the

cell uses to process such a local signal is a simple one and can be expressed via an if-then syntax:

If *memX and synX-y in comtact*

Cleave y from synX-y

Produce output

Else

Keep synX-y intact

At this point, it is important to remember the nature and role of such generative rules, as postulated earlier in 2.2.1: *a generative rule is a tool for us to represent a computational behavior, rather than a tool for a cell to use to compute.* From the cell's point of view, the tTa's cleavge is purely incidental; it's an artifact of the transmission of forces when atoms on the receptor protein and the sender protein bind together. The sole motivation of these atoms is to become a more stable and expend less energy, not to cleave a protein domain that they are not even aware of. Yet they do. And now it is this cleaved intercellular protein domain's, tTa's, turn to find a new spatial configuration where it can again be stable, instead of hurling around in the cytoplasm. Such a configuration happens to exist within the nucleus, on a piece of DNA molecule, within the proximity of a protomoter sequence called *Tre-Tight.* Once our tTa finds and binds to this promoter region, this spatial interaction results in other proteins to join in, resulting in the expression of a gene downstream of the Tre-Tight promoter. Expression of this gene creates a protein that constitutes the output of such a signaling activity. In this way, without even being aware, tTa plays a critical role in transmitting a message from the cell's surface exposed to the environment, into its nucleus, where this message culminates in an output by interacting with the DNA.

In the context of our DNA circuit above, this output is a Red Fluorescent Protein (RFP). By measuring the change in the production of this RFP protein, we can measure how successful

the communication is. The Green Fluorescent Protein (GFP) and the Blue Fluorescent Protein (BFP) constitutively expressed in the sender and receiver cells, respectively, is for us to be able to see these cells under the microscope, as well as quantify this synthetic cell-to-cell signaling behavior as shown on the plot below [Fig4-11b]. This plot shows the change of RFP in a receiver cell as a function of the amount of DNA circuit this single cell receives. The y axis shows the amount of RFP in MEFL units (molecules of equivalent fluorescein), and the x axis shows the relative number of DNA circuits in a given cell based on the amount of BFP produced in this cell.

Data presented below is acquired in a co-culture experiment designed to accurately test the functionality of such a synthetic communication system. In this co-culture experiment, there are four major "wells" (i.e., an isolated chamber on a multi-chamber tissue culture plate), in each of which, a different communication condition was tested [Fig4-11a]:

- **condition #1** is the experimental setup, where the sender and receiver cells are mixed together in 1:1 ratio in a single well. Under this condition, we expect to see the second highest RFP output following the positive control condition.

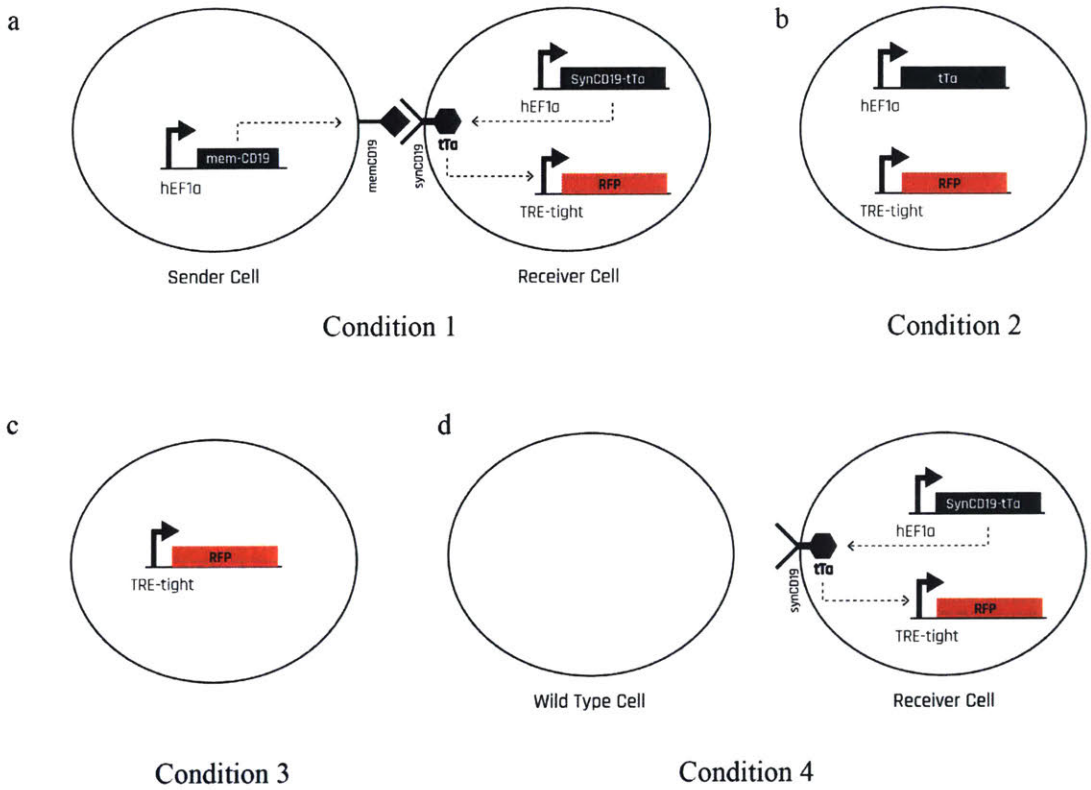
- **condition #2** is the positive control condition, where only one type of cell is cultured within a well. This cell includes a constitutive expression of the tTa, as well as the tTa responsive promoter Tre-Tight and a RFP protein downstream of it. Under this condition, we expect to see the highest RFP output because no matter how successful the cell-to-cell communication is, it will never result in the cleavege of as much tTa as constitutively expressing it. In this way, this "ceiling" condition shows us what is the maximum RFP a cell can produce.

- **condition #3** is the first negative control, or "leakiness control," where, again, only one type of cell is cultured within a well. This cell does not include any tTa, neither as the intracellular domain of the receptor protein, nor constitutively produced. It only

includes the tTa responsive promoter Tre-Tight and a RFP protein downstream of it. Under this condition, we expect to see the lowest RFP output because no matter how unsuccessful the cell-to-cell communication is, there will always be more tTa than *none* within the cell. This is due to "leakiness" of the receptor protein (i.e., the tTa domain gets cleaved even if there is neighboring sender cells in the environment), which we will account four in the next condition. In this way, condition #3 constitutes a "floor," showing the minimum amount of RFP a cell can produce, which is bigger than zero, this time due to the "leakiness" of the promoter Tre-Tight (i.e., Tre-Tight gets activated even if there is no tTa in the environment).

- **condition #4** is the second negative control, or "leakiness control," where, wild type cells, which do not have the sender molecule on their cell surface, and the receiver cells are mixed together in 1:1 ratio in a single well. Under this condition, we expect to see the second lowest RFP output because there are no senders cells around to transmitting a message for receiver cells to then take it and use it to create the RFP output. Nevertheless, there is still some RFP observed in this condition due to the leakiness of the receptor protein, which is caused by the other mechanical forces surrounding it, as well as the leakiness of the Tre-Tight protomer, which is negligible next to the former. Such leaky behavior is inevitable in a biological system and it is considered to be *noise*, for which we normalize in our rigorous quantitative analyses.

Data presented in the figure below confirms our hypothesis above and it shows an above 10 fold difference between the experimental setup (condition #1) and its direct negative control (condition #4). Each one of these conditions is represented with a different colored line on Fig4-11b, corresponding the condition illustrations shown on Fig4-11a.


a
mem-CD19
hEF1α
memCD19
synCD19
tTα
SynCD19-tTα
hEF1α
RFP
TRE-tight
Sender Cell
Receiver Cell
Condition 1
b
tTα
hEF1α
RFP
TRE-tight
Condition 2
c
RFP
TRE-tight
Condition 3
d
Wild Type Cell
SynCD19-tTα
hEF1α
tTα
synCD19
RFP
TRE-tight
Receiver Cell
Condition 4

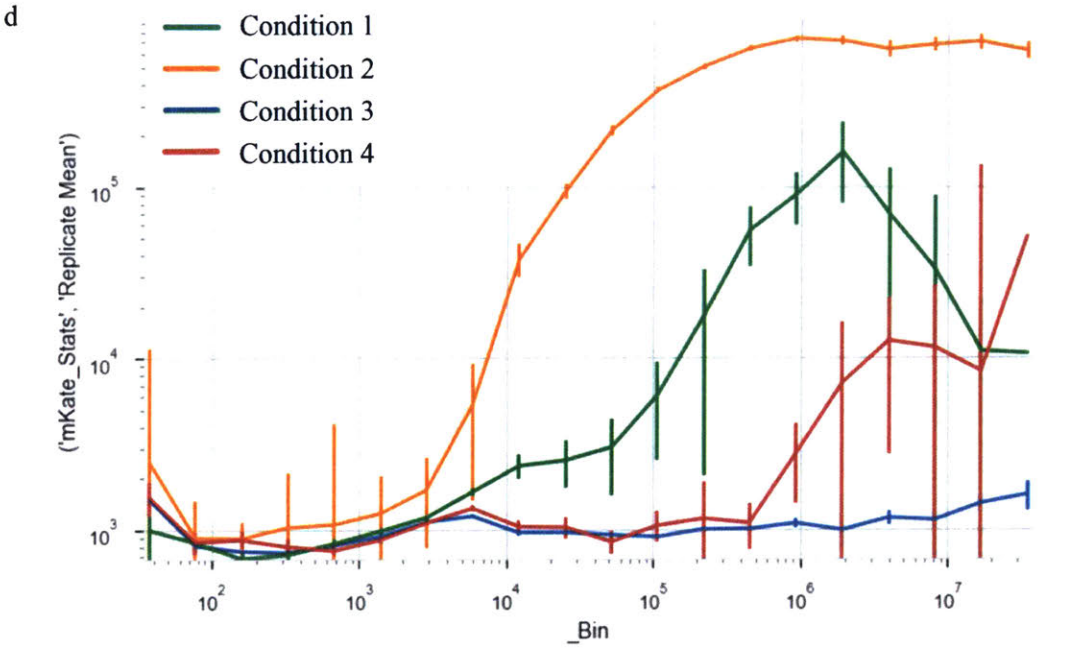

d
Condition 1
Condition 2
Condition 3
Condition 4
('mKate_Stats', 'Replicate Mean')
10^5
10^4
10^3
10^2
10^3
10^4
10^5
10^6
10^7
_Bin

Figure 4-11

Below are confocal microscopy images verifying the data presented above [Fig4-12], demonstrating the synthetic cell-to-cell signaling phenomenon unfolding on a FACETS precursor sphere. Fig4-12a is the experimental setup under condition#1 where the blue receiver cells "light up" in red to show that they receive the message sent by the green sender cells upon contact. Fig4-12c breaks down this image into different channels. Looking at condition #4, the direct negative control of this setup, shown on Fig4-12b, we can see not as many cells expressing RFP, due to the lack of the senders.

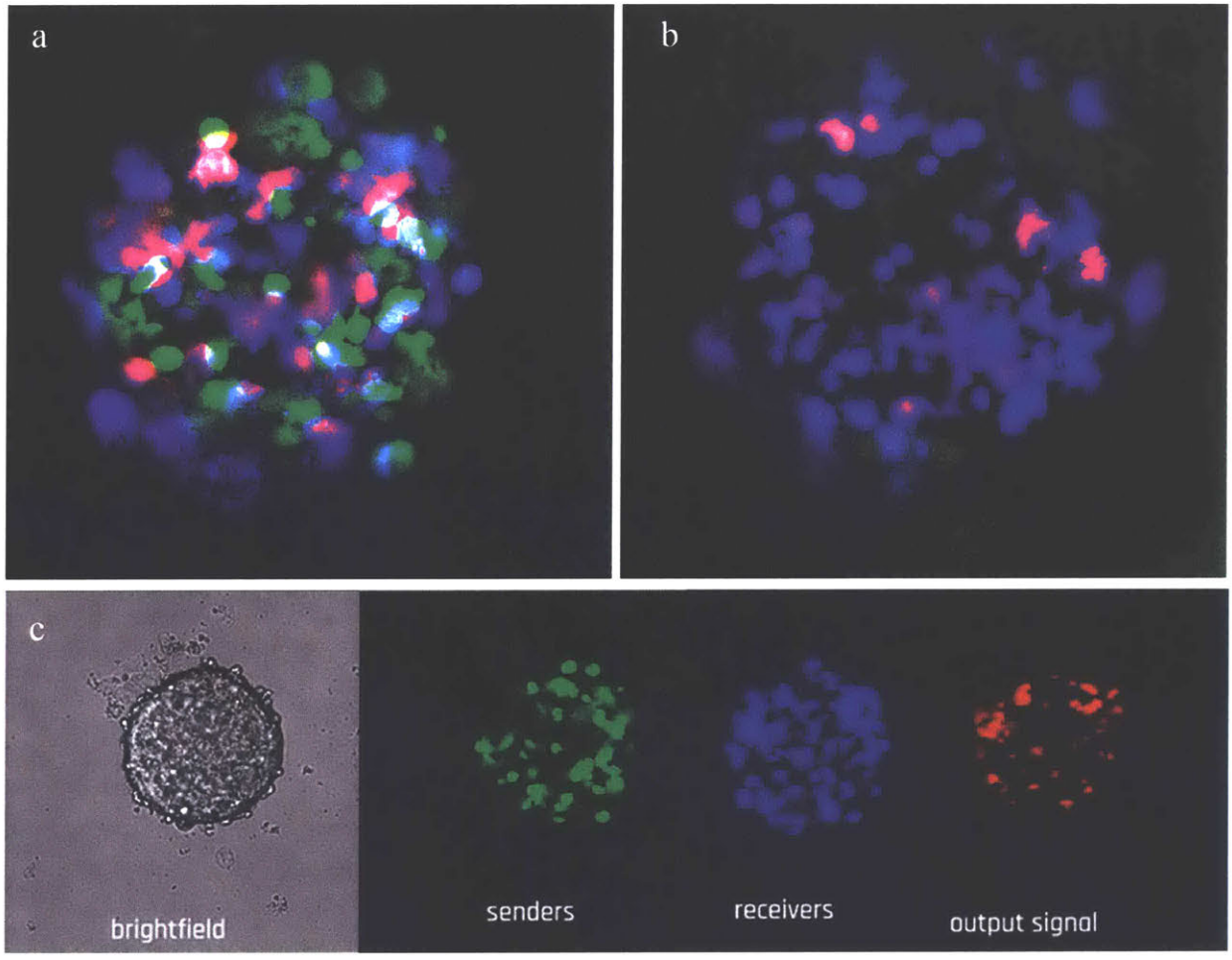


Figure 4-12: not all cells fluoresce because some have not received any of the DNA circuit

Such a synthetic cell-to-cell communication system can act as a morphogenetic function alone, and furthermore, can be combined with other morphogenetic functions, such as

differential adhesion. As an example, below are confocal microscopy images showing how we can merge the synthetic cell-to-cell communication module synNotch with the synthetic differential adhesion module Cadherins, which were previously introduced in 4.1. In this set up below, synthetic cell-to-cell communication unfolds in a construct that also undergoes differential adhesion, as opposed to the well-mixed co-cultures above. The core of the self-stratified construct below consists of the sender cells and its crust is made up of receiver cells. A more advanced version of this set up, where the differential adhesion is mediated by synNotch based cell-to-cell communication can be found in *Programming self-organizing multicellular structures with synthetic cell-cell signaling* [3].

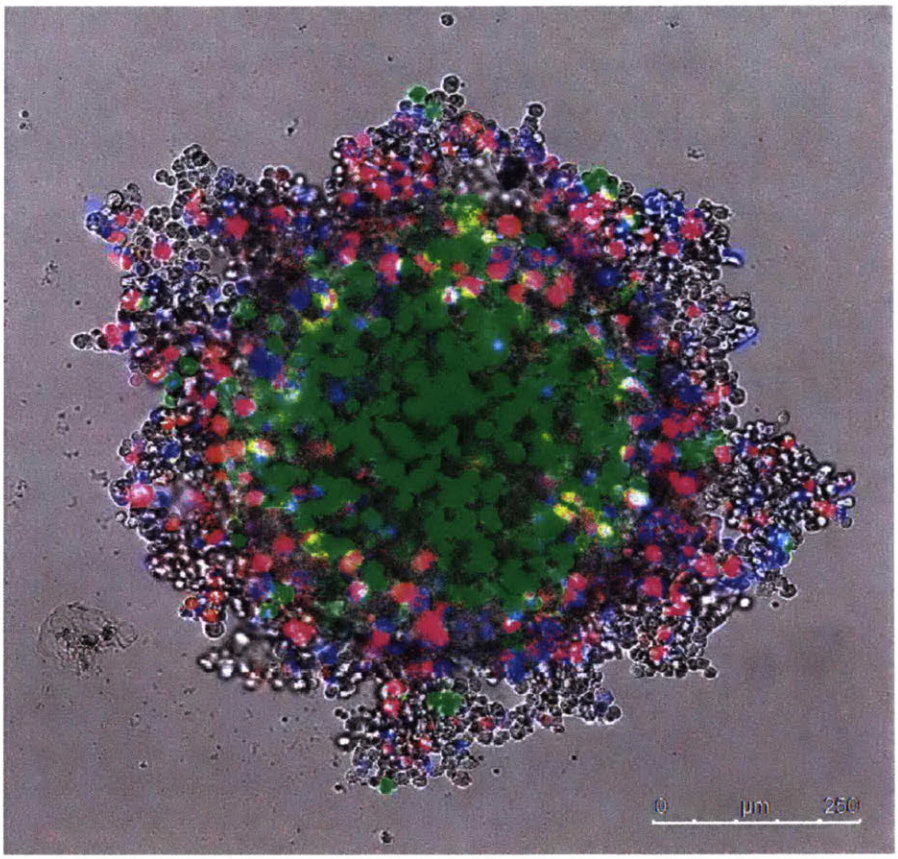


Figure 4-13: not all cells fluoresce because some have not received any of the DNA circuit

Finally, in this section, I introduced an example of how an individual genetic module of a complex synthetic genetic circuit, such as the synNotch module of the Turing patterning circuit

[Fig4-8], can be isolated to test and further improve its functionality in a controlled setup. Furthermore, I showed how such a genetic module can be further combined with another modules to create novel morphogenetic functions, such as synthetic cell-to-cell communication mediated autonomous differential adhesion (Toda et al.).

The select set of results I discussed above is only a small portion of the many experiments I conducted in order to improve the synNotch performance in mammalian Human Embryonic Kidney (HEK) and Chinese Hamster Ovary (CHO) cells. I documented many of the protocols I used during these experiments on an online platform, which can be found at `http://www.synbioprotocols.info`. Fig4-14 shows a screen shot from its main page featuring some of the protocols used for DNA circuit construction.

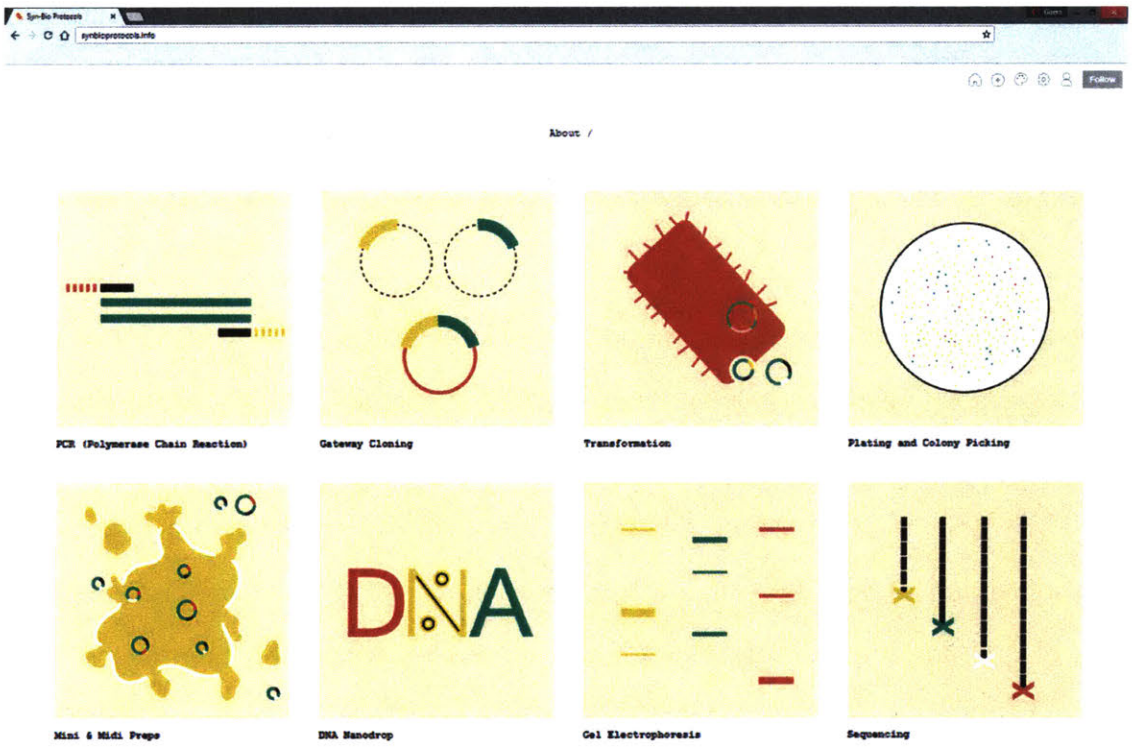


Figure 4-14

In order to give the design audience a better idea of the synthetic biology environment, Fig4-15 shows a typical work space in which I have carried out some of these mammalian cell culture protocols.

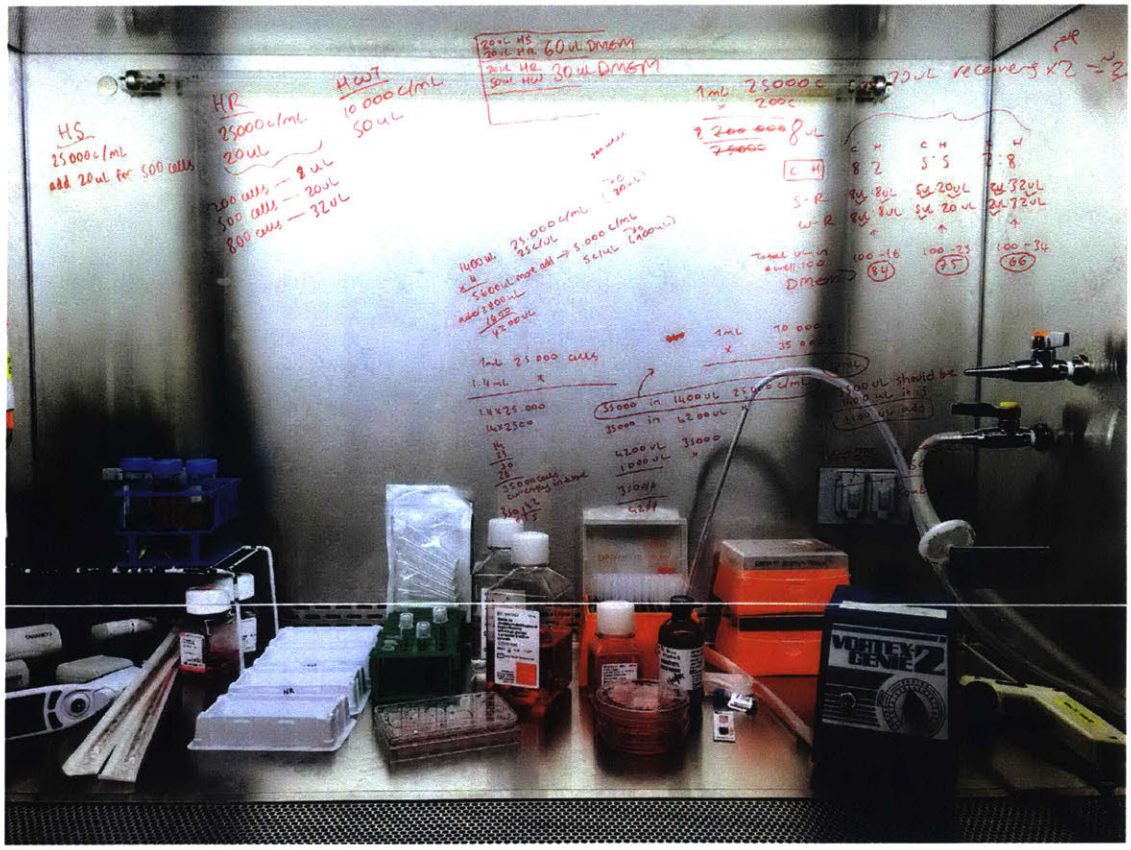


Figure 4-15

In terms of future work, given how ambitious the genetic algorithm of a FACETS, much remains to be done. The most obvious next step is to combine the synNotch module with the Phloretin module and test the Turing pattern function through the genetic circuit proposed above. Then, once the morphogen diffusion rates of this Turing system tuned such that it is able to generate a single spot on a FACETS crust, it can then be further combined with the rest of the morphogenetic functions, such as the band-pass filter and the time-keeper oscillator, for the composition of the full FACETS synthetic genetic regulatory network.

In this Results and Discussion Chapter, I showed how the interplay between activators, repressors, promoters and genes forming the back bone of a synthetic genetic regulatory network, or a DNA circuit, can enable us to introduce novel behaviors into living cells. Through such synthetic gene regulatory networks made up of DNA, we can create combinatorial

logic gates and feedback loops inside living cells, and thereby introduce them the artificial generative rules necessary for their development into architectures of our desire. In this way, biological cells can become programmable building blocks with which to fabricate higher-order constructs.

Chapter 5

Conclusions

In this thesis, I introduced ASSEMBLE, where the fabrication process starts with the designer uploading the 3D model of their structure of interest into a global-to-local structural compiler, which then "discretizes" this structure into a lattice made up of morphogenetic building elements (i.e., multicellular living structures of our own design such as a FACETS) attached to one another via embodied assembly cues. Accordingly, the structural compiler determines which assembly cues (e.g., protrusion number and glue type) needed on each morphogenetic building element making up the target structure, as well as the generative rules needed for these elements to be grown out of single cells. These rules are then inserted into single cells' genomes in the form of genetic circuits made up of DNA to facilitate their programmed growth into morphogenetic building elements. One cell and one DNA circuit encoding the rules for a given morphogenetic building element is sufficient to grow millions of this building element —thanks to the cell's ability to exponentially self-replicate. This is one of the unique hallmarks of biofabrication: its building blocks, cells, provide an infinite supply of building material. And what separates ASSEMBLE from the other architectural biofabrication methods is its exploiting this infinite source of building material such that it autonomously builds itself into a building element without the need for external machinery such as a 3D printer of a

mold. Furthermore, what separates ASSEMBLE from the other synthetic morphogenetic approaches is its concurrent use of *top-down discrete assembly*, wherein the assembly cues on the resulting morphogenetic building element enable it to recognize and selectively bind to the similar building elements, scaling up into the order of centimeters. In this way, by harnessing both the hallmarks of both biology (i.e., biological cells' capacity for self-replication, self-organization, and responsiveness to the environment) and architecture (i.e., its tradition of discrete assembly via joints with specificity) I introduce a novel fabrication process wherein micron-scale single cells self-replicate and self-organize into millimeter-scale morphogenetic building elements, which then, by interacting with one another, self-assemble into an even higher-order, centimeter-scale target architectures. In this way, ASSEMBLE enable the fabrication of centimeter-scale structures out of single cells.

There are significant implications for architects in having the power to design a precursor cell that autonomously self-constructs into a building element with prespecified material and structural specifications. Even though the state of the art for controlling morphogenetic processes limits the dimensions of these individual elements to millimeter-scale, when coupled with discrete assembly strategies as introduced in this thesis, these building elements can form self-constructing structures that span centimeters and even meters.

5.1 Contributions

Beyond being intellectually novel and interesting, the ability to encode the building specifications directly into the building material itself can contribute to architectural fabrication in many ways:

I **Biofabrication can become independent of machine turnover and this reduced fabrication complexity can pave the way to sustainable mass fabrication of cellular architectures.**

At first sight, constructing with self-replicating biological cells, which constitute an infinite material supply with negative carbon footprint, appears to be an inherently sustainable process amenable to mass fabrication. However, one must remember that for this infinite raw material to become useful architectural building blocks, it must somehow acquire shape. How this process occurs is a crucial determinant in how easily a biofabrication process can be scaled into mass-fabrication and how sustainable it would be.

Current architectural biofabrication practices rely on top-down methods (e.g., 3D printing, molding) that rely on external machinery to impose a global shape into the multicellular society. This reliance on external machinery for giving form to a building material that is already extremely capable in generating form from within adds unnecessary layers of complexity into the biofabrication process, making such top-down biofabrication processes unsustainable and unprofitable for mass fabrication. These methods require individual shape-giving equipment to remain in constant use for a single object to be biofabricated; thus, if one needs to fabricate two objects in parallel, they would also need to duplicate the shape-giving equipment. Therefore, on top the initial investment required by any biofabrication process for building and maintaining a laboratory, such top-down biofabrication methods add the extra cost of time, labor, and energy *per building block to be constructed.* In other words, within the current top-down biofabrication practices, the fabrication cost and complexity grows proportional to the number and type of building blocks to be constructed.

Therefore, the primary focus of the ASSEMBLE method is to stop treating biological cells as substitutes for plastic or concrete to mold or 3D print with, and instead start seeing them for what they really are: cells are living computers with the ability to generate form from within. In this way, by focusing on cells' ability to self-organize, ASSEMBLE emancipates the biofabrication process from the wait time of a 3D printer

turnover, or the requirement of designing and fabricating a new mold for every new architectural building block to be constructed. It enables us to construct multiple architectural building blocks simultaneously at no additional time, labor, or energy cost. Such manufacturing scalability is a prerequisite for any fabrication process that claims to be sustainable because sustainability is a measure of cumulative impact; it cannot be judged at the level of a one-off product —no matter how low its carbon footprint or how fast it decomposes when composted.

II **We can create living architectures with integrated material characteristics.**

Beyond the ability to self-organize into diverse structures, biological cells are able to grow a myriad of different materials with diverse characteristics such as bone (brittle and tough) or collagen (elastic and smooth), as well as integrate these materials together at a high resolution. With ASSEMBLE, not only can we specify what structure cells should build but also which parts of the structure should be made of which biomaterials. In this way, we can grow architectures with hybrid material properties that cannot be achieved even by state-of-the-art multi-material 3D printers.

Furthermore, as noted above, living cells are able to sense and respond to a range of external stimuli in very useful ways, as are the higher-order structures they create (e.g., perceptional richness of organisms). In this way we can grow cellular architectures that have an intrinsic ability to respond to their environments, as well as exhibit the emergent properties observed in higher order organisms, such as healing.

III **We can construct with ease in extreme environments (i.e., both remote terrestrial and extra-terrestrial).**

Transportation of the building materials and construction machinery into extreme environments is currently either impossible or very difficult and costly. Spacecrafts leaving the Earth's orbit must keep their weight to an absolute minimum to be able

to escape the gravitational field; cargo planes and ships bound for the Earth's isolated deserts or poles have inadequate capacity to carry the massive infrastructure needed in traditional construction methods to self-sufficiently build in these remote regions.

What if these vehicles instead carried a tube of engineered cells that encode the building instructions necessary to self-replicate and construct themselves into architectural building blocks? These architectural building blocks can be grown in such a way that they express self-assembly cues on their surface, which enable them to be easily assembled into inhabitable structures at the destination. All the cells would require to be kept alive is a miniscule infrastructure, comprised of a tissue culture hood, an incubator, and a refrigerator, which would be a one-time investment.

In this way, humans one day may leave the Earth's orbit with a tube of engineered cells; have them self-replicate and self-construct into architectural building elements during the long space travel; and finally use them to build self-assembling, inhabitable structures in the final extra-terrestrial destination.

List of Figures

Bibliography

[1] Ernesto Andrianantoandro, Subhayu Basu, David K Karig, and Ron Weiss. Synthetic biology: new engineering rules for an emerging discipline. *Mol. Syst. Biol.*, 2:2006.0028, May 2006.

[2] Brian P. Teague, Patrick Guye, and Ron Weiss. Synthetic Morphogenesis. *Cold Spring Harbor Perspectives in Biology*, 8(9):a023929, September 2016.

[3] Satoshi Toda, Lucas R. Blauch, Sindy K. Y. Tang, Leonardo Morsut, and Wendell A. Lim. Programming self-organizing multicellular structures with synthetic cell-cell signaling. *Science*, page eaat0271, May 2018.

[4] Jesse Tordoff and Ron Weiss. Self-organizing multicellular structures designed using synthetic biology. *Nature*, 559(7713):184–185, July 2018.

[5] J. A. Davies and E. Cachat. Synthetic biology meets tissue engineering. *Biochemical Society Transactions*, 44(3):696–701, June 2016.

[6] Patrick Guye, Mohammad R. Ebrahimkhani, Nathan Kipniss, Jeremy J. Velazquez, Eldi Schoenfeld, Samira Kiani, Linda G. Griffith, and Ron Weiss. Genetically engineering self-organization of human pluripotent stem cells into a liver bud-like tissue using Gata6. *Nature Communications*, 7:10243, January 2016.

[7] G A Holt, G Mcintyre, and E Bayer. Optimizing biomass blends for manufacturing molded packaging materials using mycelium. *of association for the advancement of ...*, 2012.

[8] Ginger Krieg Dosier. *Methods for making construction material using enzyme producing bacteria.* Google Patents, May 2014.

[9] Abir U. Igamberdiev and Nikita E. Shklovskiy-Kordi. Computational power and generative capacity of genetic systems. *Biosystems*, 142-143:1–8, April 2016.

[10] Jamie A. Davies. Machines for living in: Connections and contrasts between designed architecture and the development of living forms. *Architectural Research Quarterly*, 20(01):45–50, March 2016.

[11] Scott F. Gilbert and Sahotra Sarkar. Embracing complexity: organicism for the 21st century. *Developmental dynamics*, 219(1):1–9, 2000.

[12] John Conway. The game of life. *Scientific American*, 223(4):4, 1970.

[13] A discussion of the Game of Life. Technical report, Stanford University.

[14] A. M. Turing. The Chemical Basis of Morphogenesis. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 237(641):37–72, August 1952.

[15] Przemysław Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants.* The virtual laboratory. Springer, New York, first soft cover printing edition, 1996. OCLC: 845212499.

[16] Adam Runions, Miltos Tsiantis, and Przemyslaw Prusinkiewicz. A common developmental program can produce diverse leaf shapes. *New Phytologist*, 216(2):401–418, October 2017.

[17] Allen R. Sanderson, Robert M. Kirby, Chris R. Johnson, and Lingfa Yang. Advanced Reaction-Diffusion Models for Texture Synthesis. *Journal of Graphics Tools*, 11(3):47–71, January 2006.

[18] Shigeru Kondo and Takashi Miura. Reaction-diffusion model as a framework for understanding biological pattern formation. *Science*, 329(5999):1616–1620, September 2010.

[19] Hillery C. Metz, Marie Manceau, and Hopi E. Hoekstra. Turing patterns: how the fish got its spots: News and views. *Pigment Cell & Melanoma Research*, 24(1):12–14, February 2011.

[20] Jeremy M. Berg, John L. Tymoczko, Gregory J. Gatto, and Lubert Stryer. *Biochemistry.* W.H. Freeman & Company, a Macmillan Education Imprint, New York, eighth edition edition, 2015.

[21] Bill Hillier and Julienne Hanson. *The social logic of space.* Cambridge Univ. Press, Cambridge, repr edition, 2005. OCLC: 255563984.

[22] Bill Erickson and Tony Lloyd-Jones. Experiments with settlement aggregation models. *Environment and Planning B: Planning and Design*, 24(6):903–928, 1997.

[23] Scott Camazine, editor. *Self-organization in biological systems.* Princeton studies in complexity. Princeton University Press, Princeton, N.J, 2001.

[24] Andrew P. Smith. An investigation of the mechanisms underlying nest construction in the mud wasp Paralastor sp. (Hymenoptera: Eumenidae). *Animal Behaviour*, 26:232–240, February 1978.

[25] H. Spemann and Hilde Mangold. über Induktion von Embryonalanlagen durch Implantation artfremder Organisatoren. *Archiv für Mikroskopische Anatomie und Entwicklungsmechanik*, 100(3-4):599–638, September 1924.

[26] Erik D. Demaine, Martin L. Demaine, Sándor P. Fekete, Mashhood Ishaque, Eynat Rafalin, Robert T. Schweller, and Diane L. Souvaine. Staged self-assembly: nanomanufacture of arbitrary shapes with O(1) glues. *Natural Computing*, 7(3):347–370, September 2008.

[27] Hanan Samet. The Quadtree and Related Hierarchical Data Structures. *ACM Computing Surveys*, 16(2):187–260, June 1984.

[28] Elise Cachat, Weijia Liu, Kim C. Martin, Xiaofei Yuan, Huabing Yin, Peter Hohenstein, and Jamie A. Davies. 2- and 3-dimensional synthetic large-scale de novo patterning by mammalian cells through phase separation. *Scientific Reports*, 6(1), August 2016.

[29] Jörn Starruß, Walter de Back, Lutz Brusch, and Andreas Deutsch. Morpheus: a user-friendly modeling environment for multiscale and multicellular systems biology. *Bioinformatics*, 30(9):1331–1332, May 2014.

[30] N. Tompkins, N. Li, C. Girabawe, M. Heymann, G. B. Ermentrout, I. R. Epstein, and S. Fraden. Testing Turing's theory of morphogenesis in chemical cells. *Proceedings of the National Academy of Sciences*, 111(12):4397–4402, March 2014.

[31] David Karig, K. Michael Martini, Ting Lu, Nicholas A. DeLateur, Nigel Goldenfeld, and Ron Weiss. Stochastic Turing patterns in a synthetic bacterial population. *Proceedings of the National Academy of Sciences*, 115(26):6572–6577, June 2018.

[32] Leonardo Morsut, Kole T. Roybal, Xin Xiong, Russell M. Gordley, Scott M. Coyle, Matthew Thomson, and Wendell A. Lim. Engineering Customized Cell Sensing and

Response Behaviors Using Synthetic Notch Receptors. *Cell*, 164(4):780–791, February 2016.

[33] Xavier Duportet. *Developing new tools and platforms for mammalian synthetic biology: From the assembly and chromosomal integration of complex DNA circuits to the engineering of artificial intercellular communication systems.* PhD, Universite Paris Diderot (Paris 7), 2014.