

A THEORY OF LOCAL-TO-GLOBAL ALGORITHMS FOR
ONE-DIMENSIONAL SPATIAL MULTI-AGENT SYSTEMS

A dissertation presented

by

Daniel Yamins

to the

School of Engineering and Applied Sciences
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of

Applied Mathematics

Harvard University
Cambridge, Massachusetts

November 2007

© Copyright by Daniel Yamins 2008
All Rights Reserved

Abstract

A *spatial multi-agent system* is a decentralized system composed of numerous identically programmed agents that either form or are embedded in a geometric space. The agents' computational constraints are spatially local. Each agent has limited internal memory and processing power, and communicates only with neighboring agents. The systems' computational goals, however, are typically defined relative to the global spatial structure.

In this thesis, I develop the beginnings of theory of spatial multi-agent systems, for the simple case of pattern formation in a one-dimensional discrete model. First, I characterize those patterns that *are* robustly self-organizable in terms of a simple "necessary condition on solvability". I then solve the inverse problem, constructing an algorithmic procedure that generates robust local rule solutions to *any* desired solvable pattern. Next, I analyze resource usage and runtime properties of such local rule solutions.

I apply this suite of mathematical techniques to two diverse "global-to-local" problems: the engineering goal of developing a generic "global-to-local" compilation procedure, and the scientific goal of analyzing an embryological development process.

Acknowledgments

I would like to acknowledge the members of the Fontana lab for their support and engagement: Javier Apfeld, Eric Deeds, Jerome Foret, Thomas Kolokotronis, Debora Marks, Catalina Romero, Van Savage, and Nicholas Stroustrup.

I gratefully acknowledge Walter Fontana for paying me, and giving me a great place to work.

I happily thank members of the Nagpal group for their ideas and involvement: Julius Degesys, Radhika Nagpal, Ankit Patel, Ian Rose, Justin Werfel, and Chih-han Yu.

I am deeply indebted to Radhika Nagpal for being a true adviser.

I thank Richard Lewontin, Jake Beal, Gerry Sussman, Zak Stone, Lea Goentoro, Aneil Mallavarapu, James McLurkin, Hal Abelson, Stephen Maturin, Vincent Danos, Jeremy Gunawardena, Michael Rabin, Harry Lewis, Pamela Silver, Salil Vadhan, Chris Sander, Cris Moore, Van Parunak, Eric Smith, Stuart Shieber, Margot Seltzer, Harold Morowitz, Sam Lipoff, and Stu Lipoff for their very valuable scientific contributions and their lively interest.

I especially thank Eric Deeds, Catalina Romero, Thomas Barnet-Lamb, and Ankit Patel for their tremendous scientific insight, numerous fun and relevant contributions, and unparalleled enthusiasm.

I owe Sam Lipoff, Roxanne Yamins, Chenling Zhang, Yanjun Wang, HPAIR, Vikas Prabhakar, Catalina Romero, Lea Goentoro, Jack Aubrey, Jennifer Haraguchi, Larry Abramson, Debbie Marks, Maria Kim, Oliver Jacob, Ari Lamstein, and Joanna Yeo, for putting up with me and trying to keep my mind on other things.

Tian Mayimin has been my constant.

Above all, I thank my parents Janice and David Yamins, who have been my constant and most ardent supporters, my most respected advisors, and greatest sources of inspiration.

Contents

1	A Simple Model	7
1.1	Static Configurations	7
1.2	Local Rule Dynamics	9
1.2.1	Timing Models	10
1.3	Patterns	11
1.3.1	Pattern Properties	12
1.3.2	Admissible Sizes and Pattern Growth	14
1.4	Solutions	14
1.5	Illustrative Examples	16
1.6	Related Work	19
2	Local Checkability: A Necessary Condition	20
2.1	Local Checkability: A Necessary Condition	20
2.2	Local Checks Are Part Lists	23
2.2.1	Generating Configurations	23
2.3	LCSs are Graphs	25
2.3.1	Meaning of the Graph Structure	26
2.3.2	Admissible Sizes	31
2.3.3	Pattern Growth	33
2.3.4	Summary	36
2.4	Local Checkability and Formal Languages	36
3	Local Checkability is Sufficient: Generic Constructions	39
3.1	Single-Choice Patterns: A Gradient Algorithm	39
3.2	The Naive Backtracking Algorithm	40
3.2.1	Distributed, Virtual, Robust Heads	43
3.2.2	A Turing Operator Basis	44
3.2.3	Other Orderings	44
3.2.4	Viewing F_{Θ}^O on $G(\Theta)$	45
3.2.5	The Effect of the Ordering Function O	47
3.3	Run-Time Analysis of Naive Backtracking Algorithm	47
3.3.1	Several Examples	50
3.3.2	The Maximally Futile Subgraph	52
4	Faster Algorithms and Pattern Classification	57
4.1	A Linear Time Algorithm	57
4.1.1	The Strategy	58
4.1.2	Single Terminus Graphs	60
4.1.3	The General Case	63
4.2	Locally Patchable Patterns	66
4.2.1	Dynamic Local Patching	69
4.2.2	Getting Almost Correct	71
4.3	Pattern Complexity Classes	75

5	Lower Bounds and Measure of Global Order	77
5.1	Elementary Arguments	78
5.2	Discrete Fourier Analysis of Configurations	79
5.3	\mathcal{P} as a Slowly-Growing Order Measure	82
5.3.1	84
5.4	A More Robust Fourier Transform	88
5.5	Graph-Based Computation of \bar{P}	91
5.6	A Frequency Independent Approach	95
6	The Radius-State Tradeoff	99
6.1	The Static Tradeoff	99
6.1.1	Example: The Coordinate Pattern	99
6.1.2	Trading Radius for State	101
6.1.3	Trading State for Radius	103
6.2	The Dynamic Trade-Off	107
6.2.1	Isolated Turing Head Algorithms	107
7	Pattern Space Analysis	111
7.1	$\mathbb{D}(r, m)$ and the DeBruijn Graph	111
7.2	Simple Properties of DeBruijn Graphs	114
7.3	A General Realization of DB	115
7.4	A Useful Fact	117
7.5	Two Operations	118
7.6	Subgraph Geometry	121
7.7	The Interpretation of $\mathbb{D}(r, m)$ Structure	123
7.7.1	$k = 0$: Topological Structure and Static Encodings	124
7.7.2	$k = 1$: Induced Topology and Local Checkability	125
7.7.3	$k > 1$: Local Geometry and Robustness	126
8	Two Applications	129
8.1	A Prototype Global-to-Local Compiler	129
8.1.1	What Kind of a Thing is a Global-to-Local Compiler?	129
8.1.2	Pattern Description for Input Specification	130
8.1.3	The Compiler	131
8.2	Analyzing a <i>Drosophila</i> Gene Network	133
8.2.1	<i>Drosophila melanogaster</i>	133
8.2.2	Multi-Agent Model	135
8.2.3	Analysis	137
9	Conclusions and Future Work	143
9.1	Discussion	143
9.2	Future Work	143
9.3	Generalization	143
9.3.1	Application	145
9.3.2	Deeper Analysis	148
	Bibliography	149
A	Local Balls and Parts	151
A.1	Counting Ball Types	151
A.2	Reduction to Parts	152
B	Naive Backtracking Proofs	155
B.1	Appendix: Proof of Prop. 14	155
B.2	Appendix to §3.3	157

C	Details of Faster Algorithm Proofs	164
C.1	Proofs for §4.1.2	164
C.2	Details of the General Algorithm	168
C.3	Proof of Prop. 22	171
D	Details of \tilde{P}	175
D.1	\tilde{P} is More Robust	175
D.2	Computing \tilde{P} from \mathcal{F}	176
D.3	Computing Examples of \tilde{P}	179
E	Generic Dynamic Encoding	181
F	Other Approaches to Pattern Description	184
F.1	Direct Specification	184
F.2	Formal Logic	184
G	Details of Pattern Space Analysis	187
G.1	Gross Structure of Pattern Space	187
G.2	Proof of Prop. 40	189
G.3	Proof of Prop. 43	191
G.4	Proof of Prop. 45	192
H	Drosophila Regulatory Networks	197
H.1	The Chao-Tang Model	197
H.2	Analysis	199

Introduction

Spatial Multi-Agent Systems

A *spatial multi-agent system* is a decentralized system composed of numerous identically programmed agents that either form or are embedded in a geometric space. The agents' computational constraints are spatially local. Each agent has limited internal memory and processing power, and communicates only with neighboring agents. The systems' computational goals, however, are typically defined relative to the global spatial structure.

The world is filled with examples of spatial multi-agent systems. Embryonic development is a prototypical biological example, where large numbers of identically-programmed cells interact through coupled gene regulatory networks to generate a spatially complex organism from an undifferentiated embryo [38, 6]. Swarms of agents, both natural and engineered, cooperate to achieve apparently intelligent search, construction and transport tasks from innumerable local actions [25, 31, 32, 4]. Sensor networks distributed over large open regions or embedded in buildings react to dynamic global environments by collectively processing spatially localized data [13, 36]. A "paintable computer" that could form image displays and data storage devices on-the-fly would be an interesting, if not yet entirely accessible, form of spatial computing [3].

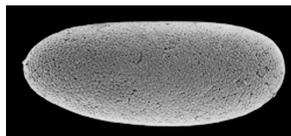
The Need for a Theory

Computer scientists who come into contact with natural spatial multi-agent systems often are intrigued with the idea of adding those systems' capabilities to their programming repertoire. But when trying to figure out exactly how, they quickly encounter the challenge of *programmable self-organization*. How does one design local interaction rules to achieve prespecified global goals? Despite the many compelling examples in nature, decentralized systems seem difficult to reason about. Any would-be programmer of spatial computers has probably experienced the feeling that iterated local rules sometimes generate spiraling cascades of complexity – and sometimes don't – for no immediately discernible reason.

The difficulty of rationalizing local rule behavior is compounded by the desire to replicate the *robustness* that many natural decentralized systems possess. Local rule programs should be stable to significant variations in both initial conditions and agent timing, so that unexpected initializations and perturbations are automatically self-repaired, and unplanned communications delays and temporary hardware failures are not fatal. Furthermore, some global tasks may not be solvable at all, given the local agents' computational and informational limitations. It would be very useful to have an idea ahead of time of whether a given problem is solvable by local interactions using given resources – and if not, be able (if possible) to shift resource allocations accordingly.

Converse to these software engineering concerns is the possibility that thinking about natural spatial computers, *qua* computers, would be helpful scientifically. A programming language that systematically generates the spatial patterns created by biological embryos might provide reasonable null hypotheses to which the real biological programs – i.e. the development gene networks – could be compared. Following the tradeoffs between resources like internal cellular "memory" usage and intercellular communications throughput could indicate the shape of evolutionary constraints.

For all these reasons, it would be useful to have a systematic theory of local-to-global algorithms in spatially distributed multi-agent systems.



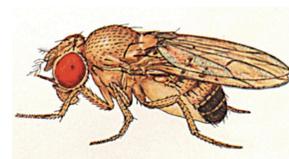
(a) An embryo of the fruit fly *Drosophila melanogaster*.



(b) Cell interaction causes differentiation,



(c) resulting in complex morphological changes,



(d) which produce the adult fly.



(e) A *Polistes humilis* wasp begins nest construction.



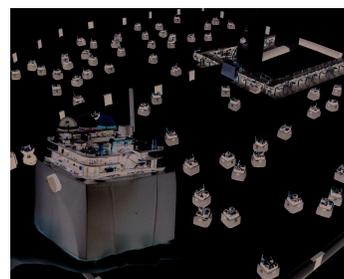
(f) Structure grows as wasps independently apply local building rules.



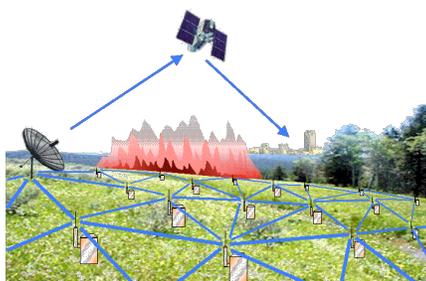
(g) Highly organized structure emerges from stigmergic interaction.



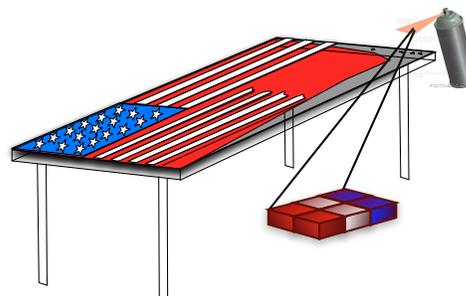
(h) A flock of white Ibis solves a distributed consensus in determining which direction to fly.



(i) Mobile robot swarms can perform search functions.



(j) Sensor networks aggregate and process local data (e.g. temperature) before uploading to central receiver.



(k) In a paintable computer, individual microscale spray-on light-emitting processors decide which color to display by communicating with nearby processors, forming an image.

Thesis Statement

In this thesis, I develop the beginnings of theory of spatial multi-agent systems, for the simple case of pattern formation in a one-dimensional discrete model. First, I characterize those patterns that *are* robustly self-organizable in terms of a simple “necessary condition on solvability”. I then solve the inverse problem, constructing an algorithmic procedure that generates robust local rule solutions to *any* desired solvable pattern. Next, I analyze resource usage and runtime properties of such local rule solutions.

I apply this suite of mathematical techniques to two diverse “global-to-local” problems: the engineering goal of developing a generic “global-to-local” compilation procedure, and the scientific goal of analyzing an embryological development process.

Specific Contributions

I begin by dividing the larger problem of one-dimensional pattern formation into a series of subproblems, and then develop solutions to each. These are:

- An **existence problem**: I demonstrate a simple criterion, called *local checkability*, that any robustly solvable pattern must satisfy. Local checkability is a “necessary condition:” if a pattern is not locally checkable, no robust local rule can self-organize it.
- A **construction problem**: I show that local checkability is also a “sufficient condition,” by describing a generic construction which yields robust local solutions to any locally checkable pattern. The construction exploits a *distributed signalling system* implemented by a *self-organizing Turing machine*.
- An **optimization problem**: The initial construction above is slow. I show how to greatly improve runtime performance by building *smart decision pruning protocols* into the signals sent by the self-organized Turing machine.
- A **resource management problem**: I show that there is a *resource tradeoff continuum* between the two main resource parameters of the multi-agent system: the agent communication radius and agent memory size. I describe algorithms for “tuning” solutions along the continuum.
- A **performance limitation problem**: I show how to obtain tight lower bounds on how fast any local rule can create local checkable patterns, both in average- and worst-case. The technique behind these results is the analysis of a *quantitative measure of global order*, derived as a form of *Fourier analysis of patterns*.

Each of these problems/solutions are aided by a *graph-theoretic characterization* of locally checkable patterns, a tool at the foundation of my theory.

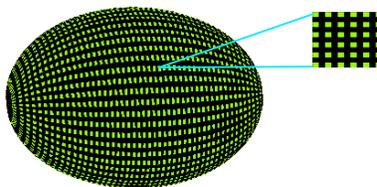
Together, these ideas constitute a suite of techniques that can be combined into global-to-local applications. In this thesis, I sketch their application to two contrasting uses:

- Building a **Global-to-Local Compiler**: This is a procedure which takes as input a target pattern and set of resource limits, and whose output is an optimized local rule generating the pattern using no more than the specified resources. The utility of such a compiler is that it “takes out the work” of having to figure out exactly how to program local rules, and allows global design intent to be directly transduced into local agent programs.
- **Analyzing the structure of a developmental gene regulatory network** in the common fruit fly *Drosophila melanogaster*. Building a toy model of *Drosophila* embryo as a spatial multi-agent system, I show how features of the known genetic regulatory structure might be consequences of local-to-global information constraints.

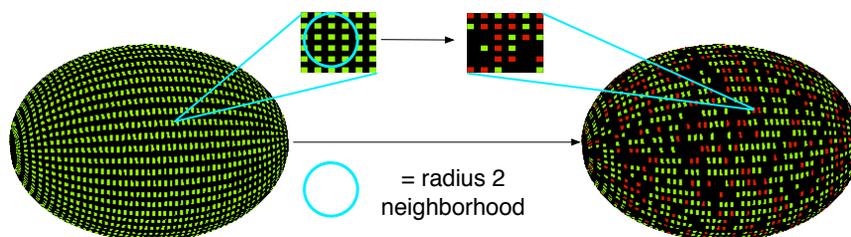
The methods presented concentrate only address one-dimensional pattern formation. However, many of the techniques developed here naturally generalize to more complicated underlying spaces, as I discuss briefly at the end of the thesis.

An Illustrative Example

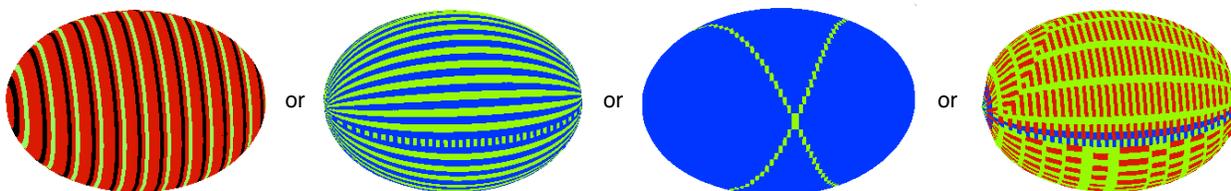
Imagine that we're given a distributed multi-agent system embedded in an ellipsoidal surface. Each agent has some internal state, whose value is indicated to the external environment by color:



Let's assume each agent can only see local information. That is, the rule each agent uses to update its own state (e.g. change color) can only be a function of the states of agents within a neighborhood of some fixed "locality radius" R :



Now, suppose our goal is to find local rules – with radius no larger than R – which, when run separately and identically on each agent, cause the system to robustly form various spatial patterns. For example, given any of these (or similar) patterns:



we'd like to find local rules that *robustly generate* the patterns – meaning, that the rules cause the pattern to form regardless of initial conditions or temporary agent failures.

A *robust global-to-local compiler* is a procedure that allows us to achieve this in an automated fashion. It takes as input a description of a desired spatial pattern, as well as a statement of some bounds on the resources available to the system, and produces as output a local rule that will construct the pattern robustly. This thesis provides a set of tools that can be combined both to make a global-to-local compiler and to analyze its outputs, including the following ingredients: an "existence module," which determines if the input pattern is solvable via local rules at all, and if so, what the minimal locality radius r required to solve the pattern is; a "construction module," which synthesizes a local program that, when possible, achieves the input pattern within the given resource limitations; a "resource tradeoff module," which re-balances the use of certain resources in the system to accommodate the limitations on the others – e.g. increasing the amount of state memory used by each agent, to compensate a limitation of agents to only using nearest neighbor ($r = 1$) communication; and, analysis tools – including, for example, a "measure of global order" which provides a quantitative observable of the system's self-organization.

The figure on the facing page shows such a compiler in action:

- A) shows the console of the compiler implemented in Matlab, interactively taking commands from the user. In this case, pattern is inputted via a "unit" to be repeated along the ellipsoid's major axis (here, [1 2 0 0 0 3]). The compiler determines minimal required radius (here, 3) and receives input about user's maximal available radius (in this case, 1). When the former is larger than the latter (as in this case), the compiler invokes the radius-state tradeoff module.

- B) shows snapshots at various points along the trajectory of the local rule produced by the compiler on a randomly chosen initial condition. The rule is robust both to initial conditions and timing delays.
- C) shows a measure of global order used to analyze the system. The measure increases over time as self-organization occurs.

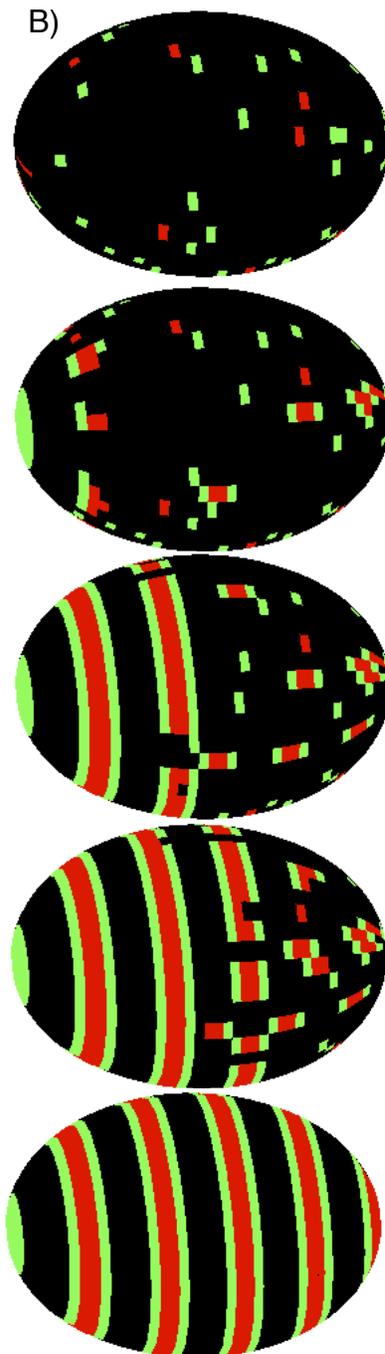
A)

```

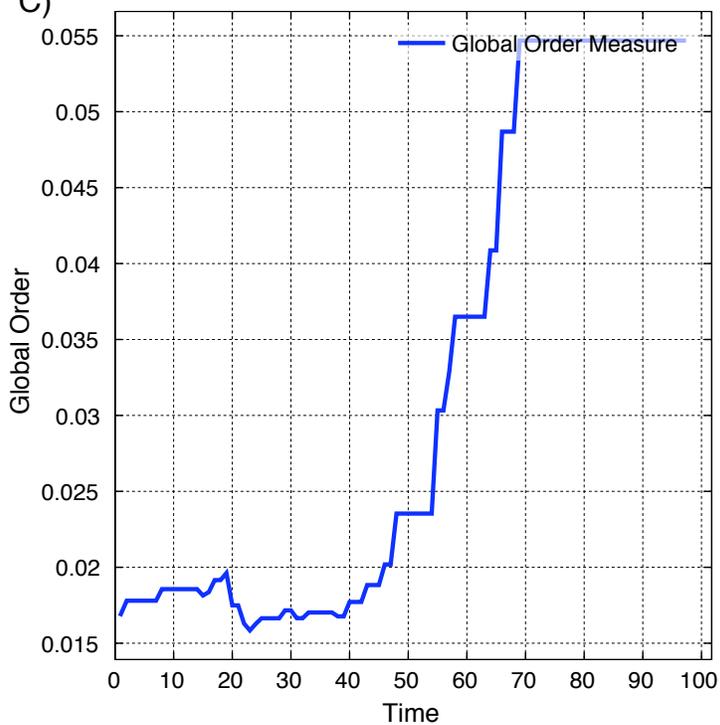
MATLAB 7.5.0 (R2007b)
File Edit Debug Desktop Window Help
/Users/DKY1/sci
Shortcuts How to Add What's New
>> Y = Two_D_Ellipsoidal_Glocal_Compiler();
What pattern would you like? [1 2 0 0 0 0 3]
To form that pattern with 4 states you need a minimal radius of 3.
What is your maximum radius? 1
To form the pattern with radius 1 you need 4 extra state slots.
How long an ellipsoid do you want? 40
How wide an ellipsoid do you want? 40
A robust local rule solution to this pattern
has been written to the file "local_rule.txt" and the
output variable contains a trajectory generated by the
rule on a randomly chosen initial condition.
>>
Start

```

B)



C)



Related Work

Connections to two related lines of research arise repeatedly throughout this thesis: the theory of Cellular Automata, and the theory of formal languages.

Cellular Automata: The one-dimensional multi-agent model I use is similar to that of Cellular Automata. Studies of cellular automata go back over 50 years, from the pioneering work of Ulam and von Neumann to the more recent attempts to classify automata rules, and their use in modeling real biological systems [33, 37]. Specific rules, like Conway’s Game of Life, have been shown to generate a variety of complex behaviors from varying initial conditions [5]. The main emphasis of these studies has been to understand the local-to-global connection: given a local agent rule, what types of patterns can it produce?

In contrast, engineering emphasizes the inverse global-to-local question: given a pattern, which agent rules will robustly produce it? Genetic algorithms have been used to “evolve” CA rules that, although somewhat difficult to analyze, approximately achieve target computations [27]. Recent work has shown that a more principled approach is tractable, allowing the creation of pattern formation languages for CA-like amorphous computers and modular robots [29, 15, 32, 1]. This thesis develops the “principled approach” further, obtaining more comprehensive theoretical results in a simpler model.

In summary: while the basic Cellular Automata model is similar, the questions and results are quite different. To throw light on these relations, throughout the thesis I note where they arise.

Formal Languages: The computational “goals” of the systems I work with are one-dimensional patterns. My notion of a one-dimensional pattern is identical with that of a formal language, and the locally checkable patterns are related to regular languages. The graph characterization of locally checkable patterns I use in various places is equivalent to the characterization of regular languages in terms of Non-Deterministic Finite Automata (NDFAs).

The differences between my work and the results from formal languages are significant, however. First, I address a problem of *construction* rather than *recognition*. The patterns that my model makes are physically embodied in the states of the multi-agent system and the goal is to robustly have the system generate the pattern from many initial configurations – not have it “recognize” if an already generated word fits a given pattern. Second, the main *dynamical* constructions in my theory are the local rule algorithms that construct the patterns. The behavior of these local rules is *not* easily characterized by languages. So, while my systems’ computational goals are essentially the construction of formal languages, the basic tools by which this is accomplished don’t easily fall within the theory of those languages. Again, I note throughout where relationships with language theory arise.

Thesis Outline

Chapters 1-3 form the “basic module” of this work. Chapter 1 introduces the one-dimensional multi-agent systems model used throughout. Chapter 2 introduces the concept of local checkability, the proof of its necessity, and characterizes the locally checkable properties as graphs. Chapter 3 contains the local rule algorithm construction showing local checkability is sufficient, and an analysis of the constructions’ run-time performance.

Chapters 4-7 can be thought of as “further topics.” Chapter 4 discusses techniques for constructing fast algorithms. Chapter 5 introduces the Fourier analysis of patterns, and uses it to provide proof of the optimality of the constructions in chapter 4. Chapter 6 describes the radius/state resource tradeoff. The first portion of Chapter 6, which may be relevant to many readers, can be independently from 4 and 5. Chapter 7 contains an analysis of the space of all locally checkable patterns and shows how this sheds light on pattern robustness.

Chapter 8 contains the two brief applications of the theory. It depends on material from all of the preceding chapters except 5, although if certain results are accepted on faith, chapters 4 and 7 can also be skipped. Chapter 9 describes future work. Throughout, purely technical issues that arise are treated in appendices at the end.

Chapter 1

A Simple Model

This chapter introduces a simple mathematical framework for describing one-dimensional distributed multi-agent systems and the patterns they can form.

First, I describe (§1.1) the static configurations that can appear in the system – single time-step “snapshots” of all the agents, with their simple one-dimensional spatial relationships and various internal states. Then I introduce a notion of dynamics (§1.2), local rules by which agents act to modify their states as a function of the information they receive from neighboring agents. Next, I introduce one-dimensional patterns (§1.3), and the notion of local rule solutions to these patterns (1.4). In §1.5, I sketch illustrative results from the simplest cases and formulate the general questions (§1.5) that motivate the next several chapters. Finally, I describe the relationship of this model with related work in cellular automata in §1.6.

1.1 Static Configurations

The main objects in our model are *configurations*. A configuration is a structure that looks like this:

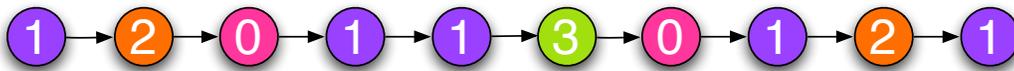


Figure 1.1: A ten-agent configuration with four internal states: purple (0), orange (1), pink (2), and green (3).

Intuitively, configurations are meant to provide a simple model of a one-dimensional multi-agent system. Mathematically, they are node-labeled graphs. Nodes of the graph correspond to agents (or cells), and the colored label of the nodes correspond to agents’ internal states. The nodes are arranged geometrically into a 1-D lattice, with the spatial relations expressed by the directed edges between the nodes.

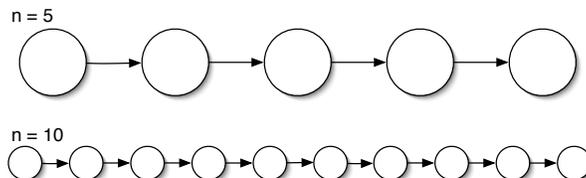
To formalize this picture, we first define the graph of the underlying one-dimensional geometry.

Definition 1 [*Underlying 1-D Geometry*] The directed line of length n , denoted L_n , is the graph

$$L_n = (V_n, E_n)$$

where $V_n = \{1, \dots, n\}$ are the n nodes and $E_n = \{(1, 2), (2, 3), \dots, (n - 1, n)\}$ are the edges.

Example 1 The $n = 5$ and $n = 10$ versions of L_n look like:



L_5 models a system with 5 agents, and L_{10} a system with ten agents. Generally, the size- n graph L_n is the underlying geometry of a system with n agents.

Now we can “put in” the agents’ internal states by labeling each position in the L_n graph with a “color.”

Definition 2 [System Configurations] Let S be a finite set of size m . An S -configuration of size n is a graph

$$X = (V, E_n)$$

in which the edges are the same as in L_n , but the nodes are now pairs (i, s_i) , with $s_i \in S$. Let $\mathcal{C}_{n,S}$ be the set of all S -configurations of size n , and let $\mathcal{C}_S = \bigcup_{n \in \mathbb{N}} \mathcal{C}_{n,S}$.

The set S is intuitively meant to designate the possible internal states of the agents, and the labeling of the nodes is the idea that each agent is in at least one of these states at any given time. Two different nodes (i, s_i) and (j, s_j) can have the same label, meaning that agent i and agent j have the same internal state.

We can think of S -configurations as finite m -ary sequences by assigning to each of the m states a unique integer in $\{0, \dots, m-1\}$ and reading the states off from left to right. For any m -ary sequence x let $x(i : j)$ denote the restriction of the sequence between its i -th and j -th places (inclusive) and let $x_1 \circ x_2$ denote the concatenation of the two sequences x_1 and x_2 . If x is a consecutive subsequence of y , write $x \in y$. Restriction and concatenation, and \in , correspond to obvious graph operations and relations on configurations.¹

Given a configuration X and an agent a in X , we wish to define the local neighborhood around a in X . The neighborhood will of course be a function of a choice of a *radius* parameter r .

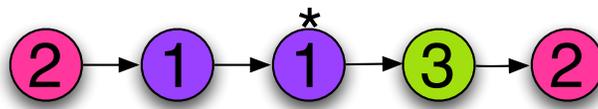
Definition 3 [Local Balls] The ball in X of radius r at agent i , denoted $B_r(i, X)$, is the subgraph of X whose nodes are

$$\{b_j \in V(X) \mid |i - j| \leq r\} \cup \{(i, \text{star})\}$$

and whose edges are induced from X . Let $\mathcal{B}_{r,S}$ denote the set of all r -balls with states in S .

We have added a special symbol \star to the label of the agent i around which the ball is taken (formally, was done by adding the pair (i, \star) to the set of nodes). The point of adding the special \star symbol is to be able to distinguish the “center” of the ball from the other elements in the cases when the configuration X itself is of on the order of r , the radius of the ball.

For example, in the configuration in figure 1.1, the local ball of radius 2 around agent 5 looks like:

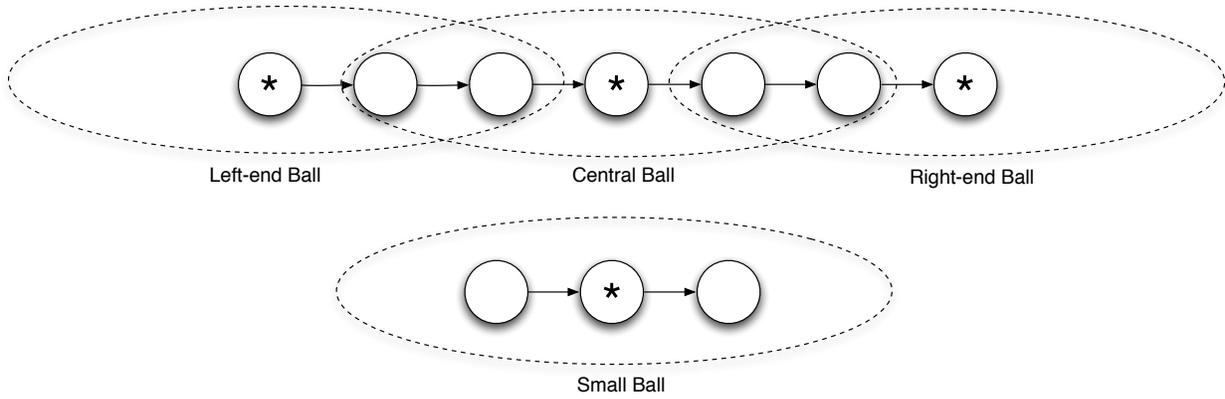


There are essentially three basic isomorphism types of balls:

- Central balls, of size $2r + 1$, which come up in the center of configurations of size larger than the scale of the radius r . Denoted $\mathcal{B}_{\text{central}}(r)$
- Left-end and right-end balls, which come up at the left and right ends of configurations, denoted $\mathcal{B}_{\text{left}}(r)$ and $\mathcal{B}_{\text{right}}(r)$, respectively. And,
- Small balls, which arise in configurations that are small compared to r – denoted $\mathcal{B}_{\text{small}}$

Intuitively, these arise as in the picture:

¹The reason that we use the graph representation at all, and don’t just use the sequence representation, is that the graph description will enable generalization later on.



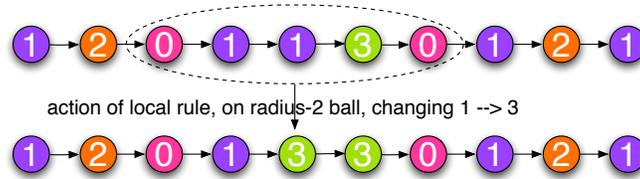
The total number of distinct balls is shown in appendix §A.1 to be exactly $m \binom{m+1}{m-1}$, where $m = |S|$.

Given any local ball B let $\star(B)$ denote the position of the starred agent. Any r -ball can be thought of as a pair $\mathbf{B} = (B, \star(B))$, where B is the underlying string of the ball. We can also define local coordinates $B(i) = B(\star(B) + i)$, so that $B(0) = B(\star(B))$, $B(1)$ is the agent adjacent to the right, $B(-1)$ to the left etc... If $\star(B) = |B|$, then B is the right-end ball and the agent the right-most agent. If $\star(B) = 1$ then the agent is the left-most agent. If B is a ball of radius $r + k$, let $B' = B(i - r : i + r)$, in local coordinates. Two balls B_1 and B_2 are *isomorphic*, denoted $B_1 \sim B_2$, if there are identical as graphs, or, alternatively, if they are identical when described in local coordinates.

1.2 Local Rule Dynamics

So far, we've just set up a static model of configurations; a configuration represents the overall state of the system at a given snapshot in time. Now we need to add dynamics.

Intuitively, dynamics should be generated by agent-based programs running identically on each of the agents in the 1-D multi-agent system, and drawing information only from other nearby agents. Such "local rules" should enable an agent to *act*, changing its state as a function of the state configuration within a fixed-radius local ball:



A local rule can naturally be thought of as a look-up table F , by which an agent a takes local information from the radius- r ball $B_r(a, X)$ around a , and chooses a new state to adopt as a function of what it sees. The inputs to the look up are local r -balls, so therefore the domain of F should be the set of all r -balls \mathcal{B}_r . The outputs of the function F are the new states that the agents will adopt, so therefore the range of F should be the set of all possible agent internal states S . We put one restriction on the kind of function that f can be, namely that whenever B_1 and B_2 are locally identical balls, then $f(B_1) = f(B_2)$. This encodes mathematically the idea that all central agents enact the same local rule, regardless of absolute position. The end-agents can see that they're on the end because the local ball structure is different for those agents than for the agents in the center. A single local rule can therefore instruct the end agents to act differently than the center agents.

Formally,

Definition 4 [Local Rules] A local rule of information radius r is any function from local balls of radius r to the set of states S :

$$F : \mathcal{B}_{r,S} \rightarrow S$$

such that when $B_1 \sim B_2$, $F(B_1) = F(B_2)$. We denote the information radius of f by $r(f)$. Let \mathcal{D}_r denote the set of all local rules of radius r , and let $\mathcal{D} = \bigcup_{r \in \mathbb{N}} \mathcal{D}_r$.

A local rule can act on a configuration X one agent at a time, by changing the label of a single agent $i \in \{1, \dots, |X|\}$ from whatever it is in X , to $f(B_{r(f)}(i, X))$. The situation depicted in the figure just above is the action of a rule defined by

$$f((pink, purple, purple^*, green, pink)) = green$$

on agent 5 of the configuration depicted in figure 1.1. We can also allow f to act on more than one agent simultaneously by having some given *subset* of agents all update their states according to f . Given $c \subset \{1, \dots, |X|\}$, define $f(c, X)$ to be the configuration obtained by replacing the labels of each $i \in c$ with $f(B_{r(f)}(i, X))$.

By stringing together such actions, we will obtain dynamics. Of course, the dynamics depend on the order in which the agents are “called”:

Definition 5 [Call Sequences] A call sequence for a size- n configuration is an infinite sequence of sets of agents, i.e. $c = (c_1, c_2, c_3, \dots)$ where each $c_i \subset \{1, \dots, n\}$.

Given a local rule f , an initial configuration X_0 , and a call sequence s , we can make a trajectory:

Definition 6 The trajectory of X_0 generated by (f, c) is the set

$$\{f_c^n(X_0) | n \in \mathbb{N}\},$$

where

$$f_c^n(X_0) = f(c_n, f_c^{n-1}(X_0))$$

and $f_c^0(X_0) = X_0$. The omega-limit of a trajectory is the set of all configurations that appear infinitely many times; it is denoted $\Omega(f, c, X_0)$.

Intuitively, a trajectory is the result of each agent iteratively applying the local rule f , in the order given by the call sequence c , starting at initial condition X_0 . The omega-limit is the state (or states) to which the system eventually converges. Note that since $\mathbb{C}_{n,S}$ is a finite set of size $|S|^n$, and since $f(a, X) \in \mathbb{C}_{n,S}$ if $X \in \mathbb{C}_{n,S}$, that the omega-limit of all trajectories is non-empty. If $\Omega(f, s, X)$ contains a single configuration, then we say that the trajectory has a well-defined limit denoted $\lim(f, c, X)$. A singleton omega-limit corresponds to a steady state, while larger ones corresponds to a limit cycle.

1.2.1 Timing Models

A timing model is simply a set of call sequences.

Definition 7 [Timing Models] A timing model is a set

$$\mathcal{S} = \bigcup_{n \in \mathbb{N}} \mathcal{S}_n,$$

where \mathcal{S}_n is a non-empty set of call sequences for configurations of size n . A timing model \mathcal{S} is asynchronous if all call sequences $c \in \mathcal{S}$, are sequences of individual agent names. A timing model \mathcal{S} is live if for all $c \in \mathcal{S}_n$, each agent $i \in [n]$ appears in c infinitely many times. A timing model is uniform if the probability $p_t(i)$ that agent i appears in c_t is identical for all i and t .

Three archetypal timing models include:

- The *synchronous timing model*, denoted \mathcal{S}^s , the set of call sequences

$$\{c_n = ([n], [n], [n], \dots)\},$$

where $[n] = \{1, \dots, n\}$. In words, all agents are called simultaneously at all timesteps.

- The *k*-bounded asynchronous model, denoted $\mathcal{S}(k)$, the timing model in which $\mathcal{S}_n(k)$ consists of all sequences

$$\{c = (i_1, i_2, \dots)\}$$

of individual agents $c_j \in [n]$, such that if a given agent i appears $k + 1$ times in c between timesteps t_1 and t_2 then all $j \in [n]$ appear in c between t_1 and t_2 . In words, all agents are called one at a time, and no agent can appear more than k times in any period of time unless all agents have been called at least once during that same period.

- The *totally asynchronous timing model*, denoted \mathcal{S}^a , the set of all call sequences in which each node is called one at a time in some unknown order, with the one restriction that each agent is always eventually called infinitely many times.

All three of these timing models are live (\mathcal{S}^a is by definition the largest asynchronous live model), and all three are uniform.

A natural quantity associated with any live timing model is the average length of a “round”, i.e. a minimal period in which all agents are called once. Formally, for each $s \in \mathcal{S}_n$ and $t \in \mathbb{N}$, let

$$r(s, t) = \min\{t' > t \mid s(t : t') \text{ contains a call to every agent in } [n]\}.$$

If \mathcal{S} is uniform, then the average round time

$$R_{\mathcal{S}}(n) = \langle r(s, t) \rangle_{s \in \mathcal{S}_n}$$

is independent of t . For the synchronous timing model, evidently $R_{\mathcal{S}^s}(n) = 1$. For the *k*-bounded asynchronous model, it is easy to see that there is a constant $C < k$ such that $R_{\mathcal{S}(k)}(n) \sim Cn$.

For the totally asynchronous model, the round-time is equivalent to the following computation: consider a binary x of length n containing K 1s. Then, consider the process in which at each timestep one $j \in \{1, 2, \dots, n\}$ is drawn randomly from the uniform distribution, and $x(j) \leftarrow \max(0, x(j) - 1)$. Let E_K denote the expected number of timesteps requires for E_K to become 0 identically. It is not hard to see that $E_K = nH_K - K$, where $H_K \triangleq \sum_{l=1}^K (1/l)$ is the K -th harmonic number. The round time $R_{\mathcal{S}^a}(n) = E_n$, so because the harmonic numbers grow asymptotically like $\log(n)$, $R_{\mathcal{S}^a}(n) \sim C'n \log(n)$ for a constant C' .

1.3 Patterns

Intuitively, a pattern is an arrangement of states that is ordered in some fashion. This order can be described in a number of ways, the simplest being to define patterns as sets configurations.

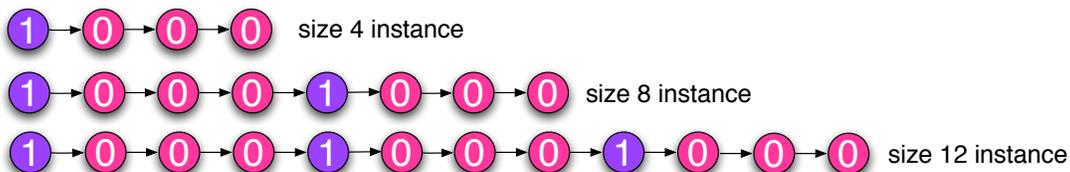
Definition 8 [Patterns] A pattern over state set S is a subset

$$T \subset \mathbb{C}_S.$$

The elements $X \in T$ are instances of the pattern.

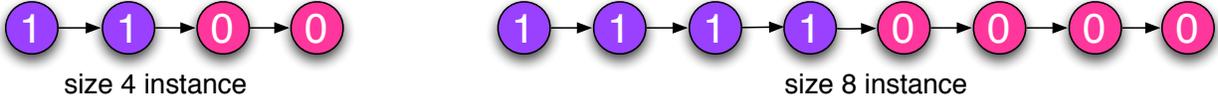
In this very naive definition, the pattern itself is the totality of its instances. More abstracted notions of pattern will be explored in later chapters. Here we explore some specific examples.

Example 2 [Repeat Patterns] For example, a pattern might have one agent in state 1, followed by three agents in state 0, like this: . By repeatedly concatenating this segment with itself we obtain a version of the 1000 pattern at every size that is a multiple of 4:



This pattern is an example of a more general class of *repeat* patterns, characterized by repeating a single finite pattern generator an integral number of times. In terms of the m -ary sequence description, if q is a fixed m -ary sequence, then the q -generated pattern T_q is the set of configurations $\{q^m | m \in \mathbb{N}\}$. For example, if $q = 1$ then $T_q = \{1, 11, 111, 1111, \dots\}$. If $q = 12$ then $T_q = \{12, 1212, 121212, \dots\}$. Evidently $T_q \cap \mathbb{C}_n \neq \emptyset$ only when $n = m|q|$ for some m , i.e. the pattern T_q only has versions at sizes that are multiples of the size of the generating unit. Hence the number of instances of T_q up to size n grows linearly with n .

Example 3 [Proportionate Patterns] In contrast, consider the “half pattern” $T_{1/2}$ consisting of configurations of the form $1^n 0^n$, for each n :



$T_{1/2}$ is an example of the general class of *proportionate* patterns, characterized by having a subpattern (or a change in background pattern) appear at a specific fractional value along an otherwise (piecewise) uniform structure. The general specification of a proportionate pattern is a list of pairs $F = \{(q_i, p_i) | 1 \leq i \leq K\}$ where q_i are finite units representing the uniform background regions and the p_i are finite configurations representing the “interrupting” structures. For each integer n , let

$$X_n = q_1^{on} \circ p_1 \circ q_2^{on} \circ p_2 \circ \dots \circ q_K^{on} \circ p_K.$$

The pattern T_F is the set of all such X_n for $n \in \mathbb{N}$. In the case of $T_{1/2}$, $K = 2$ and $F = \{(s_1, s_2), (s_1, \emptyset)\}$. In general, the structure p_i appears in X_n at the fractional position

$$\frac{\sum_{j < i} n|q_j| + |s_j| + n|q_i|}{\sum_{j=1}^K n|q_j| + |p_j|} \rightarrow \frac{\sum_{j \leq i} |q_j|}{\sum_{j=1}^K |q_j|}$$

where the limit is taken as $n \rightarrow \infty$. A proportionate pattern is *trivial* when all the p_i are empty and the q_i are degenerate – in which case it is simply a repeat pattern.

1.3.1 Pattern Properties

The repeat patterns are especially nice in that any subsection of any instance of a repeat pattern can be expanded to a pattern of an arbitrary larger size. This idea, however, makes sense for any pattern.

Definition 9 [Expandability] A configuration X is T -consistent if there are configurations Y and Z such that $Y \circ X \circ Z \in T$. A T -consistent configuration X is T -expandable if for all n , there is $Y \in T$ with $|Y| > n$ such that $X \subseteq Y$. The failure of X to be expandable means intuitively that any structure containing X can not be scaled larger than some fixed size. A pattern T is completely expandable if all its instances are expandable and r -expandable if all T -consistent configurations of size r or less are expandable.

Example 4 Consider any nontrivial proportionate pattern T_F where $|F| > 1$. Then T_F is not completely expandable, but is l -expandable, where $l = \min_i |q_i| - 1$.

The repeat patterns satisfy the strengthening of expandability to *repeatability*.

Definition 10 [Repeatability] We define X to be T -repeatably if for all n there is $Y \in T$ such that X appears as a subsequence of Y in at least n distinct places, i.e. there are $X_j \in Y$ for $j = 1, \dots, n$ such that $X_j \equiv X$ for all j and $X_i \neq X_j$ when $i \neq j$. A pattern T is completely repeatable if all T -consistent configurations are T -repeatably and r -repeatably if all T -consistent configurations of size r or less are repeatable.

Any T that is completely (or r -) repeatable is evidently completely (or r -) expandable. The following examples delineate the relationships between repeatability and expandability:

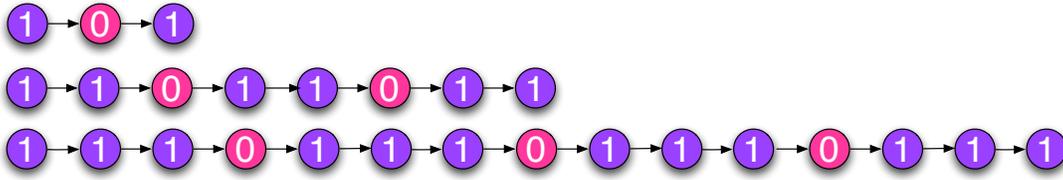


Figure 1.2: The first three elements of the pattern whose n -th instance is described by the binary sequence $0^n(10^n)^n$.

Example 5 Consider proportionate patterns T_F with $F = \{(q_1, p_1), (q_2, \emptyset)\}$ where $p_1 \neq \emptyset$. The half pattern $T_{1/2}$ in figure 3 in §1.3 above is such a pattern. Any such T_F is completely expandable but not r -repeatable for any r .

Now define $x_n = 0^n(10^n)^n$, as illustrated in Fig. 1.2.

Example 6 Let $T_k^1 = \{x_n | n \geq k\}$. T_k^1 is k -repeatable but not completely expandable.

Example 7 Let

$$T_k^2 = \{x_k x_{k+1} \dots x_n | n \geq k\}.$$

T_k^2 is k -repeatable and completely expandable but not completely repeatable.

Example 8 Finally, let $y_n = y_{n-1}y_{n-1}x_n$ with $y_1 = x_1$, and define $T^3 = \{y_n | n \geq 1\}$. T^3 is completely repeatable but not a union of repeat patterns.

What are the minimal “modules” of repeatability?

Definition 11 [Minimal Segments] Suppose u is a configuration such that i) u^n is T -consistent for all $n \geq 1$ and ii) there is no v such that $u = v^{|u|/|v|}$, such that v^n is T -consistent for all $n > 1$, unless $v = u$. Such a u we call a minimal T -segment.

Example 9 For any repeat pattern T_q , the subconfiguration q is the unique segment in the T_q pattern iff there is no q' such that $q \neq (q')^l$ for any $l > 1$. Similarly, q_i are segments of the proportionate patterns.

The concepts of expandability, repeatability, and segments regard the “scability” of a pattern, its configurations, and subconfigurations. “Connectability” properties of patterns are also informative.

Definition 12 [Connectability, Alternability, Compatibility] Define X to be T -connectable to Y there are $i < j$ such that $Z(i : i + |X| - 1) = X$ and $Z(i : i + |Y| - 1) = Y$. X is directly connectable to Y if j can be chosen $\leq i + |X| - 1$. X and Y are T -alternatable if X is T -connectable to Y and vice-versa. We say X and Y are T -compatible if a configuration connecting X and Y is repeatable.

Example 10 In the repeat pattern T_q , the consistent configuration q is obviously connectable to itself. In the proportionate patterns, each q_i is connectable to q_j for $j \leq i$, and direct connectability corresponds to having the intervening p_i 's empty.

Example 11 Let q_1 and q_2 be distinct finite configurations such that neither q_i is a power of the other, and consider the union pattern $T = T_{q_1} \cup T_{q_2}$. The configurations consistent with the T_{q_1} part of the pattern are not connectable to the configurations consistent with T_{q_2} , and vice-versa. Now define the pattern T' by

$$T' = \{q_1^i q_2^j | i, j \in \mathbb{N}\}.$$

In T' , q_1 is connectable to q_2 , but q_1 and q_2 are not alternatable. In the pattern T'' defined by the free group of words over q_1 and q_2 , q_1 and q_2 are repeatedly alternatable, i.e. they are compatible.

Sometimes the presence of one subconfiguration *requires* that another show up somewhere else in any T -admissible structure. Formally,

Definition 13 [Requirement] *The T -consistent configuration X T -requires Y if every T -admissible Z such $X \in Z$ also has $Y \in Z$. The T -required set of X , denoted $R_T(X)$, is the set of T -required configurations.*

For each X , $R_T(X)$ is a *co-filter*, meaning that if $Y \in R_T(X)$, for all $Z \in Y$, we also have $Z \in R_T(X)$. The question is to determine the maximal elements of the filter and conditions for when, given $Y_1, Y_2 \in R_T(X)$, there is $Z \in R_T(X)$ with $Y_1, Y_2 \in Z$.

The notions of connectability, alternability, and requirement are all notions of “correlation”, in that they regard the relationships between a configurations subconfigurations. These concepts are all “boolean”, in that in given situation they either hold or not. More quantitative measures of correlation can also be useful, are introduced and used later on in chapter 5.

1.3.2 Admissible Sizes and Pattern Growth

The repeat pattern T_q only has instances of size $n \cdot |q|$, for each $n \in \mathbb{N}$. In general, we can define:

Definition 14 [Admissible Sizes] *For any pattern T the set of admissible sizes is*

$$\text{Sizes}(T) \triangleq \{n \in \mathbb{N} | T \cap \mathbb{C}_n \neq \emptyset\}.$$

One of the simplest questions about how a pattern scales is its size. The repeat and proportionate patterns are perhaps the simplest patterns that have an infinite number of instances, both growing linearly with n . When $T \cap \mathbb{C}_n$ has more than one element, it means that the pattern is flexible, consistent with multiple different arrangements. For example the union pattern $T = T_{1000} \cup T_{0111}$ allows two possible arrangements at each size that is a multiple of four, so that there are there are approximately $2 \cdot (n/4) = n/2$ instances of T of size n or less.

These ideas suggest that we define:

Definition 15 [Growth Function] *For any pattern T the growth function is*

$$\text{Growth}_T(n) \triangleq \sum_{i \leq n} |T \cap \mathbb{C}_i|.$$

The relevant question about the growth function is its asymptotics as $n \rightarrow \infty$. If the growth function grows very fast, then we’ll want to know its exponential growth rate k_T given by

$$k_T = \lim_{n \rightarrow \infty} \frac{1}{n} \text{Log}(G_T(n))$$

if it exists. If $G_T(n)$ grows slower, we might try to measure a polynomial order α_T and leading coefficient C_T given by

$$\alpha_T = \lim_{n \rightarrow \infty} \frac{\text{Log}(G_T(n))}{\text{Log}(n)}; \text{ and } C_T = \lim_{n \rightarrow \infty} \frac{G_T(n)}{n^{\alpha_T}},$$

respectively, if they exist.

1.4 Solutions

The central goal of this work is to study how multi-agent systems can use local rules to create order from disorder. The concept of “disorder” is defined for the present purposes by taking the whole configuration space \mathbb{C} as a large set of completely unconstrained initial conditions, while “order” is defined by a relatively smaller set of final states constrained to some pattern T . A local rule that creates T will guide the system from arbitrary initial states along trajectories to limits that lie in the prescribed set of final states defined by T . This is only possible on initial conditions X_0 for which T contains a configuration of size $|X_0|$, since the

local rules do not create or remove agents. Moreover, it will depend on a choice of timing model since call sequences are a necessary input to make a trajectory.

We capture the above intuition by defining:

Definition 16 [Solutions] A local rule f is a solution to pattern T in timing model \mathcal{S} if

- for all sizes n , and
- all configurations X of size n , and
- for all call sequences s applying to configurations of size n ,

the limit of the trajectory generated by f starting at X under s is a well-defined element and an element of T whenever T contains at least one configuration of size n . Symbolically,

$$\lim_{n \rightarrow \infty} f_s^n(X) \in T \text{ whenever } T \cap \mathbb{C}_n \neq \emptyset.$$

We impose the condition that $T \cap \mathbb{C}_n \neq \emptyset$ because it would be unfair to expect a rule that cannot add or remove agents to push into T a configuration whose size is wrong. Intuitively, a solution “drives” all initial conditions in \mathbb{C} to the prescribed pattern T :

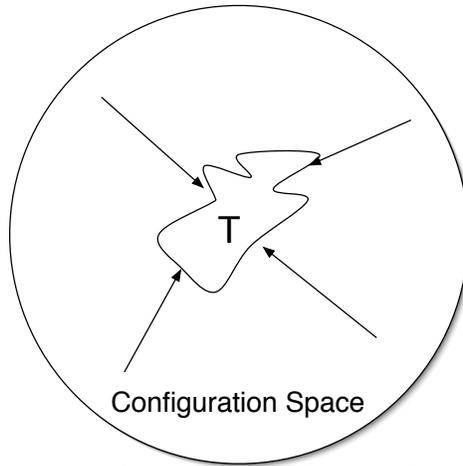


Figure 1.3: A local rule f solving T drives arbitrary configurations in \mathbb{C} to T .

Run-Time Complexity

Given a local rule f that is a solution to pattern T , it is natural to ask how long it takes to produce its solution. For initial condition X and call sequence s , define the *runtime* of f on X, s , denoted $T(f, X, s)$, to be the minimum N such that $f_s^M(X) = f_s^N(X)$ for all $M \geq N$. If there is no such N , evidently $T(f, X, s) = \infty$.

Different timing models may have different numbers of agents called at each time step, so to account equally for this, we define the *per-agent normalized runtime* by

$$T'(f, X, s) = \frac{1}{n} \sum_{i=1}^{T(f, X, s)} |s_i|$$

where $n = |X|$ is the size of X and $|s_i|$ is the number of agents called at time i . To remove dependence on call sequence s , we average:

$$T_{\mathcal{S}}(f, X) = \langle T'(f, X, s) \rangle_{s \in \mathcal{S}_n}$$

taking the uniform distribution over call sequences.

Because $T(f, X)$ will typically scale with $|X|$, what we really want to measure is the scaling dependence. We can capture this in two basic ways: average-case or worst-case.

Definition 17 [Measures of Run-Time Complexity] The average-case runtime scaling function is

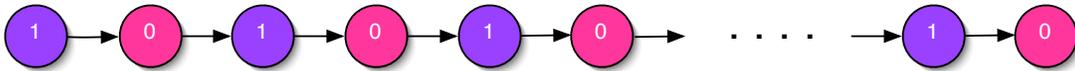
$$T_S^{avg}(f)(n) = \frac{1}{|T \cap \mathbf{C}_n|} \sum_{X \in T \cap \mathbf{C}_n} T_S(f, X).$$

The worst-case runtime scaling function is:

$$T_S^{worst}(f)(n) = \max_{X \in T \cap \mathbf{C}_{\leq n}} T_S(f, X).$$

1.5 Illustrative Examples

To make all these definitions concrete, we look at a couple of simple examples. Consider the repeat pattern T_{10} , whose typical element looks like:

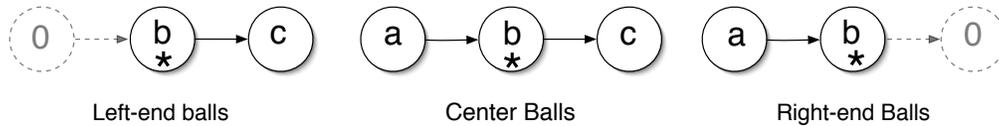


We seek a solution to T_{10} , assuming that each agent has two possible internal states (0 and 1). To do this, first fix the radius $r = 1$. Recalling the definition from before, a local rule with radius 1 is a function

$$f : \mathcal{B}_1 \rightarrow \{0, 1\}$$

in which the domain \mathcal{B}_1 is the set of radius-1 local configurations an agent could see, and the target $\{0, 1\}$ is the set of states that the agent could change to as a function of what it sees.

With radius 1 there are just three kinds of local balls: the one around the left-end agent, the one around the right-end agent, and balls around all other “central” agents:



A central ball with two states, therefore, is a 3-tuple (a, b^*, c) , where $a, b, c \in \{0, 1\}$. Given such a ball B , define $B(-1) = a$ (“the state of the agent to my left”), $B(0) = b$, (“my state”) and $B(1) = c$ (“the state of the agent to my right”). A right-end ball is a pair (a, b^*) , while a left-end ball is (b^*, c) . Define $B(1)$ and $B(-1)$ respectively as defaulting to 0 in these cases.

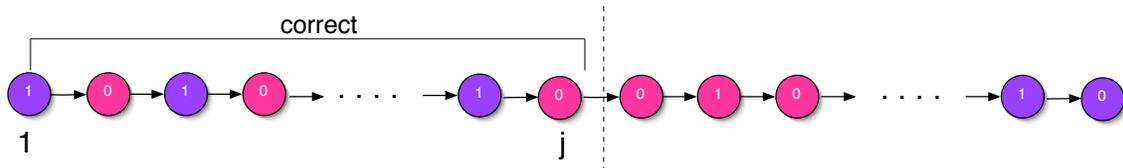
Now, define the local rule F_{10} according to the formula:

$$F_{10}(B) = 1 - B(-1).$$

In words, what F_{10} does is to set a given agent’s state to the opposite of the state of the agent to the left, except if the agent is the left-end, in which case it always sets to 1 (recall that $B(-1) = 0$ by default for the left-end agent).

Proposition 1 F_{10} is a solution to T_{10} in all live timing models.

Proof: Suppose X_0 is any initial configuration that already satisfies the T_{10} pattern up to agent j ,



that is,

$$X_0(1 : j) = (10)^{\circ \lfloor j/2 \rfloor} \circ 1^{\text{odd}(j)},$$

where $odd(j)$ is 1 if j is odd and 0 otherwise. Now, notice first that F_{10} fixes the state of all the agents in $X_0(1 : j)$. Hence for any call sequence s , and all $m > 0$,

$$F_s^m(X_0)(1 : j) = (10)^{\circ \lfloor j/2 \rfloor} \circ 1^{odd(j)}.$$

Next, suppose that the call sequence s calls agent $j + 1$ at least once, say at timestep k . The action of F_{10} on agent $j + 1$ will be to set it to the opposite of the state of the j th agent. Hence $F_s^{k+1}(X_0)$ now satisfies the pattern from agent 1 to $j + 1$. Thus we can apply the same argument just made, but now starting from $j + 1$. Repeating this inductively, F must eventually drive the all agents X to the correct state, as long as each agent is eventually called repeatedly. ■

Notice that the way that F_{10} works relies heavily on distinguishing left from right. Our 1-D model carries a global orientation, since the underlying spatial graph L_n is directed. The algorithm would break down if this were not present.

Given that F_{10} is a solution to T_{10} , it is natural to ask how efficient it is.

Proposition 2 *In the totally synchronous and totally asynchronous timing models, F_{10} has a worst-case runtime that scales linearly with the number of agents. The same holds for average-case runtime in the synchronous model.*

Proof: The proof of proposition 1 shows that

$$T(F_{10}, X, s) \leq \sum_{j=1}^n k_j(s) - k_{j-1}(s),$$

where $k_j(s)$ is inductively defined as the first call to agent j in s after $k_{j-1}(s)$, and $k_0(s)$ is defined to be 0.

For the totally synchronous timing model \mathcal{S}^{sync} , $k_j(s) = j$, so $T^*(f, X, s) \leq |X|$ for all X . There are $|X|$ agents called at each step, so

$$T_{\mathcal{S}^{sync}}(F_{10}, X) \leq n$$

for all X with $|X| = n$. On the other hand, for the initial condition $X_0 = (01)^{\lfloor n/2 \rfloor} 0^{odd(n)}$, it is evident that $T_{\mathcal{S}^{sync}}(F_{10}, X) = n$. X_0 must therefore be a worst case, so

$$T_{\mathcal{S}^{sync}}^{worst}(F_{10})(n) = n.$$

In fact, we can compute a general formula for the synchronous timing model, namely that

$$T(f, X) = |X| - p + 1$$

where p is the location of the left-most error, i.e. the first position where the T_{10} is not satisfied. Now, $Prob[p = P] = \frac{1}{2}^P$, so

$$T_{\mathcal{S}^{sync}}^{avg}(f)(n) = \sum_{P=1}^n (n - P + 1) \cdot (1/2)^P = n - 1 + 2^{-n}.$$

For the totally asynchronous timing model,

$$Prob[k_j(s) - k_{j-1} = \delta] = \frac{1}{n} \left(1 - \frac{1}{n}\right)^{\delta-1}$$

and $k_j(s) - k_{j-1}, k_i(s) - k_{i-1}(s)$ are statistically independent events for $j \neq i$. Hence,

$$T_{\mathcal{S}^{async}}(f, X) \leq n \cdot \frac{1}{n} \cdot \sum_{\delta=1}^{\infty} \frac{\delta}{n} \left(1 - \frac{1}{n}\right)^{\delta-1} = n.$$

On the other hand, for the same initial condition $X_0 = (01)^{\lfloor n/2 \rfloor} 0^{odd(n)}$, similar arguments show that, again, $T_{\mathcal{S}^{async}}(f, X) = n$. ■

The average-case runtime of F_{10} in the asynchronous model also scales linearly with system size, although with a different constant, and requires somewhat more involved arguments to demonstrate.

Proposition 1 demonstrates that T_{10} is solvable with a rule of radius 1. However, this is not true of all patterns. For example:

Proposition 3 *The repeat pattern T_{1000} is not solvable with any rule f having $r(f) = 1$.*

Proof: Suppose f were putation solution to T_{1000} with radius 1. The size-8 version of T_{1000} is 10001000. For this to be a fixed point of f , all of the radius 1 balls in it must be fixed points of f . In particular, this means that $f((1^*, 0)) = 1$ and $f((1, 0^*), 0) = f((0, 0^*), 0) = f((0, 0^*)) = 0$. But then 10000000 must also be a fixed point of f ; since this is not in T_{1000} , f cannot be a solution. ■

In spite of proposition 3, it turns out to be possible to generalize proposition 1 and 2 to find linear-time solutions for all repeat patterns by increasing the radius, as we'll see in Chapter 3.

This is in sharp contrast to:

Proposition 4 *The 1/2-proportion pattern $T_{1/2}$ described in the previous chapter is not solvable with local rules. The same holds for all non-trivial proportionate patterns.*

Proof: $T_{1/2}$ is the set of configurations whose binary sequences are $\{1^n 01^n | n \in \mathbb{N}\}$. Suppose f is a putative solution for $T_{1/2}$. Then let $X \in T_{1/2}$ be a configuration with $|X| > 2r(f)$. (Since $T_{1/2}$ admits configurations of all odd sizes, this is evidently possible.) Let $m_X = (|X| + 1)/2$, the middle position in X . As binary sequences, for positions i within distance r of the middle, i.e. $m_X - r \leq i \leq m_X + r$, then $B_{r(f)}(i, X) = 1^{r-i+m_X} 0^* 1^{m_X-r-i}$. Since X is the unique element in $T_{1/2}$ of size $|X|$, this means that $1^{r-i+m_X} 0^* 1^{m_X-r-i}$ must be fixed points of f . For positions $r + 1 \leq i < m_X - r$ or $m_X + r < i < |X| - r$, $B_{r(f)}(i, X) = 1^{r(f)} 1^* 1^{r(f)}$. For the left-end and right-end positions, the balls are sequences of 1s as well (with \star not in the center). All of these must also be fixed points of f . Now let $n_{i_1}, n_{i_2}, \dots, n_{i_k}$ be a sequence of positions with $r < i_j < n - r$ and $i_j - i_{j'} > 2r(f)$ for $j' < j$. Let Y be the configuration given by all 1s, except 0's at the positions n_{i_j} , i.e.

$$Y = 1^{n_{i_1}} 01^{n_{i_2}-n_{i_1}} 01^{n_{i_3}-n_{i_2}} \dots 01^{n-n_{i_k}} = \left(\bigcirc_{i=1}^K 1^{n_i-n_{i-1}} 0 \right) \circ 1^{n-n_K}$$

where $\bigcirc_{i=1}^n a_i = a_1 \circ a_2 \circ \dots \circ a_n$. Then Y is also a fixed point of f , because all its local balls b are exactly those from X . But $Y \notin T_{1/2}$. Thus any supposed solution for $T_{1/2}$ must have stable configurations that are not in $T_{1/2}$, and therefore can't actually be a solution. A similar argument holds for any other non-trivial proportionate pattern. ■

General Problems

These simple results indicate several general analysis and design problems, including:

- **[Existence and Characterization]:** As we saw in §1.5, some patterns are solvable (like the repeat patterns) while others are not (like the proportionate patterns). This suggests that we find a "existence" criterion for solvability.
- **[The Inverse Problem]:** The inverse of the existence problem would to systematically construct an explicit solution to each pattern that is not ruled out by the existence criterion, like we did in the specific case of the pattern T_{10} in §1.5.
- **[Efficient Algorithms]:** As evidenced by proposition 2, local rules can be subjected to asymptotic runtime analysis, just like any other algorithms. How can we construct the most efficient algorithms for solving a given one-dimensional pattern?
- **[Measures of Global Order]:** The central idea behind self-organization is that order can be achieved from disorder by purely local interactions, and as per 1.4, patterns are supposed to be simple exemplars "global order". To what extent can the amount of order in a pattern be measured and quantified?
- **[Resource Trade-Offs]:** Our one-dimensional model has two basic resource parameters: the radius r of the local rule, and the number of $m = |S|$ of possible internal states that an agent can have. An important question is to characterize the inherent tradeoffs between these two resources.
- **[Robustness Analysis]:** Patterns are subject to *perturbations* – small changes in the set of configurations considered as belonging to the pattern. We thus would want to explore the *robustness* of a pattern's features under such perturbations.

These questions motive the remaining chapters of this thesis.

1.6 Related Work

The model presented in sections 1.1 and 1.2 is evidently similar to that of 1-dimensional cellular automata [37]. The most important difference is that our project is *task driven*, not *mechanism driven*. Specifically, the goal is to determine criteria for when given self-organization problems – e.g. patterns – are *engineerable*, and having determined such criteria, carry out the engineering. We start with the task, and perform engineering activities – both reverse engineering to determine existence of , and forward engineering to actually build the relevant local rules when possible. This is direct contrast the project of most cellular automata studies, which start with local rules and then ask: “what behaviors does this rule generate?” This difference is the motivation behind the definition of “robust solution” in §1.4: we want to engineer and analyze robust solutions, and a concept that is largely absent (as far as we can tell) from the standard theory of cellular automata. The effect that this difference of motivation has on the results of the theory become apparent in the following chapters.

The “task driven” motivation also influences several specific differences between the models. The configurations in my model are *finite*, whereas 1-D cellular automata typically are considered on the bi-infinite integer lattice. Finiteness matters for two reasons. First, the concept of scalable pattern, which is our present interpretation of the idea of global task, only makes complete sense when a pattern has “versions” (or instances) at different sizes. A proportionate pattern with three proportionate substructures, for example, would not have obvious meaning if all configuration had the same (infinite) size. The second reason for using finite structures is that they have *boundaries*. These boundaries are important sources of structure “seeding”, as order arises (as we will see in future chapters) in part by having the edge-agents act differently from other agents. This would be impossible in the infinite lattice case, since the local geometry of all agents is identical. In the usual CA context, such symmetries are broken by constraining the system to specific initial condition. As a result the algorithms tend to depend heavily on structures in the initial conditions (such as Wolfram’s “Turing universal” CA constructions), and thus are not robust in the sense we care about.

In addition, the timing models, presented in 1.2.1, are not all synchronous, whereas cellular automata rules typically are studied in a synchronous update environment. Distributed systems, including those our theory is meant to apply to, are typically asynchronous [23]. Our notion of robustness specifically invokes timing-model invariance, and the algorithms we construct in future chapters are specifically designed to handle unpredictable update orders. The behaviors generated by the rules we develop appear qualitatively different that the kind of “brittle complexity” exhibited by CAs whose behaviors are strongly timing dependent.

Chapter 2

Local Checkability: A Necessary Condition

In definition 16 of §1.4, I defined a local rule f to be a *solution to pattern T in synchrony model \mathcal{S}* if

- for all sizes n , and
- for all configurations X of size n , and
- for all call sequences $s \in \mathcal{S}$ applying to configurations of size n ,

the limit of the trajectory generated by f starting at X under s is a well-defined element of T whenever T contains at least one configuration of size n . Symbolically,

$$\lim_{n \rightarrow \infty} f_s^n(X) \in T \text{ whenever } T \cap \mathbb{C}_n \neq \emptyset.$$

One of the main problems posed in §1.5 was the question of when a given pattern T is solvable. This can be thought of as an existence question, for it asks when the solution space

$$\mathcal{F}^{\mathcal{S}}(T) = \{f \in \mathcal{D} \mid f \text{ is a solution to } T \text{ in } \mathcal{S}\}$$

is nonempty.

In this chapter, I introduce a very simple necessary criteria for solvability. The condition is called *local checkability*, which roughly expresses the idea that a solvable pattern T must possess a subpattern T' that is locally recognizable as a stop state. Any locally checkable property is recognized by an object called a *local check scheme*, which specifies local views that are recognized and accepted.

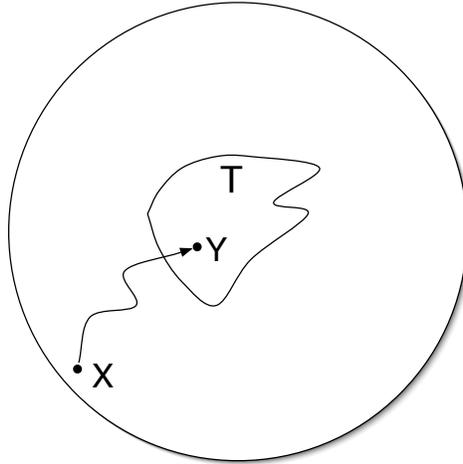
In §2.1, I define local checkability precisely, and prove its necessity. In §2.2, I characterize locally check schemes as “part lists” and show how many properties of local check schemes reduce to computations about these part lists. In §2.3, I show how to characterize local check schemes as graphs, and compute many of the basic properties of patterns as a function of the corresponding graph’s structure. The graph characterization turns out to be crucial for many of the applications in future chapters.

2.1 Local Checkability: A Necessary Condition

Let T be a pattern. Assuming that T contains at least one configuration X at size n , the definition of a local rule f being a solution to pattern T requires that

$$\lim_{n \rightarrow \infty} f_s^n(X) = Y$$

for some fixed configuration $Y \in T$. Schematically, it looks like this:

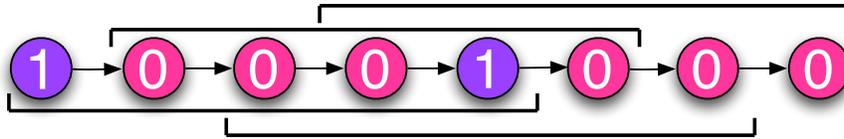


What do we know about this limiting configuration Y ? If Y is to be a stable limiting point, then for all agents $A \in Y$, the action of f must be to *not change anything*. When f is applied to the local balls $B_r(A, Y)$, the result must be to keep whatever state A already has:

$$f(B_r(A, y)) = \text{state}(A, Y)$$

where $\text{state}(A, Y)$ denotes the state that agent A already has in configuration Y . The local balls around the agents in Y are therefore all fixed points of the local rule f . Moreover, the agents' local balls overlap, forming a mutually interlocking configuration-wide stop state.

For example, if a local rule f with radius 2 is a solution to the T_{1000} pattern (introduced in the previous chapter), then in the figure below:



all of the bracketed things are radius-2 balls that must be fixed states of f . That is,

$$f((1, 0, 0^*, 0, 1)) = f((0, 0, 0^*, 0, 1)) = f((0, 1, 0^*, 0, 0)) = 0; \text{ and } f((0, 0, 1^*, 0, 0)) = 1.$$

In general, there are two conditions that f has to satisfy for this to work. First, for each n for which $T \cap \mathcal{B}_n \neq \emptyset$, the local fixed states of f must be able to assemble into a stop state of size n – otherwise, the local rule would never come to a fixed limit. Second, the fixed states cannot assemble into any configuration that is *not* in T – otherwise the system would end up in a bad deadlock away from T , when started from such a configuration.

But now think of the above not as conditions for when some given f can solve T but rather as conditions on when T is solvable in the first place. What these considerations suggest is that the for T to admit any solution f , it must be possible to find a coherent set locally-specifiable stop states as a subset of T – and this may not be possible for all patterns T . To see this formally, we define the notion of *local checkability*.

Recall the definition of $\mathcal{B}_{r,S}$, the set of balls of radius r with states in the set S (normally we'll drop the subscript S). Consider a binary-valued function Θ on \mathcal{B}_r , i.e.

$$\Theta : \mathcal{B}_r \rightarrow \{0, 1\}.$$

Θ should be thought of as a “recognition function” – $\Theta(z) = 1$ means that the local ball z is “recognized” by the central (that is, \star 'ed) agent to be a correct local stop state, and $\Theta(z) = 0$ otherwise. For any such function define

$$\Theta(X) \triangleq \prod_{a \in V(X)} \Theta(B_r(a, X)).$$

This definition means that, when applied to a whole configuration X , $\Theta(X) = 1$ only when $\Theta(B_r(i, X)) = 1$ for all agents i in X – that is, when *all* agents recognize a stop state.

Definition 18 [Local Check Schemes] Let T be a pattern. Θ is a local check scheme for T of radius r if

- For all $X \in \mathbb{C}_S$, $\Theta(X) = 1 \Rightarrow X \in T$.
- For all n such that $T \cap \mathbb{C}_{n,S} \neq \emptyset$, there is $X \in \mathbb{C}_{n,S}$ such that $\Theta(X) = 1$.

The smallest r for which there exists a check scheme of radius r for T is the local check radius of T , denoted $LCR(T)$. If there is no local scheme for T of any finite radius with m states, then $LCR(T)$ is defined to be ∞ .

Intuitively, the first condition in the above definition is the one that prevents bad deadlocks, while the second condition requires there to be at least one good fixed point. This is formalized by the following:

Proposition 5 If f is a solution to T (in any valid synchronicity model), then $r(f) \geq LCR(T)$.

In words, **local checkability is necessary for solvability**.

Proof: Suppose f is a solution to T . Then if $f(c, X) = X$ for all call sequences c only if $X \in T$. Moreover, for each n , choosing any $X \in \mathbb{C}_{n,S}$, let $Y = \text{lim}(f, s, X)$; then $Y \in \mathbb{C}_{n,S}$ and $f(c, Y) = Y$ for all c , so $Y \in T$. Now define $\Theta : \mathcal{B}_{r(f)} \rightarrow \{0, 1\}$ by $\Theta(B_{r(f)}(a, X)) = 1$ if and only if $f(B_{r(f)}(a, X)) = l(a)$, where $l(a)$ is the state of agent a . Notice that $X = f(c, X)$ for all c if and only if for all $a \in V(X)$, $f(B_{r(f)}(a, X)) = l(a)$, which holds if and only if $\prod_{a \in V(X)} \Theta(B_{r(f)}(a, X)) = 1$. Hence, $\Theta(X) = 1$ implies $X \in T$, and for each n there is a $Y \in \mathbb{C}_{n,S}$ such that $\Theta(Y) = 1$. Thus Θ is a local check scheme for T , and $LCR(T) \leq r(f)$. ■

Proposition 5 implies that for any pattern T that is solvable on all initial conditions, there is a local check scheme Θ of some finite radius for T . This establishes a necessary condition for the existence of local solutions to a pattern T : that $LCR(T) < \infty$. This simply describes the fact that for a problem to be solvable by local rules, the stopping condition of being inside the pattern must be locally recognizable.

Example 12 The pattern T_{1000} is 2-locally checkable, with local check $\Theta(b)$ given by

- $\Theta(b) = 1$ if $b = 010^*00, 001^*00, 100^*01$, or 000^*10 , arising as a central ball in a large configuration. (See the discussion in §1.1 for clarification of “small” vs. “large” configurations.)
- $\Theta(b) = 1$ if $b = 1^*000, 10^*00, 100^*0$, or 1000^* , arising as a left- or right-end balls in a medium or large configuration, or a small configuration.
- $\Theta(b) = 0$ in all other cases.

A slight generalization of the above construction shows that **all Repeat Patterns Are Locally Checkable**:

Proposition 6 The repeat pattern T_q has local check radius $r(T_q) \leq |q|/2$.

On the other hand, a radius of 1 with two states is insufficient to provide an LCS for the T_{1000} pattern. The argument for this is very similar to the proof of proposition 3 in §1.5. Suppose Θ were a putative radius-1 LCS. The radius-1 neighborhoods include $10^{star}0, 00^*0, 00^*1$, and 01^*0 . Any LCS for T_{1000} would have to accept 000 , but then it would also have to accept 0 strings of any length, contradicting the first condition on an LCS. Hence, $LCR(T_{1000}) = 2$. There are repeat patterns for which $LCR(T_q)$ is strictly less than $|q|/2$.

The proof of proposition 4 of §1.5, that proportionate patterns are not locally solvable, is also easily adapted to show that proportionate patterns are not locally checkable either. That argument shows that **proportionate patterns are not locally checkable**:

Proposition 7 For any nontrivial proportionate pattern T_F , $LCR(T_F) = \infty$.

The local checkability criteria therefore subsumes all the results of section 1.5.

2.2 Local Checks Are Part Lists

Since a local check scheme Θ is a function from the set of r -balls $\mathcal{B}_{r,S}$ to $\{0, 1\}$, the inverse of 1, that is $\Theta^{-1}(1)$, is a subset of $\mathcal{B}_{r,S}$. $\Theta^{-1}(1)$ is the set of local r -ball configurations that Θ “accepts.” Let

$$\Theta(\mathbb{C}) \triangleq \{X \in \mathbb{C} \mid \Theta(X) = 1\}$$

be the set of full configurations accepted by Θ , which we’ll call the Θ -admissible configurations. As a set of configurations, $\Theta(\mathbb{C})$ is by definition a pattern. If T is any pattern and Θ is a local check scheme for it, then by definition $\Theta(\mathbb{C}) \subset T$ and $B_r(X) \subset \Theta^{-1}(1)$. Hence:

$\Theta^{-1}(1)$ should be thought of as a list of valid local parts, and the pattern $\Theta(\mathbb{C})$ generated by Θ is the set of one-dimensional structures that can be built from putting those local parts together.

Example 13 For example, suppose Θ is a local check of radius 3 such that $\Theta^{-1}(1)$ contains precisely the balls: 1^*000 , 10^*00 , 100^*0 , 1000^* , 10^*001 , 100^*010 , 1000^*100 , 0001^*000 , 0010^*001 , 0100^*010 , 1000^*100 , 0010^*00 , 0100^*0 , 1^*00 , 10^*0 , 100^* , 1001^*001 , 0010^*010 , 0100^*100 , 1001^*00 , 0010^*0 , 0100^* and 1001^*000 . One can (laboriously) check that $\Theta(\mathbb{C}) = \{(100)^n(1000)^m \mid n, m \in \mathbb{N}\}$.

2.2.1 Generating Configurations

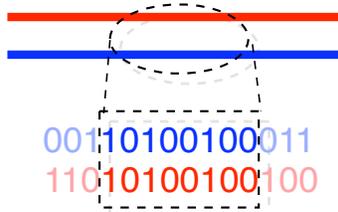
Given a configuration X , define

$$B_r(X) = \{B_r(i, X) \mid i = 1, \dots, |X|\},$$

the set of r -balls in X , or put another way, the parts used in X . For two configurations X and Y , the common parts are the elements of $B_r(X) \cap B_r(Y)$. The basic fact about local checkability is:

Proposition 8 Θ -admissible configurations can be “spliced” at common parts.

Proof: Suppose X, Y are two Θ -admissible configurations with a common part $z \in B_r(X) \cap B_r(Y)$ arising at position i in X and position j in Y :



By definition, for all positions $i \in \{1, 2, \dots, |X|\}$, Θ maps the local view around agent i to 1, i.e. $\Theta(B_r(i, X)) = 1$. The same holds for Y . Hence, for all r -balls (“parts”) $z \in B_r(X) \cup B_r(Y)$, $\Theta(z) = 1$.

Now consider the splicing $X' = X(1 : i+r) \circ Y(j+r+1 : |Y|)$. The first i r -balls in X' and X are the same, while the remaining parts are identical to the corresponding parts in Y . Hence, $B_r(X') = B_r(X(1 : i)) \cup B_r(Y(j+r+1 : |Y|))$. Similarly, for $Y' = Y(1 : j+r) \circ X(i+r+1 : |X|)$, we have $B_r(Y') = B_r(Y(1 : j)) \cup B_r(X(i+r+1 : |X|))$.



Hence

$$B_r(X') \cup B_r(Y') = B_r(X) \cup B_r(Y).$$

That is, X' and Y' have the same local parts as X and Y . Thus by definition X' and Y' are Θ -admissible. ■

At any fixed radius r there are finitely many possible valid parts. in fact, $B_r(X) \subset \Theta[m]$, and so

$$\left| \bigcup_{X \in \Theta(\mathbb{C})} B_r(X) \right| \leq |\Theta[m]| = |S|^{2r(\Theta)+1}.$$

Now suppose that there is a configuration $X \in \Theta(\mathbb{C})$ with $|X| > |S|^{2r+1} + 2r + 1$. Because X has at least $|S|^{2r+1}$ central positions (the other $2r + 1$ account for the left and right ends), there are in X at least $|S|^{2r+1}$ positions with index between r and $|X| - r$, more than the number of distinct possible r -balls. Hence, there is a repeated part, that is, $r + 1 \leq i, j \leq |X| - r$ such that $X(i - r : i + r) = B_r(i, X) = B_r(j, X) = X(j - r : j + r)$. But then by repeated application of proposition 8,

$$X(1 : i - 1) \circ X(i : j)^n \circ X(j + 1 : |X|)$$

is Θ -admissible for all n (where z^n means z concatenated with itself n times). Of course, given any integers k and m , n can be chosen such that $|X| + n(j - i) = k'm$ for some $k' \geq k$.

Recall the definitions of minimal segments, and admissible sizes from §1.3. The discussion in the previous paragraph proves:

Proposition 9 *A locally checkable pattern $T = \Theta(\mathbb{C})$ is infinite if and only if:*

- *It contains a configuration of size greater than $|S|^{2r(\Theta)+1} + 2r(\Theta) + 1$,*
- *It contains at least one minimal segment (and all minimal segments have size at most $|S|^{2r(\Theta)+1}$).*
- *Sizes(T) contains infinitely many multiples of every integer.*

In §1.3 I also introduced various scale properties, like repeatability and expandability. It is shown in appendix §A.2) that in locally checkable patterns, these properties reduce to their application to sets of local parts. Because a local check scheme is a 1-D tile set or a set of local parts, the generic pattern properties that *a priori* require considering all the elements of a pattern as a whole, actually are equivalent to considerations about the local part set.

Now, recall from §1.1 that, with respect to a choice of radius r , there are two basic kinds of configurations: large configurations, of size $|X| > 2r$, and small configurations of size $\leq 2r$. For any given local check scheme Θ , describing the Θ -admissible configurations can be split into the problems of describing the small- and large-size Θ -admissible configurations separately. The situation for small-size configurations is simple. *Any* subset of configurations of size $\leq 2r$ is realizable by some local check of radius r . This is a simple consequence of the fact that the local ball around any agent is necessarily distinct from the local balls around all the other agents. Generating configurations of this size consists of independently picking a set of accepted parts for each distinct agent position. Thus, finding the small Θ -admissible configurations is a trivial matter of making a (finite) list.

However, for large configurations with size $2r(\Theta) + 1$ or larger, there are some dependencies between the possible configurations. Determining the large Θ -admissible configurations is a question of figuring out the generic ways in which the accepted parts fit together. If some $B \in \Theta^{-1}(1)$ actually appears in a Θ -admissible configuration, that would mean that there are sufficient other Θ -accepted balls that fit together to complete a full configuration. There would have to be a string of Θ -accepted balls off to the left of B that terminated in Θ -accepted left balls, and a string of Θ -accepted balls off to the right that terminated in Θ -accepted right balls. (For discussion on left-, right- and central balls, see §1.1 or the appendix of chapter 1.)

Now, this may not be possible. In other words, there might be some Θ -accepted balls B that cannot appear in any full configuration because they cannot be completed in both directions by other Θ -accepted balls. I call these balls *spurious*. Given a local check scheme, we can ignore the spurious local views without affecting the set of configurations that can be generated from it. Henceforth, we will assume that spurious views have been removed.

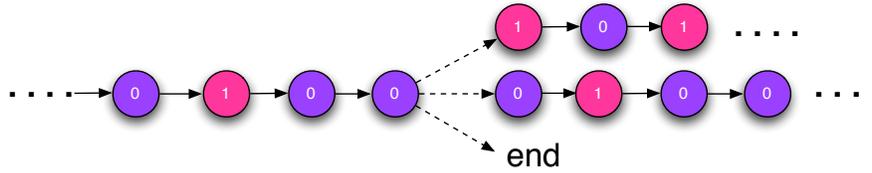
Non-spurious Θ -accepted balls will be connectable to one or more other local balls on either side. Consider the local check scheme from example 13 above.

Example 14 • The left-end ball $b = 1^*000$ can be connected to no other balls to the left, and to one other Θ -accepted ball to the right, namely 10^*001 .

- The right-end ball $b = 0100^*$ can be connected to no other balls to the right, and to one other ball to the left, namely 0010^*0 .

- The central ball $b = 0010^*010$ can be connected to exactly one other ball on the left and right (1001^*001 and 0100^*100 , respectively). The same applies to the ball $b = 0100^*010$, for instance.
- The central ball $b = 0100^*100$ can be connected to one Θ -accepted ball to the left (namely, 0010^*010), and to three distinct Θ -accepted balls to the right – namely, the central balls 1001^*000 and 1001^*001 and the right-end ball $1001^{star}00$.

Given a Θ -accepted ball B , let's informally define the number of Θ -accepted parts connectable to the left (resp. right) of B as the *left (resp. right) compatibility index* of B . The importance of these quantities for each accepted ball Θ is that they control the number of possible Θ -admissible configurations consistent with a given local ball. Any ball with left (resp. right) compatibility index of 1 has a unique predecessor (resp. successor). In the case of the ball $b = 0100^*100$ in the above example, the significance of having right compatibility index of 3 is that it has three possible successors:



In the middle option, the configuration switches from the 100 repeat motif, to the 1000 repeat motif. In the upper option, the 100 motif is continued. In the lower option, the configuration ends.

Formal Definition of Left- and Right- Compatibility

Defining the left- and right- compatibility indices properly is important, but slightly tricky. Let b be any m -ary string of length at most $2r + 1$. We first define a boolean $\Theta[b]$ that is 1 whenever considered as an r -ball, b is accepted by Θ , or can be extended to the left so as to be. Formally: (I) For any m -ary string b whose length $|b|$ satisfies $r(\Theta) + 1 \leq |b| \leq 2r(\Theta) + 1$, define $\Theta[b] = 1$ if the r -ball $(b, |b| - r + 1)$ is in $\Theta^{-1}(1)$. (II) For any m -ary string of length $\leq r(\Theta)$, define $\Theta[b]$ to hold when there is some m -ary string b' of length $r - |b|$ such that the r -ball $(b \circ b', 1) \in \Theta^{-1}(1)$. In all other cases, let $\Theta[b] = 0$. This boolean $\Theta[b]$ is useful here, and in the following section as well.

Now, define $R[b]$ as follows: (I) Given an m -ary string b of length $2r(\Theta) + 1$, let $R(b)$ be the set of states i such that the $\Theta[b[2 : 2r + 1] \circ i]$ holds. (II) Given an m -ary string b of length $< 2r(\Theta) + 1$, let $R(b)$ be the set of i such that $\Theta[b \circ i]$ holds. The size of the set $|R(b)|$, by definition, the right compatibility index of b .

Analogously, define $L[b]$ as follows: (I) Given an m -ary string b of length $2r(\Theta) + 1$, let $L(b)$ be the set of states i such that $\Theta[i \circ B[1 : 2r]]$ holds; (II) Given an m -ary string b of length $\geq r + 1$ and $< 2r(\Theta) + 1$, let $L(b)$ be the set of i such that $(i \circ b, |b| - r + 1)$ is in $\Theta^{-1}(1)$; and (III) Given an m -ary string b of length $r(\Theta)$ or less, define $L(b)$ to be the set of i such that for some m -ary string b' of length $r - |b| - 1$, the r -ball $(b' \circ i \circ b, 1) \in \Theta^{-1}(1)$. The size of the set, $|L(b)|$, is, by definition, the left compatibility index of b .

2.3 LCSs are Graphs

It turns out that a much more comprehensive and powerful set of computations can be made, with a bit of extra work. In fact, all local check schemes of a fixed radius are equivalent to graphs – and the graph theoretic properties of a LCS's associated graph determine much about properties of the pattern generated by the LCS.

Let $\mathcal{B}_{r,S}$ be, as defined in §1.1, the set of local balls of radius r with states in the set S . Now, define $\mathbb{D}(r, m)$ as the directed graph

$$\mathbb{D}(r, m) = (V, E)$$

where $V = \mathcal{B}_{r,S}$, and where there is an edge $(b_1, b_2) \in E$ if and only if there is a configuration $X \in \mathbb{C}_S$ containing b_1 directly adjacent to the left of b_2 .

Two balls can only be adjacent in a configuration if they differ by at most one in size and have coincident states at all common positions, i.e. if b_1 is adjacent to the left of b_2 , then then $|b_2| \geq |b_1| - 1$, and the states at

the right-most $|b_1| - 1$ positions of the ball to the left must equal the left-most $|b_1| - 1$ states of the ball on the right. Hence,

$$\mathbb{D}(r, m) = \left(\mathcal{B}_r, \{(b_1, b_2) \in \mathcal{B}_r^2 \mid b_1(1 + \max(0, \star(b) - r) : |b_1|) = b_2(1 : |b_2| - \max(0, \star(b_2) - r))\} \right).$$

A local check scheme Θ of radius r (and m states) defines two classes of balls: $\Theta^{-1}(1)$, the accepted parts, and $\Theta^{-1}(0)$, the invalid parts. Of course, $\Theta^{-1}(0) = \mathcal{B}_r \setminus \Theta^{-1}(1)$, since any local ball in \mathcal{B}_r is either valid or not. Hence, $\Theta^{-1}(1)$ determines Θ . On the other hand, $\Theta^{-1}(1) \subset \mathcal{B}_r$, so now simply assign to Θ the induced subgraph of $\mathbb{D}(r, m)$ whose nodes are $\Theta^{-1}(1)$. We thus have:

Proposition 10 *Local check schemes of radius r are in 1-1 correspondence with subgraphs of $\mathbb{D}(r, m)$, via the assignment*

$$\Theta \mapsto G(\Theta) = \left(\Theta^{-1}(1), \text{induced edges from } \mathbb{D}(r, m) \right).$$

Example 15 Take the repeat pattern T_{1000} and the local check for it described in example 12: $\Theta(b) = 1$ for b equal to:

- 010*00, 001*00, 100*01, and 000*10 (from the central balls)
- 1*000, 10*00, 100*0, and 1000* (from right- and left-end balls)

The associated graph $G(\Theta)$ is:

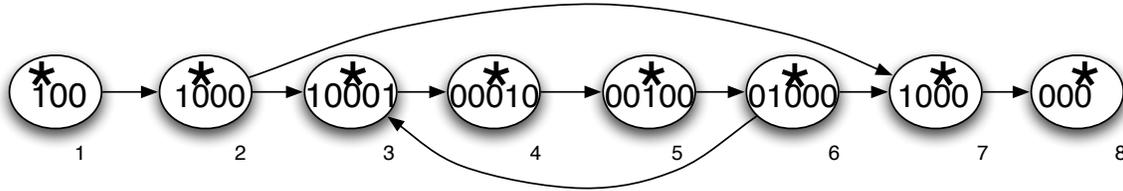


Figure 2.1: Graph associated with the pattern $\{(1000)^n \mid n \in \mathbb{N}\}$.

This graph has eight nodes and one cycle, of length 4. The paths in this graph can take either two routes: one is to bypass the 4-cycle, along the path (1, 2, 7, 8). This path corresponds to the configuration 1000, the size-4 version of the pattern. The other route is to go through the 4-cycle, repeating it some integral number of times n , along the path (1, 2, (3, 4, 5, 6)ⁿ, 7, 8). This path corresponds to the configuration 10(0010)ⁿ00, the size $4(n + 1)$ version of the pattern.

Example 15 suggests that the graph structure of $G(\Theta)$ determines properties of the pattern $\Theta(C)$ generated by Θ . We will now go through the basic properties of directed graphs and show how each can be interpreted as a property of the corresponding pattern.

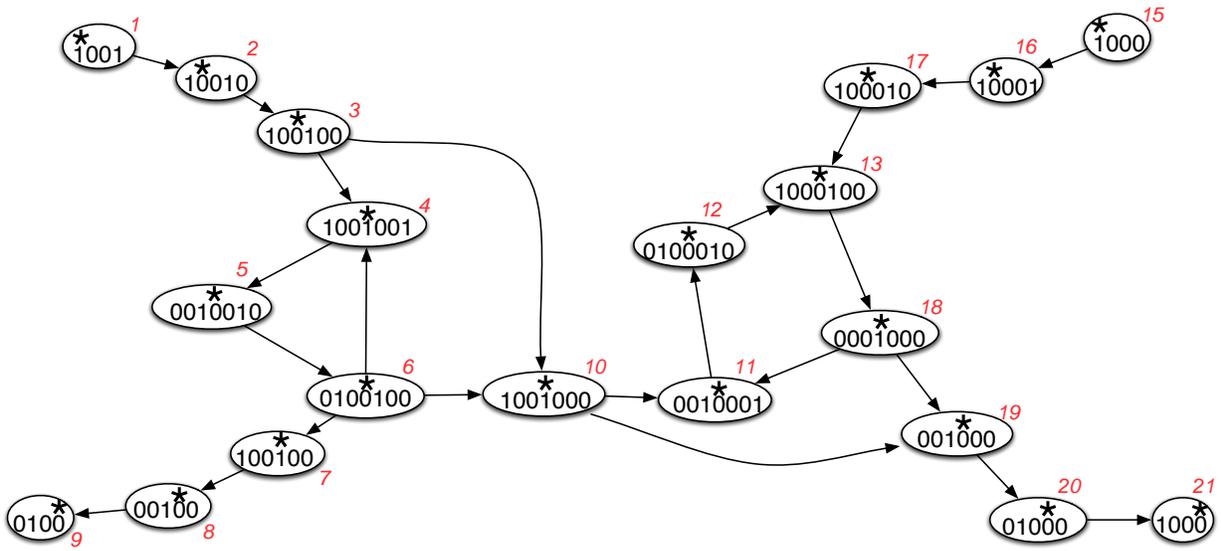
2.3.1 Meaning of the Graph Structure

Nodes and Degrees

The most basic property of a graph is its number of nodes. Notice that the local check scheme defined in example 15 has 8 nodes, each one corresponding to one of the 8 valid views (i.e. 010*00, 001*00, &c) listed just above as defining the local check. Generally:

The number of nodes of $G(\Theta)$ is equal to $|\Theta^{-1}(1)|$, the number of valid parts in Θ .

Another basic notion in a directed graph are indegrees and outdegrees. Recall in §2.2.1 we defined the left and right compatibility index of a Θ -accepted ball b to the number of distinct balls that could be placed validly directly adjacent to b on the left or the right, respectively. In example 13, I described a radius-3 local check scheme Θ for the pattern $\{(100)^n(1000)^m \mid n + m \geq 2\}$. I then illustrated local balls in Θ that have various left- and right- compatibility indices. For instance, the balls $b = 0010^*010$ and $b = 0100^*010$ both have left


 Figure 2.2: Graph associated with $\{(100)^n(1000)^m | n + m \geq 2\}$ pattern.

and right compatibility indices of 1; while $b = 0100*100$ has left compatibility index of 1 and right index of 3.

The graph associated with this local check scheme is depicted in figure 2.2. Notice that indegrees and outdegrees of the nodes associated with $0010*010$ and $b = 0100*010$ are all 1, while the node associated with $0100*100$ has indegree 1 and outdegree 3. Generally,

The left and right compatibility indices of a ball $b \in \Theta^{-1}(1)$ are equal, respectively, to the indegree and outdegree of b considered as a node in $G(\Theta)$.

A node in a directed graph is *initial* if its indegree is 0, and *terminal* if its outdegree is 0. Recall that an r -ball has, as shown in the examples above, a \star located somewhere, marking the location of the agent with whom the ball is associated. Evidently, the general rule is that:

An initial node in $G(\Theta)$ corresponds to a ball such for which $\star(b) = 1$, while a terminal node has $\star(b) = |b|$.

Paths are (Sub)configurations

Another basic notion in a directed graph is a path. In example 15, we saw how paths in the graph correspond configurations consistent with the $10(0010)^n00$ pattern. More generally, suppose we're given a path in $\mathbb{D}(r, m)$ of length N :

$$P = (p_1, p_2, \dots, p_N)$$

where each p_i is a node in $\mathbb{D}(r, m)$. By definition of $\mathbb{D}(r, m)$ each node p_i corresponds to r -ball, and any r -ball corresponds to an m -ary sequence of some length at most $2r + 1$. We can stitch the m -ary sequences together to make an m -ary sequence S_P . To do this, take the first $\star(p_1)$ states in p_1 , i.e.

$$X_P(1 : l_{\star}(1)) = p_1(1 : l_{\star}(1)).$$

Since p_2 is linked to from p_1 , it must be the case that positions 2 through $\star(p_1) + 1$ in p_1 coincide with positions 1 through $\star(p_2)$ in p_2 . Thus, just add on $p_2(\star(p_2))$,

$$X_P(1 : l_{\star}(1) + 1) = p_1(1 : l_{\star}(1)) \circ p_2(l_{\star}(2)),$$

to make a sequence of length $\star(p_1) + 1$. The same point holds for $p_3, p_4, \&c.$ This can be repeated, until we reach p_{N-1} :

$$X_P(1 : \star(p_1) + N - 2) = p_1(1 : \star(p_1)) \circ p_2(\star(p_2)) \circ \dots \circ p_{N-1}(\star(p_{N-1})).$$

Finally, we finish by adding the whole segment of p_N from $\star(p_N)$ to its right end:

$$X_P(1 : \star(p_1) + N - 2) = p_1(1 : \star(p_1)) \circ p_2(\star(p_2)) \circ \dots \circ p_{N-1}(\star(p_{N-1})) \circ p_N(\star(p_N) : |p_N|),$$

creating a subconfiguration of size $n + |p_n| + \star(p_1) - \star(p_N) - 1$. X_P is a full configuration when P is a path from an initial node to a terminal node, i.e. a so-called *maximal* path.

This construction can evidently proceed in the opposite direction, associating to every (sub)configuration X a path X_P . Thus

Subconfigurations are in 1-1 correspondence with paths in $\mathbb{D}(r, m)$ and configurations in 1-1 correspondence with maximal paths. Paths in $G(\Theta)$ correspond to Θ -consistent (sub)configurations.

Connectedness is Connectability

For locally checkable patterns, connectability of pattern subconfigurations translates directed into connect-ness properties of the $G(\Theta)$ graph.

Definition 19 [Connectedness] Given a directed graph G , a node x is connected to y in G if there is a path p in G whose first node is x and whose last node is y . A path p_1 is connected to a path p_2 if the terminal node of p_1 is connected to the initial node of p_2 .

Note that x being connected to y in a directed graph G does not imply that y is connected to x . This fact is the same as noticing that even when configuration X is connectable to Y , the reverse may not obtain. We say x and y are mutually connected if x is connected to y and vice versa.

Consider for example the pattern $T_{100} \cup T_{1000}$, consisting of the length $3n$ repeats of the unit 100, and the length $4n$ repeats of the unit 1000. The graph associated with the (radius 3) check scheme for this pattern is:

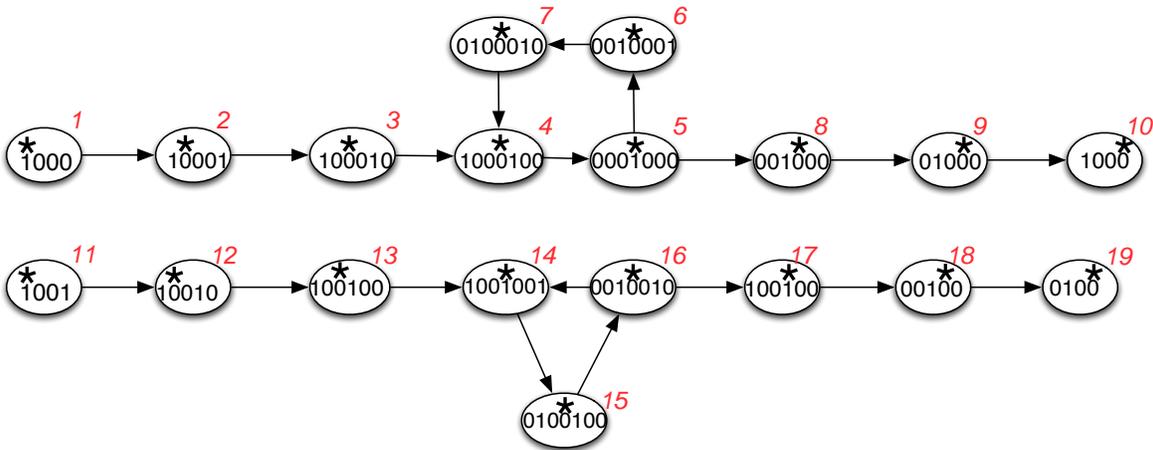


Figure 2.3: $T_{100} \cup T_{1000}$.

The fact that the two groups of nodes (1-10 and 11-19) have no connection corresponds to the fact that the pattern admits no combinations of the two repeated segments 100 and 1000.

Now consider the paths $p_1 = (1, 2, 3, 4)$ and $p_2 = (18, 19, 20, 21)$ in the graph in figure 2.2. They are connected in $G(\Theta)$; which corresponds to the fact that there is at least one Θ -admissible configuration in which p_1 arises before p_2 (for example, 1001000). However, they are not mutually connected. Generally we have:

A path p_1 in $G(\Theta)$ being connected to p_2 as paths translates to corresponding configurations X_{p_1} being connectable to X_{p_2} , in the language of definition 12. X and Y are alternatable if the corresponding paths P_X and P_Y in $G(\Theta)$ are mutually connected.

Cycles and Repeatability

For locally checkable patterns, expandable and repeatable subconfigurations and minimal segments translate into properties about the cycles of the graph $G(\Theta)$.

Definition 20 [Cycles] A cycle in a directed graph is a path C whose initial node is the same as its terminal node, and such that no two nodes in C (besides the first and last) are the same. A cycle C in G is irreducible if the graph induced by G on the nodes of C is in fact the cycle C itself. In other words, C admits no other internal edges besides the ones in the cycle itself. A graph is acyclic if it contains no cycles.

Example 16 Consider, for instance:

- In example 15, the 4-cycle is irreducible, and corresponds to the minimal segment (1000).
- In the local check scheme depicted in figure 2.2, there are two irreducible cycles: (4,5,6) and (11,12,13,18). They correspond to the minimal segments 100 and 1000, respectively.
- In the local check scheme depicted in figure 2.5, there are three irreducible cycles: (9,10), (7,8,4,17,21,15,16,10), and (17,23,24,25). They correspond to the segments 10, 10100010 and 1000, respectively. These are the only three segments consistent with the check scheme. All other repeated elements (i.e. 1010, 10001000, &c) consist of combinations of these three.

Generally:

An irreducible cycle in $G(\Theta)$ corresponds to a minimal segment in the pattern $\Theta(C)$. A reducible cycle in $G(\Theta)$ is a repeatable configuration composed of several segments. A subconfiguration X in $\Theta(C)$ is repeatable if P_X is contained entirely in a cycle in $G(\Theta)$. X is expandable if P_X is connectable to a cycle. $G(\Theta)$ being acyclic is equivalent to $\Theta(C)$ being finite.

This is summarized by the figure:

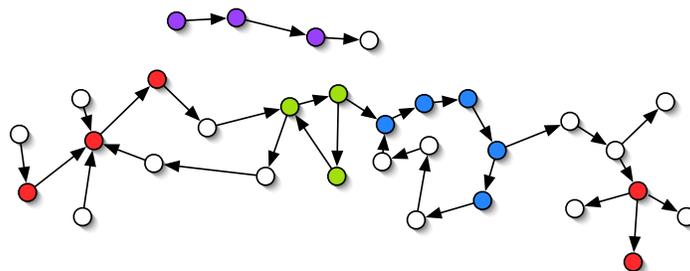


Figure 2.4: Red is expandable, green is a segment, blue is repeatable, purple is unexpandable.

It is shown in chapter 7 that every cycle in $\mathbb{D}(r, m)$ can be decomposed uniquely into a union of disjoint irreducible cycles. Hence, every repeatable configuration can be decomposed into its disjoint irreducible subconfigurations.

Connected Components

Definition 21 [Strongly Connected Components] A strongly connected component (SCC) of a directed graph is a maximal subgraph $S \subset G$ such that for every pair nodes $x, y \in S$ is mutually connected, i.e. there is a path in G from x to y AND vice versa.

Every node of a directed graph is in some strongly connected component, although the component might contain only that one node. Moreover, if nodes x, y and y, z are in common SCCs, then so must x and z be. Thus, the SCCs partition a graph.

Example 17 Consider the pattern T generated by freely alternating the segments 1000 and 10, i.e.

$$T = 101000, 100010, 101010, 10001000, 10100010, \&c.$$

This pattern has local check radius 3, graph of the local check scheme of radius 3 corresponding to T is depicted thusly:

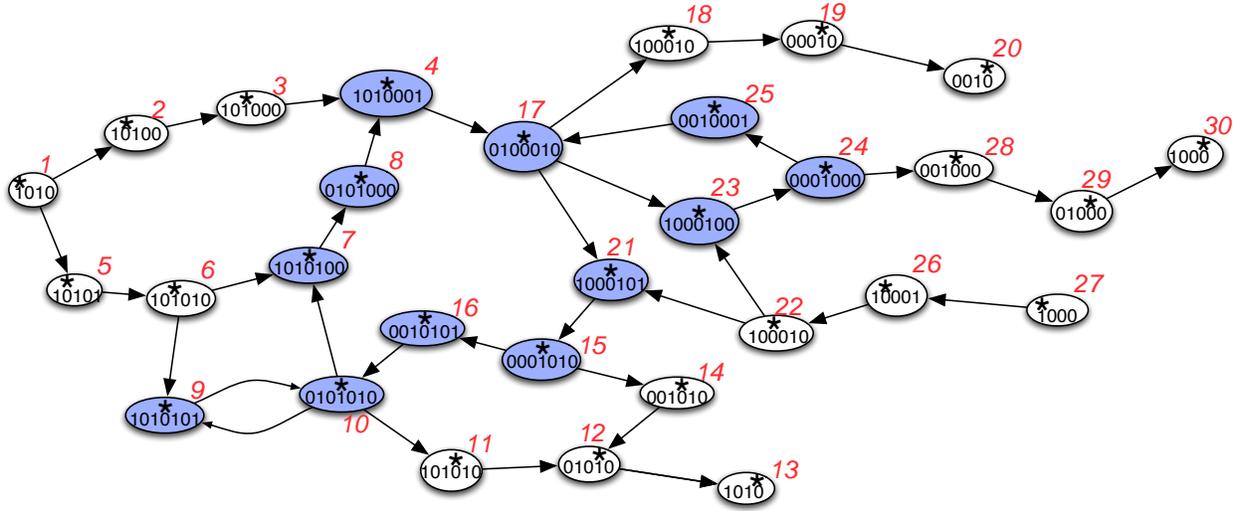


Figure 2.5: Pattern generated by free alternation of the words 10 and 1000.

By inspection, it's possible to see that the only non-trivial strongly connected component in that graph consists of the nodes 4, 7, 8, 9, 10, 15, 16, 17, 21, 23, 24, and 25 (shaded light blue). These nodes correspond to the subconfigurations in which the segment 10 is alternated with the segment 1000.

Generally, we have:

A Strongly Connected Component of $G(\Theta)$ corresponds to a maximal set of alternatable subconfigurations in $\Theta(\mathbb{C})$.

A *weakly connected component* (WCC) is a maximal subgraph of $S \subset G$ such that every pair of nodes $x, y \in S$ has either x connected to y OR vice versa. For example, in the local check scheme depicted in figure 2.3, there are two weakly connected components (nodes 1-10 and 11-19).

Example 18 In addition, consider the pattern

$$T_{1000} \cup (T_{1000} \circ 1) \cup (T_{1000} \circ 10) \cup (T_{1000} \circ 100)$$

in which for any pattern T and m -ary string z , $T \circ z$ refers to the pattern $\{x \circ z | x \in T\}$. The graph associated with the radius 3 check scheme for this pattern is shown in figure 2.6. This pattern has four weakly connected components, consisting of the nodes sets $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, $\{1, 2, 3, 4, 8, 9, 10, 11, 12, 13\}$, $\{1, 2, 3, 4, 8, 9, 10, 14, 15, 16\}$ and $\{1, 2, 3, 4, 8, 9, 10, 17, 18, 19\}$.

Overall Decomposition

Let $\mathbf{G} = \{S_i\}$ the set of SCCs in G . Define a binary relation $<_G$ on \mathbf{G} in which $S_i <_G S_j$ iff $S_i \neq S_j$ and there is a path from S_i to S_j in G . Because of the definition of strongly connected, $\mathcal{G} \triangleq (\mathbf{G}, <_G)$ is a partial ordering. A WCC can be thought of as linear subordering of \mathcal{G} .

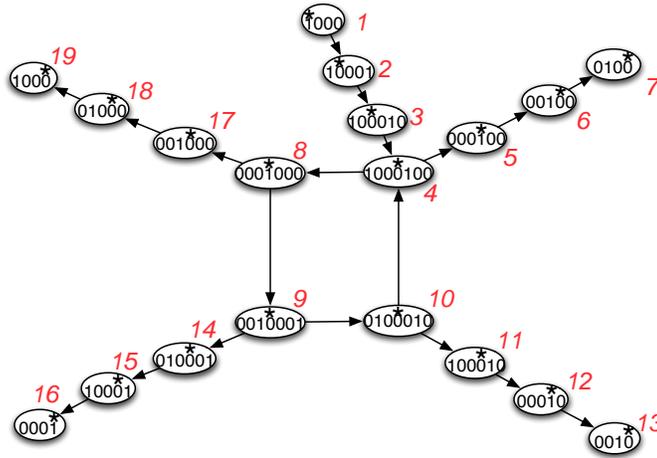


Figure 2.6: $T_{1000} \cup (T_{1000} \circ 1) \cup (T_{1000} \circ 10) \cup (T_{1000} \circ 100)$.

Given a directed graph G , a *maximal acyclic path* (MAP) in G a path P containing no cycles and such that any path in G containing P properly has a cycle. For example, in the graph depicted in figure 2.6, the MAPs are $(1, 2, 3, 4, 5, 6, 7)$, $(1, 2, 3, 4, 8, 17, 18, 19)$, $(1, 2, 3, 4, 8, 9, 14, 15, 16)$, and $(1, 2, 3, 4, 8, 9, 10, 11, 12, 13)$.

In terms of $G(\Theta)$, the overall picture is of a set of SCCs correspond to islands of configurations composed of alternatable irreducible cycles bridged by unrepeatable configurations corresponding to sections of a maximal acyclic path; and every SCC is in one or more WCC, each of which corresponds to a linear ordering of non-alternatable blocks:

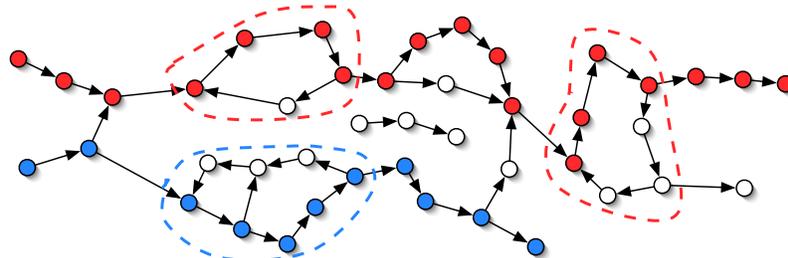


Figure 2.7: Maximal acyclic paths P_{red} and P_{blue} with $SCC(P_{red})$ and $SCC(P_{blue})$ outlined.

2.3.2 Admissible Sizes

As defined in §1.3.2, we discussed the *admissible sizes* of a pattern – the set of sizes of possible structures in that pattern. We can use the graph structure $G(\Theta)$ to compute the set of admissible sizes of the pattern $\Theta(\mathbb{C})$.

Given a directed graph G , define $A(G)$ to be the set of maximal acyclic paths in G . For any maximal acyclic path in G , define $SCC(P) = \bigcup_{x \in P} SCC(x, G)$, the set of all strongly connected components that P passes through. Let $IrrCyc(P)$ be the set of irreducible cycles in all these SCCs. Let $E(P) = \{|c|, c \in IrrCyc(SCC(P))\}$, the set of sizes of irreducible cycles of $SCC(P)$. For the red path in figure 2.7, $E(P_{red}) = \{5, 7\}$, while for the blue path $E(P_{blue}) = \{4, 8\}$. Given a set of integers like $E(P)$, we can compute its least common multiple, $L(P) = lcm(E(P))$ and its greatest common divisor $T(P) = gcd(E(P))$. For example in figure 2.7, $L(P_{red}) = 35$ and $T(P_{red}) = 1$, while $L(P_{blue}) = 8$, and $T(P_{blue}) = 4$.

The arithmetic progression denoted $A(a_0, d)$ is the set of integers $\{a_0 + md | m \in \mathbb{N}\}$. a_0 is called the initial value and d the common difference. For example $\{3, 8, 13, 18, 23, 28, \dots\}$ is $A(3, 5)$.

Proposition 11 Let $K(P) = \max_{P \in A(G)} L(P)$. Then

$$\text{Sizes}(\Theta) \cap \{K, K+1, \dots, \infty\} = \bigcup_{P \in A(G(\Theta))} A(|P|, T(P)).$$

That is:

Ignoring a finite set of size controlled the least common multiple of the cycle sizes, the set of Θ -admissible sizes is a union of finitely many arithmetic progressions whose initial values are the lengths unrepeatable patterns in $\Theta(\mathbb{C})$ and whose common differences are the greatest common divisor of the sizes of the repeatable segments connected to those unrepeatable patterns.

Proof: For each maximal acyclic path P , consider

$$\mathbb{P}_P = \{X \in \Theta(\mathbb{C}) \mid \forall x \in X, B_{r(\Theta)}(x) \subset \text{SCC}(P)\}.$$

This is the set of configurations that can be built using parts in $\text{SCC}(P)$. Let $\{C_1, \dots, C_{n(P)}\}$ enumerate the elements of $\text{IrrCyc}(P)$, the irreducible cycles in $\text{SCC}(P)$, where $n(P) = |\text{IrrCyc}(P)|$, the number of SCCs through which P travels. Since $\text{SCC}(P)$ is a linear subordering of \mathcal{G} , number the components of $\text{IrrCyc}(P)$ with respect to this order. All configurations in \mathbb{P}_P consist of some finite concatenation:

$$X = p_1 \circ l_1 \circ p_2 \circ l_2 \circ \dots \circ l_{n(P)} \circ p_{n(P)+1}$$

where

- The p_i are finite non-repeatable subconfigurations whose concatenation makes up the maximal acyclic path P , $P = p_1 \circ p_2 \dots p_{n(P)+1}$, and
- the l_i are loops contains entirely in the SCC C_i .

Of course,

$$|X| = \sum_i |p_i| + \sum_{i=1}^{n(P)} |l_i| = |P| + \sum_{i=1}^{n(P)} |l_i|.$$

Now, a loop in a strongly connected graph is a concatenation of cycles interspersed in various orders. Because all cycles in $\mathbb{D}(r, m)$ can be written as unions of irreducible cycles, we have

$$|l_i| = \sum_j m_{j,i} |c_{i,j}|$$

where j ranges over the irreducible cycles $c_{i,j}$ contained in C_i and the $m_{j,i}$ are non-negative integers expressing the multiplicity of the repeats of $c_{i,j}$ in the configuration. Now, it may not be possible to chose the $m_{j,i}$ entirely arbitrarily. This is because when the maximal acycle path P “enters” the SCC c_i , the path may have to “go through” one cycle to get to another desired cycle before repeating that descried cycle some arbitrary number of times. However, to achieve the various possible sizes, the transit need only happen once. Hence, tuples of $m_{i,j}$ can be chosen arbitrarily as long as $n_{i,j} > 0$ for all j . Hence, by the chinese remainder theorem, if we define

$$S_i = \text{lcm}(\{|c_{i,1}|, \dots, |c_{i,N_i}|\})$$

and

$$T_i = \text{gcd}(\{|c_{i,1}|, \dots, |c_{i,N_i}|\})$$

where the $c_{i,j}$ ranges over all irreducible cycles in C_i , then the set of sizes of loops l_i is

$$\text{Sizes}(C_i) = A(C_i) \cup T_i \cdot \{S_i, S_i+1, \dots, \infty\}$$

where $A(C_i)$ is a finite set whose maximum element is of size at most $2S_i$.

Thus, the admissible sizes of X in \mathbb{P}_P are

$$|P| + A(P) + \text{gcd}(\{T_1, \dots, T_{n(P)}\}) \cdot \{S(P), S(P) + 1, \dots\}$$

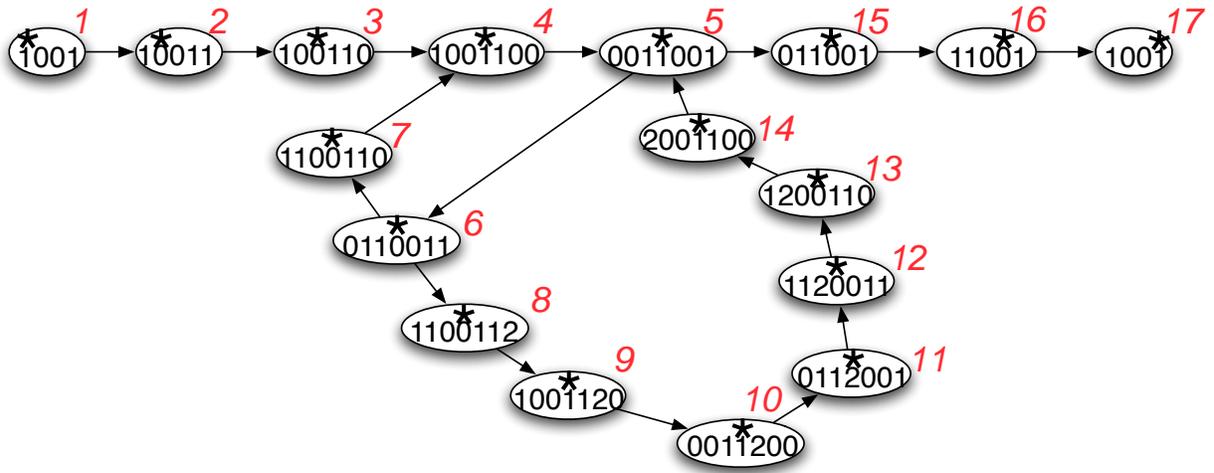


Figure 2.8: Freely generated pattern on the words 1000 and 100112001.

where $A(P)$ is a finite set whose maximum element is at most $2S(P)$ where $S(P) = lcm(S_1, \dots, S_n(P))$. Now, the lcm of a set of lcms of a sets of numbers is the same as the lcm of the union of those sets, and similarly with the gcd. Hence the possible sizes are

$$|P| + A(P) + gcd(|c_1|, \dots) \cdot \{S(P), S(P) + 1, \dots\}$$

where the gcd and lcm are taken over the set of all sizes of irreducible cycles contained in $SCC(P)$.

Now,

$$Sizes(\Theta) = \bigcup_{P \in M(G)} Sizes(\mathbb{P}_P),$$

yielding the result. ■

Example 19 • In the example in figure 2.1, there is one maximal acyclic path of length 8 and a cycle of length. Hence, all sizes are of the form $4n$ for $n \geq 1$.

- In the example in figure 2.2, there are two cycles, one of length 3 and the other of length 4. The two cycles are connected by a maximal acyclic path of length 14, and since their lengths have a greatest denominator of 1 and lcm of 6, the set of admissible sizes contains all integers larger than 19. (From the few other acyclic paths, etc... it actually contains all integers larger than 14.)
- In the example in figure 2.5, there are several maximal acyclic paths, all of even length. There are three irreducible cycles, all attached by a maximal acyclic path, and whose lengths have a greatest common divisor of 2. Hence, all even sizes past a certain size (namely, 10), are admissible.
- Consider the pattern generated by freely alternating combinations of the sequences 1001 and 100112001. The graph $G(\Theta)$ associated with Θ is shown in figure 2.8. $G(\Theta)$ has one maximal acyclic path of length 8, and one strongly connected component with two cycles of lengths 4 and 9 respectively. Because the 4 and 9 are relatively prime and have LCM of 36, the admissible sizes for this pattern include all integers larger than 42 (and some smaller sizes as well).

2.3.3 Pattern Growth

In §1.3.2, we also discussed the *growth function* of pattern, the function describing the number of different elements in the pattern of a given size. Here, we show how to compute the growth function of a locally checkable pattern in terms of the structure of its associated graph.

Let Θ be a local check scheme, and again consider a maximal acyclic path P in $G(\Theta)$, as it winds its way through several strongly connected components. Denote by $N(P)$ the number of strongly connected components that P intersects. Let

$$N(\Theta) = \max_{P \in M(G(\Theta))} N(P).$$

Intuitively $N(\Theta)$ measures the largest number of non-interchangeable modules that can be connected in Θ . For the pattern in figure 2.7, $N(P_{blue}) = 1$, $N(P_{red}) = 2$, and thus $N(\Theta) = 2$.

Now let's focus on the individual strongly connected components $c \in G(\Theta)$. (As we saw above, these are like maximal clusters of alternatable modules.) Consider each one as its own graph. In graph theory there is a useful technique, that connects matrices and graphs. Given a directed graph G , a matrix $A(G)$ is associated to G by letting A_{ij} equal 1 if node i is connected to node j , and 0 otherwise. $A(G)$ is called the *adjacency matrix* of G . Like any other matrix, $A(G)$ has eigenvalues, that are *a priori* complex numbers. It turns out that when G is strongly connected, the largest eigenvalue of $A(G)$, denoted $\lambda_{max}(G)$, is a real number no less than 1, with equality iff G contains a single cycle only.

The two components touching the red path in the above figure have λ_{max} of 1, while the component in the blue path has $\lambda_{max} = 1.1148$. Let

$$\lambda_{\Theta} = \max\{\lambda_{max}(c) | c \text{ is a SCC of } G(\Theta)\}$$

so that for the example in figure 2.7, $\lambda_{\Theta} = 1.1148$.

With these definitions at hand, we can state:

Proposition 12 *If Θ contains no alternatable segments, then*

$$Growth_{\Theta}(n) \sim Cn^{N(\Theta)}.$$

If Θ contains two or more alternatable segments, then

$$Growth_{\Theta}(n) \sim C\lambda_{\Theta}^n.$$

Proof: Given a configuration X , consider X as a path $P_X = (p_1, p_2, \dots, p_{|X|})$ in $\mathbb{D}(r, m)$. Now, let p_1^* be the last instance of the node $P_X(1) = p_1$ in P_X , and excise the portion of P_X up to p_1^* , calling the resulting path P_X^- . Then let p_2^* be the last instance of $P_X^-(2)$, the second node of P_X^- , and let P_X^{-2} be the result of excising the portion of P_X^- between node 2 and p_2^* . Repeat this process until P_X^{-g} is achieved, where g is the number of total distinct nodes of $\mathbb{D}(r, m)$ contained in the path P_X . P_X^{-g} is a maximal acyclic path. Every configuration X is thus associated with a unique maximal acyclic path P , that we'll call the *map*(X).

The number of configurations in $\Theta(\mathbb{C})$ of size n is

$$\sum_{P \in M(G(\Theta))} |\{X | \text{map}(X) = P \text{ and } |X| = n\}|$$

that is, we sum up over maximal acyclic paths the number of configurations of size n associated whose map is X , which set we denote $\Theta_P(n)$.

Given a map P let as above $\{c_1, \dots, c_{n(P)}\}$ enumerate in order the non-trivial SCCs that P goes through. Let q_i be the first node in P in $P \cap c_i$, i.e. the place where P enters the component c_i . Then, the number of configurations X of size n such that $\text{map}(X) = P$ is

$$\sum_{0 \leq j_i, \sum_{i=1}^{n(P)} j_i = n - |P|} N(c_1, p_1, j_1) N(c_2, p_2, j_2) \dots N(c_{n(P)}, p_{n(P)}, j_{n(P)})$$

in which $N(c_i, p_i, j_i)$ is the number of loops of size j_i in the SCC c_i starting at point p_i .

Now, in general, let G by any strongly connected directed graph all of whose cycles can be written as disjoint unions of irreducible cycles. Let $T(G)$ denote the gcd of the set of sizes of irreducible cycles in G . Now enumerate the nodes $\{1, \dots, |G|\}$. Let B be the adjacency matrix of G ordering with this numbering.

The number of loops through a point i of length k , denoted $N(i, k)$, is $B_{i,i}^k$, that is, the i -th diagonal entry of the k -th power of the adjacency matrix. Asymptotically therefore,

$$N(x, k) \sim \begin{cases} K_x \lambda_1(G)^k, & \text{if } k \text{ is divisible by } T \\ 0, & \text{otherwise} \end{cases}$$

where $\lambda_1(G)$ is the largest eigenvalue of B and K_x is a constant depending only on the choice of basepoint x . Hence,

$$|\Theta_P(n)| \sim C_1 \sum_{0 \leq j_i, \sum_{i=1}^{n(P)} j_i = n - |P|} \lambda_1^{j_1} \lambda_2^{j_2} \dots \lambda_{n(P)}^{j_{n(P)}} \delta(j_1, T_1) \delta(j_2, T_2) \dots \delta(j_{n(P)}, T_{n(P)})$$

where λ_i is the top eigenvalue of the adjacency matrix of the SCC c_i , $\delta(a, b)$ is the indicator function which is 1 when a is divisible by b , and C_1 is the product $\prod_i K_{q_i}$ of the constants taken at each entry point q_i .

Let $L(P) = \text{lcm}(T_1, \dots, T_{n(P)})$ and $T(P) = \text{gcd}(T_1, \dots, T_{n(P)})$. Write $n - |P|$ in the form

$$n - |P| = mL(P) + jT(P) + l$$

for $j < L(P)/T(P)$ and $l < T(P)$. It is not hard to see that the above sum is 0 when l is non-zero, and otherwise is

$$C_1 C_2(j) \sum_{0 \leq j_i, \sum_{i=1}^{n(P)} j_i = mL} \lambda_1^{j_1 L} \lambda_2^{j_2 L} \dots \lambda_{n(P)}^{j_{n(P)} L}$$

where $C_2(j)$ is a constant that depends only on j . Hence,

$$\sum_{n \in [Lm+1, L(m+1)]} |\Theta_P(n)| \sim C_1 C_2 \sum_{0 \leq j_i, \sum_{i=1}^{n(P)} j_i = mL} \lambda_1^{j_1 L} \lambda_2^{j_2 L} \dots \lambda_{n(P)}^{j_{n(P)} L}$$

where $C_2 = \sum_{j=0}^{L(P)/T(P)} C_2(j)$. (It's not too hard to evaluate this constant explicitly in terms of lcms and gcds, but this is not useful for our purposes.)

Now there are two cases: (i) when all of the top eigenvalues λ_i are 1, and (ii) otherwise. The top eigenvalue of a strongly connected graph is always real; in fact, it is always at least 1, with equality iff the graph is ring. In the case that all SCCs attached to P have only a single cycle, then

$$\sum_{n \in [Lm+1, L(m+1)]} |\Theta_P(n)| \sim C_1 C_2 \sum_{0 \leq j_i, \sum_{i=1}^{n(P)} j_i = mL} 1 = \binom{m + n(P)}{n(P) - 1}$$

so that

$$\sum_{j \leq n} |\Theta_P(j)| \sim C n^{n(P)}$$

where C is a constant depending simple number-theoretic properties of the cycle lengths. Hence, the growth is polynomial with growth order $n(P)$ and constant C .

Now consider the case when the strongly connected components in P contains more than 1 cycle. Assuming for simplicity that none of the λ_i equal 1 and all are distinct and ordered according to decreasing size so that $\lambda_1 = \max\{\lambda_i\}$, then

$$\sum_{n \in [Lm+1, L(m+1)]} |\Theta_P(n)| \sim C \left(\frac{\lambda_1^{Lm+1} - 1}{\prod (\lambda_i - 1)} + \lambda_1^{Lm} \sum_{i \geq 2} \frac{\lambda_i^{n(P)-1}}{(\lambda_i - 1) \prod_{j \neq i} (\lambda_i - \lambda_j)} \frac{1 - (\lambda_i/\lambda_1)^{Lm+1}}{1 - (\lambda_i/\lambda_1)} \right).$$

Thus

$$\sum_{j \leq n} |\Theta_P(j)| \sim C' (\lambda_1)^n$$

where C' is a constant. Hence, the growth is exponential with growth rate $\lambda_{\max}(P)$. ■

Example 20 Consider, for instance:

- In the check scheme depicted in figure 2.1, there is one cycle of length 4;

$$\text{Growth}(n) = \lfloor \frac{n}{4} \rfloor.$$

- In the check scheme depicted in figure 2.2, because of the two cycles connected together,

$$\text{Growth}(n) \sim Cn^2$$

for some constant C .

- In example in figure 2.7, $\text{Growth}_\Theta(n) \sim C \cdot 1.1148^n$.

In summary,

A lack of alternatable segments to polynomial growth, while the existence of alternatable modules generates exponential growth. The growth rates are dependent on the number and relationships between the modules.

In the polynomial (non-alternatable) case, the interpretation is easy: the polynomial growth order is equal to the maximum number of modules that can be connected. In the exponential case, the internal structure of the module inter-relationships becomes more important than the number of independent modules. If the component is regular with degree d , then $\lambda_{max} = d - 1$. On the other hand, if it is a union of C cycles of length L joined at one point, then $\lambda_{max} = C^{1/L}$. In the future, I would like to bring to bear a more general descriptive theory of the top eigenvalue to give better structural interpretation to the growth rate.

2.3.4 Summary

Theorem 1 [Pattern properties as graph properties] *The results of the previous section may be summarized as the correspondence:*

<i>Pattern Object</i>	<i>Graph Object</i>
<i>radius r local check scheme Θ</i>	<i>subgraph $G(\Theta) \subset \mathbb{D}(r, m)$</i>
<i>elements of $\Theta^{-1}(1)$</i>	<i>nodes in $G(\Theta)$</i>
<i>left/right compatibility indices in Θ</i>	<i>in/out degrees in $G(\Theta)$</i>
<i>configuration in $\Theta(\mathbb{C})$</i>	<i>Path in $G(\Theta)$</i>
<i>x is Θ-connectable to y</i>	<i>$G(x)$ is connected to $G(y)$ in $G(\Theta)$</i>
<i>x is Θ-alternatable with y</i>	<i>$G(x)$ and $G(y)$ in same SCC</i>
<i>Θ-segment</i>	<i>Irreducible cycle of $G(\Theta)$</i>
<i>x is Θ-expandable</i>	<i>$G(x)$ is connected to a cycle in $G(\Theta)$</i>
<i>x is Θ-repeatable</i>	<i>$G(x)$ subset of a cycle of $G(\Theta)$</i>
<i>unrepeatable configurations</i>	<i>maximal acyclic path (MAPs)</i>
<i>Admissible Sizes</i>	<i>cycle lengths along MAPs</i>
<i>Growth Rate</i>	<i>cycle numbers in SCCs along MAPs</i>

2.4 Local Checkability and Formal Languages

The results of this chapter suggest strongly that there is a relationship between local checkability and the theory of formal languages. Patterns, as defined in §1.3, are precisely languages. To explore the relationship more closely, it is useful to investigate the closure properties of local checkability. We investigate two parallel concepts: (a) locally checkable patterns, defined as in def. 18, and (b) *locally generated patterns*, those T for there is a local check scheme Θ such that $T = \Theta(\mathbb{C})$. Every locally generated pattern is locally checkable, but not vice versa.

Suppose we're given two local check schemes Θ_1, Θ_2 , which generate the pattern $T_1 = \Theta_1(\mathbb{C})$ and $T_2 = \Theta_2(\mathbb{C})$. Then

$$\Theta = \Theta_1 \cdot \Theta_2$$

is a check scheme for $T_1 \wedge T_2$, the logical “AND,” while

$$\Theta' = (\Theta_1 \cdot \Theta_2 + \Theta_1 + \Theta_2) \pmod{2}$$

checks $T_1 \vee T_2$, the logical ‘OR’ of the patterns. Hence the locally generated patterns are closed under logic operations, and

$$LCR(T_1 \wedge, \vee T_2) \leq \max\{LCR(T_1), LCR(T_2)\}.$$

Similarly, local generated languages are closed under concatenations: define

$$T_1 \cdot T_2 = \{X \circ Y \mid X \in T_1, Y \in T_2\}.$$

Then if Θ_1 and Θ_2 are check schemes for T_1 and T_2 , defining $\widetilde{\Theta}(b) = 1$ for all $r(\Theta_1) + r(\Theta_2)$ -neighborhoods in $T_{\Theta_1} \cdot T_{\Theta_2}$ gives local check scheme for $T_1 \cdot T_2$. Hence,

$$LCR(T_1 \cdot T_2) \leq LCR(T_1) + LCR(T_2).$$

A similar construction holds for unordered concatenation, i.e. for the pattern $T_1 \times T_2$ defined as containing configurations

$$x_1 \circ y_1 \circ \dots \circ x_n \circ y_n$$

with $x_i \in T_1 \cup \{\emptyset\}$ and $y_i \in T_2 \cup \{\emptyset\}$. Thus the locally generated languages are closed under the Kleene star operation. By using the operators $\{\wedge, \vee, \cdot, \times\}$ in arbitrarily complicated combinations, a wide variety of locally checkable complex patterns can be created. For instance, given the two simple repeat patterns T_{100} and T_{1000} , we can easily form the combination patterns

$$T_{100} \vee T_{1000} = \{(100)^n, (1000)^n, \mid n \in \mathbb{N}\}$$

and

$$T_{100} \cdot T_{1000} = \{(100)^n(1000)^m, \mid n, m \in \mathbb{N}\}.$$

The locally generated patterns are not, evidently, closed under complement, as a simple example shows. However, it is easy to show that the complement of a locally generated pattern is always locally checkable, with a construction showing that $LCR(\neg\Theta(\mathbb{C})) \leq 2r(\Theta) + 1$.¹ In light of these constructions, it is easy to see (by comparison to the basic closure operations of regular languages) that:

- The locally generated patterns are a proper subclass of regular languages, while the locally checkable languages are a proper superclass of regular languages. More specifically: every regular language L contains a locally generated sublanguage L' such that $L \cap \mathbb{C}_n \neq \emptyset$ implies $L' \cap \mathbb{C}_n \neq \emptyset$. Thus, the locally generated and locally checkable concepts have slightly less nice closure properties than the regular language concept, but they “sandwich” it.
- A local check scheme Θ is a form of finite generator of the language $\Theta(\mathbb{C})$.
- The graph $G(\Theta)$ associated with Θ is the graph of a non-deterministic finite automata (NDFFA) recognizing $\Theta(\mathbb{C})$.
- The proofs that a pattern are not locally checkable (e.g. prop. 7) are similar to applications of the pumping lemma ([18]).

In light of these correspondences, several of the results of this chapter become connected to known results from the literature on regular languages (see e.g. [8] and [26]).

There are several major differences. The most important is that we work with a *construction problem*, which concerns “producing at least one instance in each size class”, rather than a *recognition problem*, which concerns “accepting all instances of the pattern and rejecting all non-instances.” This basic difference in motivation informs the rest of this thesis, and is evident from the kinds of results that I present.

¹Specifically, define local check scheme Θ' with radius $2r(\Theta) + 1$ by setting $\Theta'(B) = 1$ if B contains a radius- r sub-ball b such that $\Theta(b) = 0$.

Second, some of the results summarized in Theorem 1 invoke distinctions which are a bit “more detailed” than the standard work on regular languages and the structure of NFA graphs. These “more detailed” properties are neither preserved by the standard wreath product decomposition of monoids, nor are they captured as invariants of the rational or algebraic expressions typically used to characterize formal language classes (as discussed in, for example, [8]). This includes, for example, the relationships between cycle sizes within strongly connected components, and the presence of given numbers and sizes of irreducible cycles. Such detailed calculations – and other, even finer invariants introduced in the next chapters (e.g. the power spectrum discussed in chapter 5, the DeBruijn geometry discussed in chapter 7) – turn out to be important for yielding qualitative results that would be opaque to the standard perspective.

Chapter 3

Local Checkability is Sufficient: Generic Constructions

Proposition 5 says that in our 1-D model with finite state, local checkability is a necessary condition for robust static solvability, i.e. only for patterns T that have local check schemes Θ can there be finite-radius rules f that can solve T on all initial conditions. In this chapter, I will prove the converse to proposition 5, establishing that:

Local Checkability is both a necessary and a sufficient condition for solvability. Moreover, the existence of a local check scheme provides enough structure for us to construct a generic algorithm producing local rules that solve any given locally checkable pattern.

In §3.1, I introduce a simple form check scheme that is especially easy to solve. These *single-choice* check schemes can be solved by “structure creation waves” generated by a discrete gradient operator. In §3.2, I describe the solution to general check schemes, introducing a *self-organize Turing machine* that implements a “Naive Backtracking” algorithm via a *distributed signalling system*. In §3.3, I compute both the average and worst-case runtime of the Naive Backtracking algorithm as a function of the structure of the graph associated with the local check scheme.

3.1 Single-Choice Patterns: A Gradient Algorithm

We will first show how to solve a subclass of local check schemes that have an especially simple structure. Define a local check scheme Θ to be *single-choice* if for any Θ -accepted ball b , the right- and left- compatibility index (as defined in §2.2.1) is at most 1. All the repeat patterns are single-choice.

For a single-choice check scheme Θ , given a Θ -accepted ball b , let that unique i be such that $b \circ i$ is Θ -accepted be denoted $\nabla_{\Theta}(x)^+$, when it exists. Intuitively, this is the “gradient” of Θ from the left direction. Note that $\nabla_{\Theta}(x)^+$ is only a function of the right-most $2r$ states in b .

We will use the gradient to produce robust solutions to single-choice check schemes. Let $R = 2r$ and let B be any R -neighborhood. Let B_- denote the r -neighborhood consisting of the portion of B to the left of the center agent, and let $B(0)$ denote the state of the center agent. Now, let f_{Θ} be the local rule of radius R defined by

$$f_{\Theta}(B) = \begin{cases} \nabla_{\Theta}^+(B_-), & \text{if } \Theta[B_-] \\ B(0), & \text{otherwise} \end{cases} \quad (3.1)$$

In words, the first clause causes an agent that is just to the right of a patch of Θ -correctness to switch its state to extend the patch of Θ -correctness rightward. The second clause instructs an agent that is not the rightmost agent of patch on the “border of Θ -correctness” to do nothing; and left-end agents are automatically on the

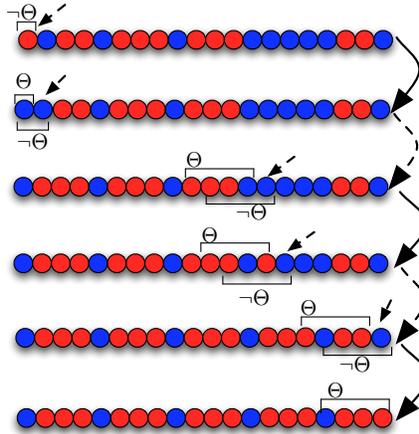


Figure 3.1: Single-choice algorithm on typical initial condition. Θ = correct locally; $\neg\Theta$ = not locally correct.

border of correctness if they have incorrect states. Intuitively, what this rule does is generate a right-moving “wave of Θ -correctness” by applying the local Θ -gradient operator (fig. 3.1).

Proposition 13 For any single-choice local check scheme Θ , f_Θ as defined in eq. 3.1 is a solution to T_Θ .

Proof: If Θ is single-choice, then for any n such that $T_{\Theta,n} \triangleq T_\Theta \cap \mathcal{C}_n \neq \emptyset$, there is a unique configuration $X_n \in T_n$. We have now to show that if X_0 is any configuration of size n and c is any live call sequence, then $(f_\Theta)_c^n(X_0)$ converges to X_n . We prove inductively that for each $j \leq n$ and some t , $X_t \triangleq (f_\Theta)_c^t(X_0) = X_n[1 : j] \circ Z$ for some configuration Z . To this end, suppose inductively that $X_0 = X_n[1 : j] \circ Y$ for some $j < n$, and that $Y(1) \neq \nabla_\Theta^+(X_n[j - 2r - 1 : j])$. In words, $j + 1$ is the first position in X_0 where the local check scheme Θ fails to hold. Then because of the second clause of the definition of f_Θ in eq. 3.1, all $k \leq j$, $F_\Theta(B_R(k, X_0)) = X(k)$. Let t be the first timestep in c such that agent $j + 1$ is called (such a t must exist since c is assumed to be live). Then $X_{t-1} = X_n[1 : j] \circ Y(1) \circ Z$ where Z is some configuration. Let $B = B_R(j + 1, X_{t-1})$. Since $\Theta[B_-]$ holds by inductive assumption, the first clause of eq. 3.1 yields $f_\Theta(B) = \nabla_\Theta^+(B_-)$. Thus $X_t = X_n[1 : j + 1] \circ Z'$, completing the induction. ■

3.2 The Naive Backtracking Algorithm

As discussed in §2.2.1, local check schemes can allow multiple choices to follow a consistent subconfiguration. For example, the radius-2 check scheme for the pattern $T_{100} \cdot T_{1000}$ is multi-choice because both 0 and 1 are acceptable states following the 2-neighborhood $00\widehat{1}00$:



Suppose we tried to solve the pattern $T_{100} \cdot T_{1000}$ along the lines of what we did for single-choice patterns. We’d have to choose a value for $\nabla_\Theta^+(00\widehat{1}00)$, say 0. In this case, the solution will write a string of repeats of 1000 until it reaches the right end. If the number of agents in the configuration is such that the end does not precisely line up with a complete unit of 1000, the only way the configuration can be solved is if the number of repeats of 100 back toward the left end of the configuration is changed. This means that the right-end agent has to communicate to left-end agents, sending a signal with the message: ‘substitute another copy of 100 in place of 1000’.

The signal will travel toward the left until reaching the left-most instance of 1000, whereupon it should cause the agent whose local radius-6 neighborhood is $1001000\widehat{1}000100$ to substitute the choice of ‘0’ state made originally with ‘1’ instead. Having done this, the signal must dissipate and cause a new gradient wave to

travel to the right, realigning all the repeats of 1000 relative to the new choice. When this wave reaches the right end, if a complete unit of 1000 fits, the configuration is solved. If not, another signal will have to be sent right-ward with the same message. Repeating these events, the configuration will eventually be solved (see fig. 3.2).

A similar idea works in general. We make the following technical definitions:

- Identify the state set S with the integers $\{1, \dots, m\}$.
- Given a Θ -accepted r -ball b , let $\nabla_{\Theta}(b)^+$ denote the minimal $i \in S$ such that $b \circ i$ is Θ -consistent, when such an i exists. This is a generalization of the Θ -gradient operator used in the previous section on single-choice patterns.
- Given a Θ -accepted r -ball b , let $\Theta^+(b)$ denotes the minimal state $j \in S$ such that $j > b(|b|)$ and that $\Theta(b(1 : |b| - 1) \circ j) = 1$, when such a j exists (where $|b|$ is the number of nodes in b). Intuitively, $\Theta^+(b)$ is the *next Θ -consistent choice, after $b(|b|)$* .
- Given a r -ball b , let $M(b)$ be the maximal l such that $\Theta[b(1 : |b| - 1) \circ l]$. Intuitively, $M(b)$ is the *last possible choice* for $b(|b|)$.
- A subconfiguration y is *right-end Θ -consistent* if there is a configuration X with $\Theta(X) = 1$ such that $X = z \circ y$ for some z . Given a Θ -consistent subconfiguration x , let $\eta_{\Theta}(x)$ denote the minimal $i \in S$ such that $x \circ i$ is right-end Θ -consistent, when such an i exists. Intuitively, $\eta[B]$ makes an appropriate Θ -consistent choice of state for an agent at the right-end of the configuration, taking into account edge-effects.

We now construct a local rule F_{Θ} , with radius $2r(\Theta) + 2$. For ease, we also assume that we have $m + 2$ extra states to work with, denoted \triangleright , \triangleleft , and $\Delta_1, \dots, \Delta_m$. Ten detailed rules defining F_{Θ} appear in the left-hand column of figure 3.2. Summarizing them, F_{Θ} is defined by:

Algorithm 1: The Naive Backtracking Algorithm

```

if Rules 1, 3, 9, 10 apply, then
  |  $F_{\Theta}(B) = \triangleright$ 
else if Rules 5, 6 apply then
  |  $F_{\Theta}(B) = \triangleleft$ 
else if Rule 2 applies then
  |  $F_{\Theta}(B) = \nabla_{\Theta}(B)^+$ 
else if Rule 4 applies then
  |  $F_{\Theta}(B) = \eta(B)$ 
else if Rule 8 applies then
  |  $F(B) = \Theta(B(-2r : 0))^+$ 
else if Rule 7 applies then
  |  $F(B) = \Delta_{B(-1)}$ 
else
  |  $F(B) = B(0)$ .
end

```

As shown in the right-hand column of figure 3.2, the 10 rules defining F_{Θ} work by “implementing a Naive Backtracking search run by a self-organizing virtual distributed Turing machine”.

To see why this is, first note that **Rules 1-4** essentially re-implement the single-choice algorithm in §3.1. A right-moving Turing head – represented by an extra state denoted \triangleright – is “born” (as per **Rule 1**) when an agent determines that it is on the “border of correctness”. Then, the head propagates across the system (as per **Rules 2-3**), writing a trail of local Θ -correctness in its wake according to the operator ∇_{Θ}^+ . When it hits the right boundary, the head halts (i.e. disappears) as per **Rule 4**, by applying operator $\eta(B)$, when possible.

Sometimes, the right-moving \triangleright -head is unable to halt and disappear, because it will encounter a situation in which no possible local completion exists – i.e. either **Rule 2** applies and $\nabla_{\Theta}^+(B)$ doesn’t exist, or **Rule 4** applies and $\eta(B)$ does not exist. In this case, **Rules 5-10** implement the signalling process described in the

discussion above, using $m + 1$ extra states denoted \triangleleft , and $\Delta_1, \Delta_2, \dots, \Delta_m$. Specifically, **Rule 5** reverses the head \triangleright , overwriting it with the state \triangleleft , which represents the turing head moving in the opposite direction. The left-moving turing head is propagated by **Rule 6** until the first point where

$$B(0) \neq M(B_-).$$

Intuitively, this is first place where a *next choice* consistent with the multi-choice local check scheme can be made – any point where the right-compatibility index is greater than 1 and the existing choice is not the “last choice”. When such a spot is reached, **Rules 7-8** have the agent just to the left of the \triangleleft head “click up” to the next choice, using the operator $\Theta(B)^+$. Finally **Rule 9** reverses the turing head back to the right direction, in the \triangleright state.

Thereupon, **Rule 2-3** apply again, and the process repeats. Each such repetition represents a one *backtracking* event. This backtracking protocol is “naive” in that every possible choice tried out, until one is found that works. For this reason, I call F_Θ the Naive Backtracking algorithm. (We’ll see less naive backtrack protocols chapter 4.) In other terms, F_Θ a brute-force search through the set of possible locally correct structures. F_Θ comes to equilibrium only when the local ball around every agent is consistent with the check scheme Θ – if no Θ -consistent state of size n exists, F_Θ will never achieve equilibrium.

Proposition 14 *The Naive Backtracking Algorithm F_Θ is a solution to Θ .*

Combining prop. 14 (which is proved in an appendix to this chapter) with propositions 5, we have:

Theorem 2 [Local Checkability \Leftrightarrow Solvability] *Local checkability is a necessary and sufficient condition for local solvability. Moreover, a locally checkable pattern $\Theta(C)$ has a robust solution F_Θ with radius $2 \cdot r(\Theta) + 2$, using $m + 2$ extra states. A single-choice pattern has a solution f_Θ with radius $2r(\Theta)$, using no extra states.*

In chapter 6, I show how to remove the requirement for extra states in the general case, at the cost of additional radius.

3.2.1 Distributed, Virtual, Robust Heads

The Turing machine implemented by F_Θ is virtual, in that the turing heads do not exist apart from states of the agents in the system. At any given time, it is an emergent structure hosted by several neighboring agents. For the head to “leave a trail of local Θ -correctness its is wake”, the agent hosting it – say, agent i – must act. Agent i determines the new state take on so that with respect to the states of the $2r + 1$ agents to its left, the pattern Θ will be locally satisfied. By choosing a new state appropriately, and releasing the turing head state, agent i can increase the extent of Θ -correctness by one agent, as long agents $[i - 2r - 1, \dots, i - 1]$ are already Θ -correct. When there are several possible choices of state that i could take on and still satisfy Θ , the agent chooses the minimum state.

Before an agent hosting the Turing \triangleright – for the sake of argument, say, agent i – will “release” it (as per **Rule 2**), it waits for agent $i + 1$ to take on the \triangleright state (as per **Rule 3**). This way, once agent i updates, the head is not lost, but instead has moved one step further to the right. The region of Θ -correctness smoothly increases as \triangleright moves along. Two-part propagation ensures that the head will remain coherent, regardless of the update synchrony (or lack thereof) in the system. An algorithm that was not expected to work on all live update call sequences, but say was restricted to the synchronous case only, would not need these protections.

For similar reasons, **Rules 7-9** require an extra states $\Delta_1, \dots, \Delta_m$ to ensure the robustness of the “next-choice update and head reversal” process. The whole point of backtracking to the first agent where a “next Θ -consistent choice” can be made is that the agent just to the left of the head, say agent i , will “click up” to that next choice. Agent i must do this before the head reverses. But if agent i ’s rule for state change depended *only* on the presence of the head to its right, it might go *past* the next choice and instead to the “next-next” choice if it was called multiple times before the head reversed. To protect against this, if agent i is in state s it must wait for the state Δ_s to appear to its right; once it sees Δ_s , it then can click up to the next Θ -consistent state It will not click up again since now its state is no longer equal to s . Similarly, the agent at the turing head must wait for agent i to change before enacting the reversal of its state from Δ_s to \triangleleft . In

other terms, the turing head stores some scratch memory to ensure that its update-and-reversal procedure is robust to asynchrony. Each different state $s \in S$ will require a different scratch state Δ_s .

Note, moreover, that this is a *distributed* Turing machine because at any given time, there may be multiple heads present in the system; the rules encode their interactions so that eventually a consistent choice is made.

3.2.2 A Turing Operator Basis

The self-organized Turing machine can be specified independently the Naive Backtracking algorithm. Specifically, a basis set of “robust Turing operators” can be extracted from the rules defining F_Θ that will allow the programming of many other algorithms. These operators are:

- A head birth operator $\mathbb{B}_\mathcal{G}$ in analogy to **Rule 1**, which generates a turing head whenever some local birth condition \mathcal{G} holds,
- a right-propagation operator $\mathbb{P}_{\mathcal{G},\Delta}^R$ in analogy to **Rules 2-3**, which moves a head along to the right in two parts as per the discussion above when local condition \mathcal{G} holds, and leaves a trail of states determined by local computation Δ
- the mirroring left-propagation operator $\mathbb{P}_{\mathcal{G},\Delta'}^L$,
- a left-right turn operator $\mathcal{T}_{\mathcal{G},\Delta'}^{LR}$ in analogy to **Rules 7-9**, which reverses a left-moving head into a right-moving head when condition \mathcal{G} holds, after having induced a local state change determined by computation Δ ; and which uses an extra Δ state to delay the application of the reversal until Δ completes,
- the mirroring right-left turn operator $\mathcal{T}_{\mathcal{G},\Delta}^{RL}$.
- and a halting operator $\mathbb{H}_{\mathcal{G},\Delta}$ in analogy to **Rule 4**, which removes a turing head whenever a local halting condition \mathcal{G} holds, and replaces it with the result of local computation Δ

Combinations of these operators can be used to compactly describe many local algorithms, including the single-choice and Naive Backtracking algorithms.

For instance, the single-choice rule in eq. 3.1 is simulated by the expression

$$\mathbb{B}_\mathcal{G} + \mathbb{P}_{\mathcal{G},\Delta}^R$$

where

$$\mathcal{G}(B) = \Theta[B_-] \wedge (-\Theta[B(-2r : 0)])$$

is the “border of Θ -correctness” condition and $\Delta(B) = \nabla_\Theta^+(B_-)$ is the local Θ -gradient. (The $A + B$ notation indicates that operator B applies if A fails to apply.)

The Naive Backtracking algorithm is equivalent to

$$F_\Theta(B) = \mathbb{B}_{\mathcal{G}_1} + \mathbb{P}_{\mathcal{G}_2,\Delta}^R + \mathcal{T}_{\mathcal{G}_5,\emptyset}^{RL} + \mathbb{P}_{\mathcal{G}_6,\emptyset}^L + \mathcal{T}_{\mathcal{G}_7,\Theta^+(B)}^{LR} + \mathbb{H}_{\mathcal{G}_4,\eta} \quad (3.2)$$

where \emptyset is the null condition and $\mathcal{G}_i(B)$ expresses the conditions defining when **Rule i** applies. This equation contains the single-choice algorithm as a subset of the terms, with backtracking implemented with the other terms.

3.2.3 Other Orderings

The backtracking search self-organized by F_Θ is “lexicographic” because it tries out each new option by picking states in the order of their numerical value. The order in which new states are picked can be chosen arbitrarily, not simply by numerical order of the states.

Recall §2.2.1 the definition of the left- (resp. right) compatibility set $L(b)$, of a Θ -accepted ball b – that is, the set of states that could be joined to the left of b and still have Θ be satisfied. Then define:

Definition 22 [*Left-Choice Ordering Function*] A Left-choice ordering function for Θ is a mapping

$$O : \mathcal{B}_r \times [m] \rightarrow [m]$$

such that $O(b, \cdot)$ is a bijection from $\{0, \dots, |R(b)| - 1\}$ to $R(b)$.

In words, O specifies, for each valid local ball b , an ordering on the various options for Θ -consistently completing b . $O(b, 1)$ is the first choice, $O(b, 2)$ the second choice, etc... and $O(b, |R(b)|)$, the last choice. Because it is a bijection, $O_b^{-1} : R(b) \rightarrow \{0, \dots, |R(b)| - 1\}$ is well-defined, by setting $O_b^{-1}(j) = k$ if $O(b, k) = j$. Define the ordering $\geq_{O,b}$ by $s_1 \geq_{O,b} s_2$ if $O_b^{-1}(s_1) \geq O_b^{-1}(s_2)$. $O(b, j)$ is a state s ; the node in $G(\Theta)$ chosen thereby is $b[2 : 2r + 1] \circ s$. Denote the latter by $\mathbf{O}(b, j)$.

Given a radius- r ball B , a state s is a “non-maximal” choice of O for B if $B(0) = \mathbf{O}(B, j)$ for $j < |R(b)| - 1$. The “next choice” after s is given by $O(B, O_B^{-1}(s) + 1)$. If B is a radius $2r + 2$ ball, and $B(0)$ is a non-maximal choice of O for $B[-2r - 1 : -1]$, then the next choice is given by the somewhat unwieldy expression $O(B[-2r - 1 : -1], O_{B[-2r-1:-1]}^{-1}(B(0)) + 1)$ – for notational ease, denote this B_O^{+1} .

For each choice function O , we can define a corresponding Generalized Naive Backtracking algorithm F_Θ^O by replacing all references to the numerical order with appropriate references to O . Formally, let F_Θ^O be defined by modifying F_Θ with the replacements:

- In **Rule 2**: the state $\nabla_\Theta^+(B)$ replaced with $O(B^{-r}, 1)$,
- In **Rule 6**: the check for $B(0) = M(B)$ replaced with the boolean $L_O[B]$, defined by

$$L_O[B] \triangleq (B(0) = O(B^{-r}, |R(B^{-r})|)),$$

- In **Rule 7**: the check for $B(0) \neq M(B^{-1})$ replaced with $L_O^-[B^{-1}]$
- In **Rule 8**: the check for $B(0) \neq M(B)$ replaced with $L_O^-[B]$, and the state $\Theta(B(-2r : -0))^+$ replaced with B_O^{+1} .
- In **Rule 9**: the check for $B(-1) = M(B-1)$ replaced with $L_O[B^{-1}]$.

The same proof that applied to F_Θ applies to F_Θ^O for any O . Of course, the original rule F_Θ can be thought of as F_Θ^O for O defined by $O(B, i)$ to be the i -th smallest state in $R(B)$.

3.2.4 Viewing F_Θ^O on $G(\Theta)$.

The operation of the Generalized Naive Backtracking Algorithm F_Θ^O can be “viewed” on the graph $G(\Theta)$. Suppose Θ is a local check scheme whose graph $G(\Theta)$ looks, schematically, like shown in figure 3.3.1. Now,

1. Suppose we’re given a configuration X . In this configuration, suppose that at some position, say agent j , there is a “mistake” relative to Θ , and j is the “left-most” such mistake. At this position j , Θ is not satisfied, but for all positions $j' < j$, Θ is satisfied. For Θ to be satisfied for all positions to the left of j , the configuration $X[1 : j - 1]$ describes a path P in $G(\Theta)$, as we saw in §2.3.1. If j is very large compared to $|G(\Theta)|$, there will have had to be at least one cycle in $G(\Theta)$ that P wraps around several times. This can be shown in the graph $G(\Theta)$: see figure 3.3.2.
2. Now, because of **Rule 1**, the algorithm F_Θ^O will create a self-organized turing head around position j . We can notate this on the graph $G(\Theta)$ with an arrow at the position where the head is located.
3. **Rule 2** and **Rule 3** will cause the turing head to move to the right. As it moves, it writes state consistent with the local check scheme Θ . That is, it moves *along the graph* $G(\Theta)$ (fig. 3.3.3).
4. If the turing head is at a node b in $G(\Theta)$ where the out-degree is 1 (as in the previous figures), the head has only one choice to stay in $G(\Theta)$ as it moves forward. However, if the node b has out-degree larger than 1, a choice must be made. The ordering function O now comes in to play. O assigns each possible next state an order (fig. 3.3.4). The rule F_Θ^O picks the first choice assigned by O to b , i.e. the state $O(b, 1)$. As the head moves forward, each “choice point” where the out-degree is larger than 1, the ordering function determines the way (fig. 3.3.5).

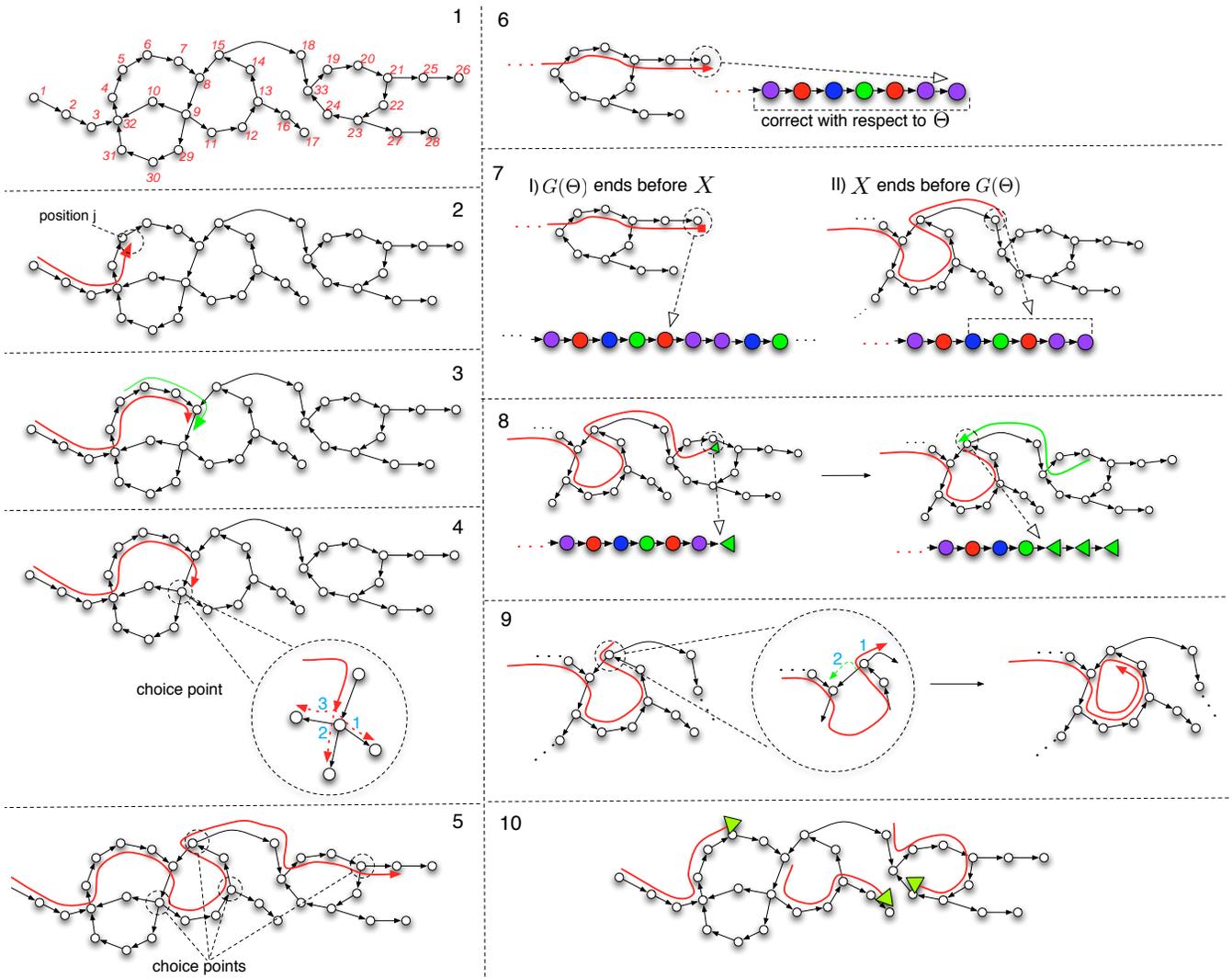


Figure 3.3: A visual description of the rule F_{Θ}^0 on the graph of $G(\Theta)$.

5. The best case scenario is that the turing head moves through onto a terminal branch of $G(\Theta)$, and ends at precisely the right point in relation to the size of the configuration X (fig. 3.3.6). This scenario can fail to come to pass if either (I) the terminal branch end in $G(\Theta)$ comes before the end of the configuration X , or (II) the configuration X comes to end first before a hitting terminal branch (or at least, the end of the terminal branch) in $G(\Theta)$ (fig. 3.3.7). In either case (I) or (II), **Rule 5** is activated; a left-moving turing head is created, and the head propagates leftward until it encounters a choice-point (fig. 3.3.8).
6. When the head arrives at a choice-point, the following evaluation takes place: was the choice made the last time the head went through this choice point the “last choice”, according to the ordering function O ? If it isn’t, that means O has at least one “next choice” possible. In this case, **Rule 8** flips the state of the agent at the choice point up to the next choice, and **Rule 7** and **Rule 9** reverses the head reverses; and the local rule returns to evaluating as in steps 1-5 (fig. 3.3.9). On other hand, if it already *had* made the “last choice”, then **Rule 6** keeps the head going until it encounters the *next* choice point to the left, and makes the same evaluation.
7. The above steps repeat until Θ -consistent configurations are searched through until a solution that is locally correct everywhere is attained (if there is one).

Throughout, I’ve spoken as if there’s one head, but actually there may be several at any given time, operating in parallel (fig. 3.3.10). The local rules specify interaction of heads so that eventually one wins out.

3.2.5 The Effect of the Ordering Function O

Different ordering functions O lead to different kinds of behavior. To see why, recall from §2.3.1 the discussion about strongly connected components in $G(\Theta)$. A SCC is a maximal set of nodes that are mutually connected; SCCs in $G(\Theta)$ correspond to maximal sets of alternatable subpatterns. In the graph schematic in fig. 3.3, nodes 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 29, 30, 31, and 32, form one non-trivial strongly connected component containing three irreducible cycles, while nodes 19, 20, 21, 22, 23, 24 and 33, form another SCC containing one cycle.

Now suppose at some point during the operation of the algorithm F_{Θ}^O , a turing head is located at a position on a nontrivial SCC. At some choice-points along the cycle, some of the options may lead out of the SCC. In figure 3.4.A), option 1 leads out of the strongly connected component, while option 2 remains in it.

The trajectories under F_{Θ}^O look superficially quite different if O generally chooses to “leave” SCCs, versus “staying within” them. In the former case, suppose we’re on a configuration that is large compared to the number of nodes in $G(\Theta)$. A head starting toward the left-end of the configuration will move forward through the SCCs to a terminal branch, hitting the edge prematurely – that is, case (I) from Step 5 above. The head will now bounce back and forth repeatedly as it edges along, poking into each terminal branch once, before adding a new cycle (see figure 3.4 column I). In the latter case, the head first makes a long run, writing many copies of the first cycle; and hits the end of the configuration before getting to a terminal branch (this is case (II) from Step 5). Then the head bounces back and forth, moving a short distance away from the right end each time, giving up one copy of the first cycle each time (see figure 3.4 column II).

Though under the various ordering functions O , F_{Θ}^O generates these different behaviors, it will always be the case that a Turing head will only return to a spot it is previously covered after trying out all possible completions from that spot consistent with Θ (that is the content of lemma 2). Moreover, over all initial conditions, all orderings have approximately the same asymptotic run-time scaling.

3.3 Run-Time Analysis of Naive Backtracking Algorithm

In §1.4, I defined the average runtime scaling function, $TTS^{avg}(F)(n)$, and the worst-case runtime scaling function $TTS^{worst}(F)(n)$, for any local rule F . The problem we wish to solve now is: as a function of local check scheme Θ and ordering O , compute these two run-time complexity scaling functions for the Naive Backtracking algorithm, F_{Θ}^O . In these discussions, we assume for simplicity that we’re working in the completely synchronous timing model, but the results remain the same in any live uniform timing model.

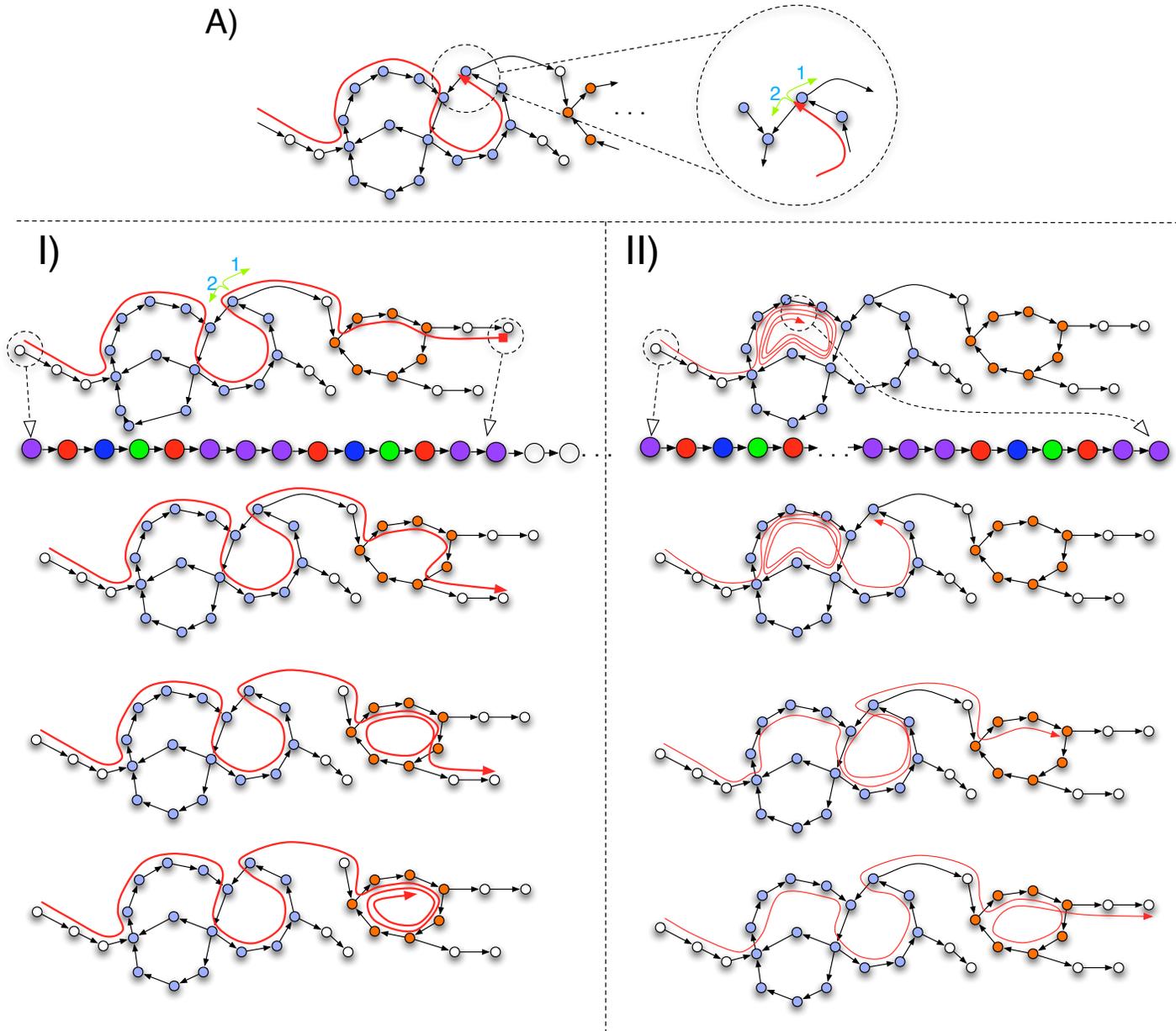


Figure 3.4: A) Option 1 leads out of the SCC comprised by the light-blue nodes, while option 2 stays within it. Column I) A typical trajectory evolution for F_{Θ}^O , when the first choices of the ordering function O are to “leave” SCCs in $G(\Theta)$. II) A typical trajectory F_{Θ}^O for an ordering choice function O whose first choices are to “stay within” SCCs in $G(\Theta)$.

Suppose X is a size- n configuration of the form

$$X = Y \circ \triangleright \circ Z,$$

where Y is a Θ -consistent subconfiguration and Z is an arbitrary subconfiguration. In words, X is a configuration in which the left-most turing head is at position $|Y| + 1$, and everything to the left of the head is correct with respect to Θ . Let

$$\text{ext}^\Theta(Y) = \{W \in \Theta(\mathbb{C}) \mid W[1 : |Y|] = Y\}.$$

That is, $\text{ext}^\Theta(Y)$ are the *extensions* of Y that are consistent with Θ . Let $\text{ext}_j^\Theta(Y) = \text{ext}^\Theta(Y) \cap \mathbb{C}_{\leq j}$, that is, the extensions of Y of size n or less. The key observation that helps us solve the run-time computation is this:

Proposition 15 [Simple Worst-Case Upperbound] *Let X be a configuration of the form $Y \circ \triangleright \circ Z$. Then*

- If $\text{ext}^\Theta(Y)$ contains a configuration of size $|X|$, $TTS(F_\Theta^O, X) \leq 4 \cdot |\text{ext}_n^\Theta(Y)|$.
- If $\text{ext}^\Theta(Y)$ does not contain a configuration of size $|X|$, for some time T such that $|\text{ext}_n^\Theta(Y)| \leq T \leq 4 \cdot |\text{ext}_n^\Theta(Y)|$, $(F_\Theta^O)^T(X) = Y \circ S \circ Z'$ where $S = \Delta_{Y(|Y|)}$ or \triangleleft .

This proposition is proved in an appendix to this chapter, but the basic reason it holds is that, on the initial condition X , the algorithm F_Θ^O searches through all the extensions to Y before moving on to any other configurations. If one of these configurations can work, then the algorithm finishes before all of the extensions to Y have been tried (option 1 in the proposition above). If none of the extensions can work, then the algorithm has to try, and reject, all of them before moving on (option 2). The relevant point is that, over all, the algorithm takes at least one, and no more than 4, timesteps per extensions (though some take more than others).

As a corollary of prop. 15, we obtain an upper bound on the worst-case. Recall the definition of the growth function $Growth_\Theta(n)$, from section 2.3.3, as the number of configurations in Θ of size n or less.

Corollary 1 *For all Θ and O ,*

$$TTS^{\text{worst}}(F_\Theta^O)(n) \leq 4 \cdot |Growth_\Theta(n)| + n + 3.$$

Proof: Suppose X is a configuration of size n . If X is not Θ -admissible, then within 3 timesteps, turing head begins to emerge – that is, there is at least one position whose state is Δ_i , \triangleleft , or \triangleright . Hence, $TTS(F_\Theta^O, X) \leq 3 + TTS(F_\Theta^O, X_1)$, where $X_1 = Y_1 \circ s \circ Z$, where Y_1 is Θ -consistent and $s \in \{\Delta_i, \triangleright, \triangleleft\}$. Let j be the position of the of left-most agent in X_1 such that $X(j) \in \{\Delta_i, \triangleleft, \triangleright\}$. If $X(j) = \Delta_i$, then the turing head is in the process of turning. Within 2 timesteps, the configuration becomes $X[1 : j - 2] \circ s_1 \circ \triangleright \circ Z$, for some subconfiguration Z , where $s_1 \in \{1, \dots, m\}$. If $X(j) = \triangleleft$, then the turing head is moving left; within j timesteps, the turing head has found a choice-point, and will, within another 2 timesteps, turn. Hence, $TTS(F_\Theta^O, X) \leq TTS(F_\Theta^O, X') + n + 3$, where $X' = Y \circ \triangleright \circ Z$, with Y being Θ -consistent.

For notational convenience, for $j \leq |Y|$, let $Y_j = Y[1 : j]$ and $b_j = Y[j - 2r - 1 : j]$. Also, abbreviate $s_1 \geq_{O, b_j} s_2$ by $s_1 \geq_j s_2$. Now, let k be the maximal $l \leq |Y|$ such that there is a state s such that $\text{ext}^\Theta(Y_l \circ s)$ has a configuration of size n , and such that $s \geq_l Y(l + 1)$, and let s^* be the O -minimal state that satisfies this. Then by definition, for all $j \in [k + 1, |Y| - 1]$, $\text{ext}^\Theta(Y_j \circ s)$ does not contain a size n configuration for any s such that $s >_j Y(j + 1)$. Similarly, $\text{ext}^\Theta(Y_k \circ s)$ does not contain a size n configuration for any s such that $Y(k + 1) <_k s <_k s^*$.

Now applying proposition 15, part 2, to $Y_{|Y|-1}$, there is a time T_1 such that $X'_{T_1} \triangleq (F_\Theta^O)^{T_1}(X') = Y_{|Y|-1} \circ \Delta_{Y(|Y|-1)} \circ Z'$, with

$$T_1 \leq \sum_{s \geq_{|Y|-1} Y(|Y|-1)} 4 \cdot |\text{ext}_n^\Theta(Y_{|Y|-1} \circ s)|.$$

Now applying prop. 15 part 2, to X_{T_1} , there is a time T_2 such that $X'_2 \triangleq (F_\Theta^O)^{T_1+T_2}(X') = Y_{|Y|-2} \circ \Delta_{Y(|Y|-2)} \circ Z'$, with

$$T_2 \leq \sum_{s >_{|Y|-2} Y(|Y|-2)} 4 \cdot |\text{ext}_n^\Theta(Y_{|Y|-2} \circ s)|.$$

Repeating this $|Y| - k + 1$ times, we have T such that $X'_T \triangleq (F_\Theta^O)^T(X') = Y_k \circ s^* \circ \triangleright \circ Z'$ where

$$T \leq 4|ext_n^\Theta(Y)| + 4 \sum_{j=k+1}^{|Y|-1} \sum_{s >_j Y(j)} |ext_n^\Theta(Y_j \circ s)| + 4 \sum_{Y(k+1) <_k s \leq_k s^*} |ext_n^\Theta(Y_k \circ s)|.$$

Applying prop. 15 part 1 to X_T , $TTS(F_\Theta^O, X'_T) \leq 4|ext_n^\Theta(Y_k \circ s^*)|$, so that

$$TTS(F_\Theta^O, X') \leq 4|ext_n^\Theta(Y)| + 4 \sum_{s \geq_j Y(j)} |ext_n^\Theta(Y_j \circ s)| + 4 \sum_{Y(k+1) <_k s \leq_k s^*} |ext_n^\Theta(Y_k \circ s)|.$$

Now, the sets

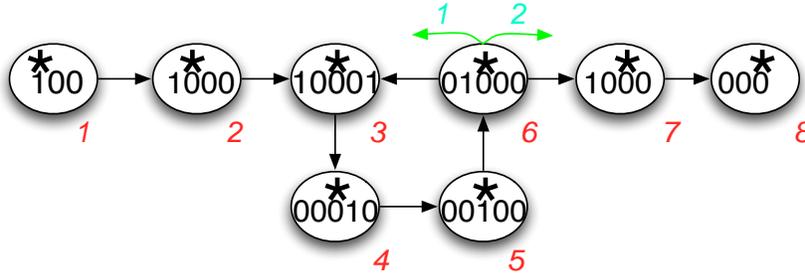
- $ext_n^\Theta(Y_j \circ s)$ for j and $s >_j Y(j + 1)$, and
- $ext_n^\Theta(Y)$, and
- $ext_n^\Theta(Y_k \circ s)$ for all $Y(k + 1) <_k s \leq_k s^*$

are all pairwise disjoint subsets of $ext_n^\Theta(Y_k)$. Hence, $TTS(F_\Theta^O, X') \leq 4|ext_n^\Theta(Y_k)|$. Since $ext_n^\Theta(Y_k)$ is itself a subset of the set of configurations of size n or less, consistent with Θ , the result is complete. ■

The bound from corollary 1 does not completely answer the questions posed at the beginning of this section because it leaves open the question of in which circumstances the bound is tight, and (relatedly) does not say anything about the average runtime. Though answering these questions turns out to be somewhat more involved, the basic result turns about to be a kind of generalization of the result in corollary 1: there is a locally checkable *subpattern* of $\Theta(\mathbb{C})$ whose growth function bounds the average and worst-case run times. The challenge is properly constructing the subgraph of $G(\Theta)$ that generates this subpattern. It turns out that the construction is somewhat subtle, and can look quite different from $G(\Theta)$ as a whole. We'll compute the answer in several illustrative cases, and then state the general result.

3.3.1 Several Examples

Example 21 Suppose that Θ is the radius-2 local check scheme associated with the pattern $\{(1000)^n \mid n \geq 2\}$. (This is the same as T_{1000} within the $n = 4$ size.) The graph $G(\Theta)$ is:



Choose the choice function O whose first choices “stay within” the cycle, as indicated by the light blue arrows in the figure above (formally, this means that $O(010^*00, 1) = 1$ and $O(010^*00, 2) = 0$)

This pattern only has solutions that are multiples of size 4. Consider an initial condition X of size $4n$ of the form $X = \triangleleft \circ Y$ where Y is any configuration of size $4n - 1$. Within one timestep, F_Θ^O brings X to the state $X_1 = \triangleright \circ Y'$ where Y' is again some configuration of size $4n - 1$. The \triangleright head will move right-ward one position every two time steps, leaving a trail of 1000 to its left as it moves. After $8n$ timesteps, the \triangleright head disappears of the right boundary. Thus, $TTS(f, X) = 2 \cdot |X| + 1$. It is not hard to see that this is the worst case scenario, so that

$$TTS^{worst}(F_\Theta^O)(n) = 2n + 1.$$

To compute the average case scaling, we use a very simple principle: suppose $A \subset \mathbb{C}$ is a set of initial conditions of *positive measure* in \mathbb{C} . That is, the ratio

$$\mu_{\leq n}(A) = |A_{\leq n}| / |\mathbb{C}_{\leq n}|$$

is bounded below by some positive number μ , for all n , where $A_{\leq n} = A \cap C_{\leq n}$. Then for any local rule F ,

$$TTS^{avg}(F)(n) \geq \mu \cdot \langle TTS(F, X) \rangle_{X \in A_{\leq n}}.$$

In other words, the average run time overall initial conditions is at least μ times the average over A . In the present case, let A consist of all initial conditions of the form $X = \triangleleft \circ Y$, as discussed above. The key point is now to notice that $\mu(A) \triangleq \liminf \mu_n(A) = 1/6$; that is because there are six states that F_{Θ}^O might use ($0, 1, \Delta_0, \Delta_1, \triangleright$, and \triangleleft), and the chance of the first agent's state being in any one of these states is 1 out of six. But we know that $TTS(f, X) = 2|X| + 1$ for all $X \in A$. Hence,

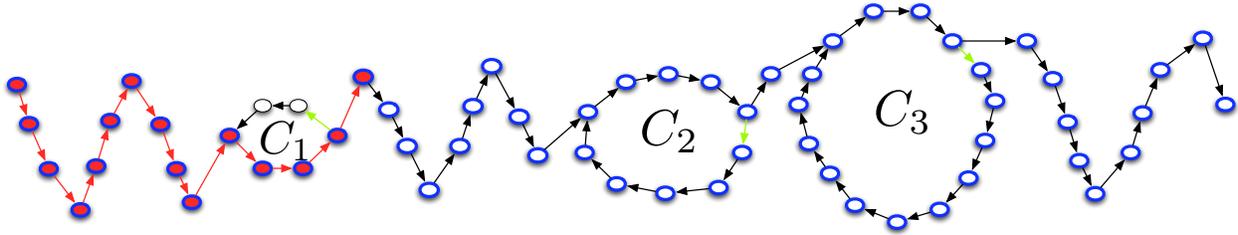
$$TTS^{avg}(F_{\Theta}^O)(n) \geq \frac{1}{3}n.$$

Summarizing, we see that the run-time of F_{Θ}^O scales linearly in $|X|$, both in the worst and average case.

Example 22 Now suppose Θ is the local check scheme of radius 10 for the pattern

$$T = \{(100000)^m (1000000000)^n (1000000000000000)^o \mid m, n, o \geq 3\}.$$

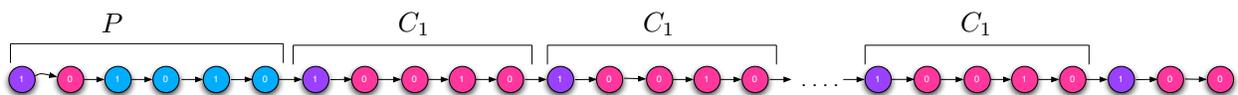
The graph $G(\Theta)$ is:



The graph has a single maximal acyclic path P of length 43 and three cycles, of length 6, 10, and 15 respectively. The cycles C_1, C_2 , and C_3 correspond to repeatable sequences 100000, 1000000000, and 1000000000000000, respectively. Take a choice function O whose first choices remain in cycles.

Since the greatest common divisor of the lengths of the three connected cycles is 1, and their LCM is 30, this pattern has solutions at all sizes, larger than 63 (and of course, some smaller ones well). Consider an initial condition X of size $n \gg 63$, of the form $X = P \circ \triangleright \circ Y$ where P is the acyclic path starting at the (unique) initial node of $G(\Theta)$, and ending just after the branch off from C_1 (this path is highlighted in red in the figure). Now, suppose that the residue of n modulo 5 is not 3, i.e. $\text{mod}(n, 5) \in \{0, 1, 2, 4\}$. In this case, no completion of the path P that is consistent with Θ can be of size n . In other terms, $\text{ext}^{\Theta}(P)$ contains no elements of size n . This is because all extensions of P are of size $33 + \text{gcd}(|C_2|, |C_3|) \cdot m = 33 + 5m$ (the 33 comes about because of the acyclic path); and any number of this form must be congruent to 3 modulo 5. To put all this another way, the only way to get the rest of the sizes is to use at least one copy of the length-6 cycle C_1 , but the initial path P bypasses C_1 so it is not accessible to any extension of P . Now, the trajectory of X under F_{Θ}^O necessarily explores all of these extensions of P of size $|X| - |P|$ or less, before having the turing head return to C_1 . The number of extensions of P of size N or less scales like $C \cdot (N - |P|)^2$, for some constant that satisfies $\frac{1}{m^{2r+1}} \leq C \leq 1$. By prop. 15, the trajectory of X under F_{Θ}^O must contain at least $C(n - |P|)^2$ steps, but take no more than $4C(n - |P|)^2$ timesteps for the turing head to come back to the left of P , i.e. if we denote by T the number of timesteps for that event to occur, there are constants $\alpha \geq 1/m^{2r+1}$ and $\beta \leq 5$ such that $\alpha(n - |P|)^2 \leq T \leq \beta(n - |P|)^2$.

Now, once the turing head comes back to the left P , it enters the cycle C_1 . Because O keeps the head within the C_1 cycle, and the head writes copies of C_1 all the way to the right-end of the configuration. Hence, after an additional $2(n - |P|)$ steps, the configuration has the form $P[1 : |P| - 1] \circ C_1^m \circ C_1[1 : k]$, where k is the residue of $m - |P| - 1$ modulo 6:



What will happen now is that the Turing head will reverse, and being to write out copies of C_2 and C_3 , removing one copy of C_1 at a time. Eventually, since the gcd of the cycle lengths is 1, a solution will be found. The key fact is that regardless of n , a solution will in fact be found within a constant number of time steps. The reason is:

Useful Fact 1 Suppose that d_1, \dots, d_k are positive integers whose greatest common divisor is D and whose least common multiple is M . Then any integer multiple of D larger than M can be as a sum of multiples of the d_i , $\sum_{i=1}^k n_i d_i$, in which $n_i \geq 0$ for all i , and in which $n_i \leq M/d_i$ for $i \geq 2$.

Most of this fact we've seen already in the previous chapter, in the proof of prop. 11. The only new part is that we can restrict $n_i \leq M/d_i$ for $i \geq 2$. In other words, as long as N is large enough we can find a solution to the diophantine equation $\sum_i n_i d_i = N \cdot D$ in which all of the variables but 1 (the first one, say) are small – bounded by a constant, M , that is independent of N . The import of this for the present situation is that, since the gcd of 6, 10, and 15, is 1, then a solution to $6m + 10n + 15o = N$ can be found with $n \leq 3$ and $m \leq 2$, as long as N is sufficiently large. As a result, at most 5 copies of C_1 need to be written over before a solution is found, regardless of the size of the configuration, and a solution is arrived at with γ timesteps for some constant $\gamma < 2 \cdot (30)^2$.

Hence,

$$\alpha(|X| - |P|)^2 \leq TTS(F_{\Theta}^O, X) \leq \beta(|X| - |P|)^2 + 2(|X| - |P|) + \gamma$$

In other words, for all X of the form $P \circ \triangleright \circ Y$, the runtime scales as the square of $|X|$. It is not hard to see that this X is the worst case scenario for configurations of size $|X|$ or less.

As for the average runtime, define A_n to be set of configurations that: (I) are of the form $P \circ \triangleright \circ Y$, (II) are of size less than n , and (III) whose size is not congruent to 5 modulo 3, and let $A = \cup_n A_n$. It is easy to compute that $\mu(A) > (m-1)/m^{|P|+2}$. Since $|P| \leq m^{2r+1}$, we have overall that

$$\Gamma_1 n^2 \leq TTS^{avg}(F_{\Theta}^O)(n) \leq TTS^{worst}(F_{\Theta}^O)(n) \leq \Gamma_2 n^2$$

for some constants Γ_1 and Γ_2 .

The key takeaway from example 22 is that most of the runtime of F_{Θ}^O come from the time that it takes to try out, and reject, the configurations in the set A . Like any other set of configurations, these configurations can be thought of as comprising a pattern. This pattern is locally checkable pattern, and its local check scheme corresponds to the subgraph of $G(\Theta)$ made up of the nodes in the path P , and all those nodes in $G(\Theta)$ to which the final node of P is connected. (These are the nodes that are outlined in dark blue in the figure above. This includes all nodes except for two nodes in the C_1 cycle.) In other terms, the runtime of F_{Θ}^O in example 22 is controlled by the size of a specific subpattern of $\Theta(\mathbb{C})$ corresponding to a subgraph of $G(\Theta)$. This idea can be generalized.

3.3.2 The Maximally Futile Subgraph

Definition 23 A marked graph is a ordered pair (G, H) where G is a directed graph and H is a path in G .

In this section we will define, for every local check scheme Θ and ordering function O , a special marked graph $(H_{O,\Theta}, \alpha_{O,\Theta})$ in which $H_{O,\Theta}$ is a subgraph of $G(\Theta)$. The graph $H_{O,\Theta}$, which I call the *maximally futile subgraph* of $G(\Theta)$ with respect to O , will control the asymptotic runtime of F_{Θ}^O in both the average and worst case. The shape of $\alpha_{O,\Theta}$ will determine the basic properties of $H_{O,\Theta}$. We will construct $H_{O,\Theta}$ by first constructing a graph $H_{O,\Theta}(P)$ for a set of paths P in $G(\Theta)$; then we will define $H_{O,\Theta}$ to be the “maximal” one, i.e. $H_{O,\Theta}(P^*)$ where P^* maximizes the size of $Growth(H_{O,\Theta}(P))$ over all such paths.

P is an *initial acyclic path* (i.a.p.) in a directed graph G , if the first node of P , P_0 , is an initial node of G , and P contains no cycles. Given a local check scheme Θ , ordering function O , and i.a.p. P in $G(\Theta)$, construct the marked graph $(H_{O,\Theta}(P), \alpha_{O,\Theta}^P)$ through the following iterative process.

1. **Base case:** Let $H_0(P) = \alpha_0^P = P$.

2. **Severl Definitions:** Suppose $H_{i-1}(P)$, a subgraph of $G(\Theta)$, and α_{i-1}^P , a path in $H_{i-1}(P)$, are given. Since the length of α_0^P is $|P|$, the length of α_{i-1}^P is $|P| + i - 1$. Hence, $\alpha_{i-1}^P(|P| + i - 1)$ is the final node of α_{i-1}^P ; denote it β_{i-1} . Then, make the following definitions: First, consider the set of nodes

$$\text{out}(\beta_{i-1}, G(\Theta)) \cap (G(\Theta) - H_{i-1}(P)).$$

These are the nodes that come out from the final node of α_{i-1}^P , that are NOT in $H_{i-1}(P)$. Among these nodes, let o_i be the one that is the minimal choice according to the ordering O . Let H_i^1 be the graph defined by adding the node o_i to $H_{i-1}(P)$, together with the edge (β_{i-1}, o_i) . That is:

$$H_i^1 = H_{i-1}(P) \cup (o_i, (\beta_{i-1}, o_i)).$$

Now, for any directed graph G and node $v \in G$, let $G_{\geq v}$ be the maximal subgraph $K \subset G$ such that for all $y \in K$, there is a path in G from v to y . Let

$$H_i^2 = H_{i-1}(P) \cup G(\Theta)_{\geq o_i} \cup (\beta_{i-1}, o_i).$$

As subgraphs of $G(\Theta)$, H_i^1 and H_i^2 define patterns \widehat{H}_i^1 and \widehat{H}_i^2 – the sets of configurations corresponding to maximal paths in H_i^1 and H_i^2 , respectively. Hence, we can consider the sets $Sizes(H_i^1)$ and $Sizes(H_i^2)$ of the sizes of H_i^1 - and H_i^2 -admissible configurations.

3. **Inductive Step:** If $S_i^2 \triangleq Sizes(\Theta) - Sizes(H_i^2)$ is infinite, then let

$$H_i(P) = H_i^2, \text{ and } \alpha_i^P = \alpha_{i-1}^P.$$

Otherwise (when S_i^2 is finite), then if $S_i^1 \triangleq Sizes(\Theta) - Sizes(H_i^1)$ is infinite, let

$$H_i(P) = H_i^1, \text{ and } \alpha_i = \alpha_{i-1} \circ o_i.$$

4. **Repeat** 1-3 until such i when the edge (β_i, o_{i+1}) is already present in $H_i(P)$, or until S_{i+1}^1 and S_{i+1}^2 are both finite.

For each P , the graph $H_{O,\Theta}(P)$ defines a corresponding pattern $\widehat{H}_{O,\Theta}(P)$ – the set of configurations corresponding to paths in $H_{O,\Theta}(P)$ starting at P_0 . Now, suppose we're given a configuration X of the form $Y \circ \triangleright \circ Z$, where the minimal acyclic path of Y is P . Notice that $H_{O,\Theta}(P)$ is defined precisely so that the trajectory of F_{Θ}^O on X searches through the $\widehat{H}_{O,\Theta}(P)$ -consistent extensions of X first. That is, the trajectory contains all configurations $W \circ \triangleright \circ Z$, for $W \in \widehat{H}_{O,\Theta}(P)$ of size $|X|$ or less and such that $W[1 : |Y|] = Y$, before it will contain any configurations $W \circ Z'$, with $W \in \Theta(\mathbb{C}) - \widehat{H}_{O,\Theta}(P)$.

Like any other pattern, we can measure the growth function $Growth(\widehat{H}_{O,\Theta}(P))(n)$ – denote it $g_{O,\Theta}(P)(n)$. Now, in light of proposition 12, there are constants $C_P \in (1/m^{2r}, 1)$, such that either $g_{O,\Theta}(P)(n) \sim C_P n^{m_P}$ or $g_{O,\Theta}(P)(n) \sim C_P \lambda_P^n$ where m_P is a non-negative integer and $\lambda_P > 1$. Hence the set of functions $\{g_{O,\Theta}(P)\}$, with P ranging over all IAPs, has (at least one) a strong pointwise maximal element. That is, there is a P^* , a constant V , and $N \in \mathbb{N}$ such that for all $n \geq N$ and all other IAPs P , $g_{O,\Theta}(P^*)(n) \geq V \cdot g_{O,\Theta}(P)(n)$. It is not hard to see that V can be chosen $\leq m^{2r+1}$, and N can be chosen $\leq m^{2(2r+1)m^{2r+1}}$ (probably much better bounds can be found). Take P_O^* to be any one of these maxima, and denote

$$H_{\Theta}(O) = H_{O,\Theta}(P_O^*); \text{ and } g_{O,\Theta}(n) = g_{O,\Theta}(P_O^*)(n).$$

I call $H_{O,\Theta}$ the *maximally futile subgraph of $G(\Theta)$, relative to choice function O* . Usually I will drop the O subscript from P_O^* , when the context is unambiguous.

$H_{\Theta,O}$ generalizes the concepts discussed in the examples in §3.3.1. The whole point of constructing $H_{O,\Theta}$ is that:

Proposition 16 [Strong Worst- and Average-Case Bounds] *There are constants A, B, C, D, E , and F , functions only of m and r , such that*

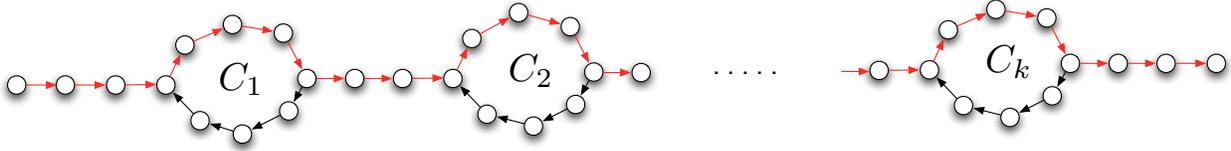
$$A \cdot g_{O,\Theta}(n) + B \cdot n \leq TTS^{avg}(F_{\Theta}^O)(n) \leq C g_{O,\Theta}(n) + D \cdot n$$

and

$$TTS^{worst}(F_{\Theta}^O(n)) \leq E \cdot \sum_{j \leq n} g_{O,\Theta}(j) + F \cdot n.$$

Since $g_{O,\Theta} \sim Cn^m$ or $\sim C\lambda^n$, $TTS^{worst}(F_{\Theta}^O(n)) \leq E \cdot n \cdot g_{O,\Theta}(n) + F \cdot n$. In words, the growth of the pattern associated with $H_{O,\Theta}$ gives, up to an additive linear factor, strong upper and lower bounds on the average run-time, and an upper bound on the worst-case runtime. Since the bound on the worst-case runtime is within at a multiplicative linear factor of the average case lower bound, the worst-case bound is close to being tight.

Example 23 Consider a pattern whose graph $G(\Theta)$ has multiple cycles, connected by a single maximal acyclic path P :



Denote the portion of the maximal path up to γ_i by p_i . Let c_i denote the length of the i -th cycle. Let $g_i = \gcd(c_i, \dots, c_k)$, the greatest common denominator of the lengths of the $k - i + 1$ cycles after (and including) C_i . For instance, in the graph in example 22, $g_1 = 1$, $g_2 = 5$, and $g_3 = 15$. Let l be the minimal j such that $g_{j-1} < g_j$; in example 22, l would be equal to 2. It is not hard to see that in this example, $H_{O,\Theta} = p_l \cup G(\Theta)_{\geq \gamma_l}$ for all ordering functions O ; in words, this consists of the cycles after (and including) C_l , and the maximal path up to γ_l . Hence, by prop. 12, the growth function $g_{O,\Theta}(n)$ scales as $O(n^{k-l+1})$. Thus, the average-case runtime of F_{Θ}^O scales as n^{k-l+1} , and the worst case no worse than n^{k-l+2} (in fact, it scales like n^{k-l+1}).

In general, the structure of $H_{O,\Theta}$ can be somewhat subtle. See figure 3.5 below for a table of Θ , O , choices and their corresponding graphs $H_{\Theta}(O)$. One important fact is that the growth function $g_{O,\Theta}(n)$ doesn't depend too much on the choice of ordering function O :

Proposition 17 [Ordering Invariance] *There is a constant $A < m^{2r+1}$ such that for any two ordering functions O and O' , $g_{O,\Theta}(n) \leq A \cdot \sum_{k \leq n} g_{O',\Theta}(k)$.*

This proposition is proved in the appendix. Again, since for all O , $g_{O,\Theta} \sim Cn^m$ or $\sim C\lambda^n$, $\sum_{k \leq n} g_{O,\Theta}(k) \leq n \cdot g_{O,\Theta}(n)$. The growth of $H_{O,\Theta}$ can therefore differ by at most a factor of n between various O . Thus, choice of ordering function O doesn't have too much impact on the run-time of F_{Θ}^O .

Related Work

The Naive Backtracking construction is an example of a somewhat vaguely-defined class of backtracking algorithms that are foundational in computer science [2]. The single-choice gradient algorithm is reminiscent of gradient constructions used in amorphous computing [1]. It is interesting to note that the latter is a special case of the former.

The results of this chapter further clarify the relationship of this theory to that of formal languages. In terms of the theory of formal languages, the computation in §3.3 shows that very subtle structure in the graph of the NDFA underlying the locally checkable language deeply influences the performance of the naive backtracking algorithm for the construction problem associated with that language. In addition, the maximally futile subgraph construction described in §3.3.2 and shown in Fig. 3.5, which characterizes the runtime, is a good example of the "more detailed" invariants that I first mentioned at the end of chapter 2. Like the other "more detailed" properties mentioned there, the maximally futile subgraph invariant is not one that is captured by the power series representation of regular languages, and is not preserved by wreath product.

The main subject of our work is the construction of robust solutions to locally checkable patterns. One of the central ideas of §3.2 is that the construction of the Naive Backtracking rule contains all the ingredients

$G(\Theta), O$	H, α	$g_{O,\Theta}(n)$
		$\sim C \cdot n$
		$\sim C \cdot n^2$
		$\sim C \cdot n^2$
		$\sim C \cdot n$
		$\sim C \cdot (1.0529)^n$
		$\sim C \cdot n$

Figure 3.5: In the left column, a set of graphs $G(\Theta)$, and choices for ordering function O (indicated by green arrows showing O 's "first choice" at choice points). In second column, the maximally futile subgraph $H_{O,\Theta}$, with the path $\alpha_{O,\Theta}$ highlighted in red arrows. In column 3, the growth rate of $H_{O,\Theta}$.

of a full-powered Turing machine, and as such is *not* easily characterized via simple languages. (Only in the case of the single-choice patterns, the local gradient rule may be thought of as a one-directional Turing machine, and so may be associated with a regular language.) The construction problem on comparatively simple patterns (i.e. regular locally generated languages) uses strictly more computational power than is contained in those languages classes themselves.

The ideas of this chapter connect to the literature on cellular automata through the idea of “particles.” Introduced in [27], a “particle” is a coherent virtual structure that emerges from the collective dynamics of the cellular automata. The “self-organized Turing heads” described in §3.2.1 are conceptually similar to particles. Both are coherent structure that are virtually constructed by the underlying dynamics. Just as particles can interact to produce new particles, so do the Turing heads interact. In one sense, the self-organized Turing machine can be thought of as performing “particle engineering” in which the coherent structures are stabilized to various perturbations in initial conditions and timing, and used to create fixed structures of some desired form as they move through the space. Future work will pursuing this connection further.

Chapter 4

Faster Algorithms and Pattern Classification

The Naive Backtracking algorithm constructed in chapter 3 can be extremely slow. In this chapter I show how to create much faster algorithms.

The basic reason why the Naive Backtracking algorithm is slow is that the self-organized Turing machine engages in many futile backtrackings. To create local rules which will cause faster pattern formation, we have to take advantage of more of the structure of locally checkable patterns. Specifically, in §4.1, I show how to use the graph structure of $G(\Theta)$ to design Less Naive Backtracking algorithms with linear runtime scaling. These algorithms fundamentally work by having the distributed signals sent by the Turing machine heads gather smart information that allows them to selectively prune the backtracking decision tree. In §4.2, I show that the runtime can be improved even further – to being constant, independent of system size – for a special subclass of patterns that are *locally patchable*. In §4.3 I step back and consider the various complexity classes of patterns in relation to the algorithms constructed in this and the previous chapter.

The key take-home result of this chapter is that there is a principled understanding of self-organizable patterns on which optimizations in local-rule design can be based. These optimizations occur at the level of “global-to-local compilation”, before any agents run the local rules – and can be integrated into the generic design process. Once local rules are produced by these constructions, they can then be run on the multi-agent systems.

4.1 A Linear Time Algorithm

As we saw in §3.3, the reason that the Naive Backtracking algorithm can be slow is that it engages in many futile backtrackings: the self-organized turing head reverses, constructs, and ultimately rejects, configurations that can’t possibly have solved the problem. Here I show how to speed up the algorithm by endowing the turing head with the ability to compute and propagate sufficient local information to avoid these futile backtrackings.

Consider the pattern

$$T_3 = \{(100000)^n(1000000000)^m(1000000000000000)^o \mid m, n, o \geq 3\}.$$

This pattern has a local check scheme Θ with radius 10, and its graph $G(\Theta)$ is depicted in example 22 of the previous chapter. There are three cycles; C_1 , of size 6; C_2 of size 10; and C_3 , of size 15. C_1 corresponds to the minimal segment 10^5 , C_2 to 10^9 , and C_3 to 10^{14} .

We saw that among the worst-case run-time configurations for the backtracking algorithm on this pattern were those of the form

$$X = (100000)^3 10000000 \circ 1 \circ Z$$

where $N \triangleq |X|$ is $\neq 3(5)$, and Z is any configuration. The reason that F_{Θ}^O takes a while to solve this configuration is that, after the birth of an \triangleright head in place of the 1 at position 27, the algorithm will then try

out all completions of $(100000)^3$ made up purely of C_2 and C_3 cycles, before changing the number of copies of the C_1 cycle that appear the beginning. But all configurations Y made of C_2 s and C_3 s alone are a multiple of 5 in size, so any X of the form $(100000)^3 \circ Y$ must be of $\equiv 3(5)$. Thus, if $N \not\equiv 3(5)$, all the completions like Y must fail; and as N scales, there will be $\sim O(N^2)$ such failures, each of which contributes to the runtime.

Now, suppose we could somehow: A) find a way for the \triangleright -head to determine the value of N modulo 5 and carry an indicator of this information along with it as it backtracks around the space; and B) have the head decide NOT to backtrack when the value of the indicator shows it would be impossible for a completion made up purely of C_2 s and C_3 s to work. The resulting algorithm's runtime when be reduced from quadratic to linear scaling in N . In what follows, I show how to achieve A) and B) with local rules, and for all local check schemes.

4.1.1 The Strategy

As defined in the previous chapter, the naive backtracking rule F_{Θ}^O has an "asymmetry". When the right-moving head \triangleright proceeds, as per **Rule 3**, it leaves behind a trail of meaningful, pattern-satisfying states in its leftward wake. It does this by propagating according to the choice $O(B(-2r-1 : -1), 1)$, where B is the local radius- $2r+2$ ball. In contrast, when the left-moving head \triangleleft proceeds as per **Rule 6**, it simply leaves behind a string of \triangleleft 's.¹ To modify the naive backtracking procedure to meet goal A), we remove this asymmetry so that both left- and right-moving heads can create meaningful patterns.

Given a local check scheme Θ and a Θ -accepted ball b , recall the left-compatibility set $L(b)$ defined in §2.2.1. In analogy with definition 22, define:

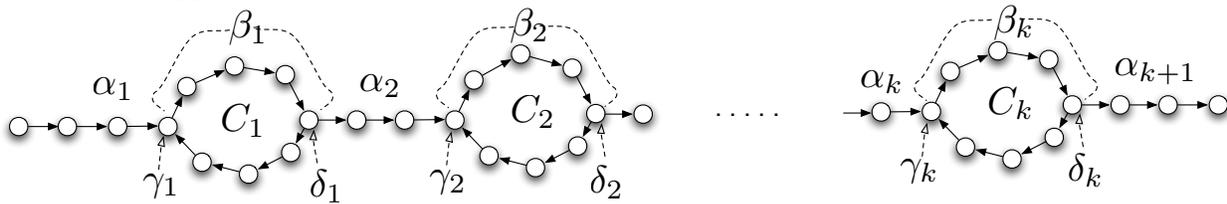
Definition 24 [Right-Choice Ordering Function] A right-choice ordering function for Θ is a mapping

$$P : \mathcal{B}_{2r+1} \times [m] \rightarrow [m]$$

such that $P(b, \cdot)$ is a bijection from $\{0, \dots, |L(b)| - 1\}$ to $L(b)$. Define the notations P_b^{-1} , \mathbf{P} , $\geq_{P,b}$ and B_P^{+1} in analogy with those defined for left-choice functions O .

Now, let's see by example what happens if we have the left-moving head propagate by choosing $P(B(1 : 2r+1), 1)$ instead of the trivial propagation.

Example 24 Suppose Θ is a local check scheme of radius r whose graph is, as in example 23,



We've denoted the cycles C_i ; the portions of the maximal acyclic path P in C_i by β_i , and the portions between C_{i-1} and C_i by α_i . The maximal acyclic path itself is

$$P = \alpha_1 \circ \beta_1 \circ \alpha_2 \circ \beta_2 \dots \circ \beta_k \circ \alpha_{k+1}.$$

The left-choice points are γ_i s and the right-choice points are δ_i s. Denote the portion of the maximal path up to, and just beyond, δ_i , by P_i . Let c_i denote the length of the i -th cycle. Let $g_i = \gcd(c_i, \dots, c_k)$, the greatest common denominator of the lengths of the $k-i+1$ cycles after (and including) C_i . Assume that O and P are again chosen to "remain" in cycles.

¹The asymmetry can easily be seen in eq. 3.2. The right-moving Turing operator is $\mathbb{P}_{\mathcal{G}_2, \Delta}^R$ where

$$\mathcal{G}_2(B) = \Theta[B_-] \wedge (\neg\Theta[B(-2r : 0)])$$

is the "border of Θ -correctness" condition and $\Delta(B) = \nabla_{\Theta}^+(B_-)$ is the local Θ -gradient. The left-moving operator is $\mathbb{P}_{\mathcal{G}_6, \emptyset}^L$, which has the trivial propagator \emptyset .

As we saw in the previous chapter, the “slow configurations” for F_{Θ}^O will in this case be those of the form $P_i \circ \triangleright \circ Z$ of size N for which that $\text{mod}(N, g_{i+1}) \neq \text{mod}(|P| - r, g_{i+1})$. Because, when projected onto the graph $G(\Theta)$, the left-most turing head is created just after cycle C_i and before cycle C_{i+1} , the trajectory of F_{Θ}^O on X will write out multiple copies of C_i , until it hits the right end. So far, F_{Θ}^O has used **Rule 1** (to start the head, and **Rule 2** and **Rule 3** to propagate the head. **Rule 4** is not applied, since when the \triangleright hits the end, there is no possible completion. Next, **Rule 5** will apply and create a left-moving \triangleleft head, so that configuration is

$$X' = P_{i+1} \circ C_{i+1}^m \circ C_{i+1}(1 : j) \circ \triangleleft$$

for some m and j .

At this point, under the operation of the Naive Backtracking algorithm F_{Θ}^O , the head would reverse and start writing a string of \triangleleft 's until it encounters the first choice point. But suppose now that we define a new algorithm \widetilde{F}_{Θ}^O that shares **Rules 1-5** with F_{Θ}^O , but replaces **Rule 6** with propagation according to $P(B(1 : 2r + 1), 1)$. By definition of a right-choice function P , the \triangleleft will move to the left, first writing α_{k+1} and then copies of C_k , so that after $2t$ timesteps (in the synchronous model), the configuration is

$$X'_t = P_{i+1} \circ C_{i+1}^{m_t} C_{i+1}(1 : j_t) \circ \triangleleft \circ C_k(l_t : |C_k|) \circ C_k^{n_t} \circ \alpha_{k+1}$$

for some n_t, m_t, j_t , and k_t .

The key realization is that as the \triangleleft moves along, the states of the $2r + 1$ agents to the right of \triangleleft provide an indicator of length modulo g_{i+1} . That is, if we denote by j the position of the \triangleleft in X'_t , the value of $\text{mod}(|X| - j, g_{i+1})$ can be determined from $X[t + 1 : t + 2r + 2]$. This is because, as an r -ball in $G(\Theta)$, $B = X[t + 1 : t + 2r + 2]$ determines a unique position p in the cycle C_{k+1} . Since Θ has radius r this position p can be determined by looking at just the $2r + 1$ states $X[t + 1 : t + 2r + 2]$ alone. If the agent at \triangleleft assumes that the states of the agents to its right have been according to $\mathcal{T}_{P_{\Theta}}^L$, then

$$t = |\alpha_{k+1}| - r + p + m|C_k|$$

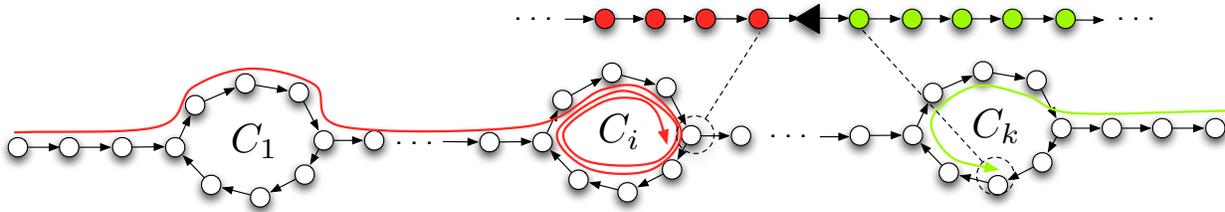
for some m . But

$$\text{mod}(t, g_{i+1}) = \text{mod}(|\alpha_{k+1}| - r + p + m|C_k|, g_{i+1}) = \text{mod}(|\alpha_{k+1}| - r + p, g_{i+1}).$$

The last expression is locally computable since p is, and $|\alpha_{k+1}|$ and r are known in advance.

Having seen how to achieve A), at least in a single example, how should we instruct the agent to act with this knowledge to achieve B)?

Example 25 Continuing with the setup of example 24, suppose X is a configuration like X' above, but assume nothing about the value of $\text{mod}(|X|, g_{i+1})$. Under \widetilde{F}_{Θ}^O as defined so far, when \triangleleft head arrives at the first non-maxed-out choicepoint, the local configuration looks like:



The agent hosting the \triangleleft can determine its relative position p in the cycle C_k , and knows that the number of agents to its right is of the form $p + m \cdot |C_k| + A$ where $A = |\alpha_{k+1}| - r$ is a predetermined constant. Now, if on the one hand, $\text{mod}(p + A, g_{i+1}) \neq 0$, then a head reversal at this point would *necessarily be futile*, since no number of the form $p + m \cdot |C_k| + A$ will ever be able to be written as $p + A + m \cdot g_{i+1}$. On the other hand, IF $\text{mod}(p + A, g_{i+1}) = 0$, then two things can be said: (i) there's a chance the head reversal will not be futile, since sometimes a number of the form $p + m \cdot |C_k| + A$ will be able to be written in the form $p + A + m \cdot g_{i+1}$; and (ii) if $m > \text{lcm}(c_{i+1}, \dots, c_k)$, then a head reversal here will *necessarily be successful*, in light of Useful Fact 1 discussed in the previous chapter. We will therefore want to define \widetilde{F}_{Θ}^O so that it reverses only when $\text{mod}(p + A, g_{i+1}) = 0$, since this criterion lets in all successful reversals, and only has at most constantly many futile ones, regardless of the size of the system.

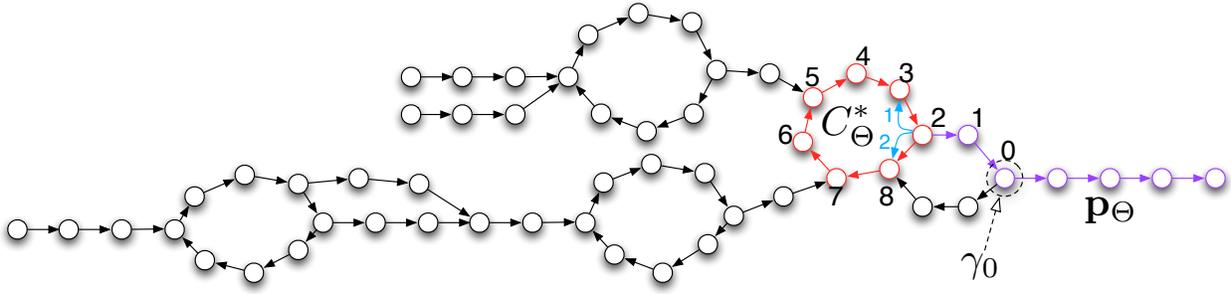


Figure 4.1: A generic single-terminus graph. γ_0 is the unique “gateway” node. The red arrows indicated the cycle C_{Θ}^* ; the purple arrows the path \mathbf{p}_{Θ} ; the blue numbers indicate the choices of the function choice function P ; and the black numbers indicate the position numbers $n(x)$ for $x \in A - A_0$.

Our strategy is now clear: achieve A) by having the left-moving turing head write copies of the “most terminal” cycle, from the right. This provides enough information about the system’s size modulo the greatest common denominator of the possible allowable segment sizes that B) can be achieved by comparing the local value of the position of the head to a “gcd criterion” for reversal. The goal now is to write rules analogous to **Rules 1-10** from the previous chapter, that will implement this strategy. I will first show how to do it for a special class of graphs (subsuming the two examples above) that I call *single terminus graphs* on which this process is relatively simple. Then, I extend the construction to all graphs by viewing them as combinations of single-terminus graphs.

4.1.2 Single Terminus Graphs

In the previous section, we used the “most terminal” cycle in the graph $G(\Theta)$ to build an indicator of system size. The fact that all configurations had a common “most terminal” cycle made it possible to use the same indicator at all times. The most general class of graphs that have a unique common “most terminal” cycle are the *single terminus* graphs:

Definition 25 A directed graph G is said to be single terminus if:

1. There is a unique “gateway” node $\gamma_0 \in G$ such that all maximal paths in G contain γ_0 , and which is contained in a non-trivial strongly connected component of G .
2. There is a unique path A_0 from γ_0 to the (therefore unique) terminal node of G . (And hence, all maximal paths in G end with p .)

A generic single terminus graph is shown in figure 4.1. The local check scheme in example 24 is evidently single terminus.

Suppose $G(\Theta)$ is be single terminus, let O and P be choice functions for $G(\Theta)$ which remain within strongly connected components. The first task we have is to locate in $G(\Theta)$ the appropriate “most terminal cycle”. To this end, make the inductive definitions

$$\gamma_i \triangleq \mathbf{P}(\gamma_{i-1}, 1) \text{ and } A_i \triangleq (\gamma_i, \gamma_{i-1}) \circ A_{i-1};$$

continuing until such i as $\gamma_i = \gamma_j$ for some $j < i$. Let A denote the resulting path, which is in the form a “cycle+line” graph. Let the cyclic portion be denoted C_{Θ}^* and the linear portion be denoted \mathbf{p}_{Θ} . In figure 4.1, the red arrows indicated the cycle C_{Θ}^* ; the purple arrows the path \mathbf{p}_{Θ} ; the blue numbers indicate the choices of the function choice function P ; and the black numbers indicate the positions of the nodes $\gamma_0, \gamma_1, \dots$ in $A - A_0$. In example 24, $C_{\Theta}^* = C_k$.

The path A plays the role of “most terminal segment”, with the cycle C_{Θ}^* being its repeated unit. In configurations of the form $Y \circ \triangleleft \circ \mathbf{p}_{\Theta}(k : |\mathbf{p}_{\Theta}|)$ or

$$Y \circ \triangleleft \circ C_{\Theta}^*(l : |C_{\Theta}^*|) \circ (C_{\Theta}^*)^m \circ \mathbf{p}_{\Theta},$$

the head will be able to determine the position k within \mathbf{p}_Θ or l within C_Θ^* . This will enable the agent to compute a “gcd criterion” and thereby determine whether to reverse or not.

Having found the most terminal unit, we now need to construct the “gcd criterion”. For any acyclic path in $G(\Theta)$, recall from chapter 2 the use of $\text{gcd}(p)$ to denote the gcd of the lengths of the cycles in strongly connected components of $G(\Theta)$ that intersect p . Let B be a ball of radius $2r + 2$. Let $b_1 = B(-2r - 1 : -1)$ and $b_2 = B(2 : 2r + 2)$, considered as r -balls.² Now, let the boolean function, $\Gamma(B)$, be defined as TRUE when:

1. $b_1 \in G(\Theta)$ and $b_2 \in A$;
2. if $b_2 \in A_0$, there is a path from b_1 to b_2 of length $\star(b_1) + \star(b_2)$;
3. if $b_2 = \gamma_i \in A - A_0 - C_\Theta^*$, there is a path from b_1 to γ_Θ of length $i + \star(b_1) + \star(b_2)$
4. if $b_2 = \gamma_i \in C_\Theta^*$, there is an acyclic path p in $G(\Theta)$ from b_1 to γ_Θ such that

$$|p| - i - \star(b_1) - \star(b_2) \text{ is divisible by } \text{gcd}(p).$$

Since the cycle C_Θ^* and line graph \mathbf{p}_Θ depend on the choice function P , so does Γ .

Intuitively, $\Gamma(B)$ is the local query that an agent makes asking if: 1) “is the local indicator of C_Θ^* -position, to be used to compute the gcd criterion, present?” and, 2) “If the gcd indicator is present, does it indicate that a reversal would be futile, or not?” $\Gamma(B)$ is a generalization of the “gcd criterion” developed by example in the previous section. In particular, suppose that X is a configuration of the form

$$X = Y \circ d \circ C_\Theta^*(l : |C_\Theta^*|) \circ (C_\Theta^*)^m \circ \mathbf{p}_\Theta$$

where $d \in \{\triangleright, \triangleleft, \triangleleft_i\}$, and Y is Θ -consistent. Then if $\Gamma(B_{2r+2}(|Y|, X))$ does not hold, $\text{ext}^\Theta(Y)$ cannot contain a configuration of size $|X|$. On the other hand, if $\Gamma(B_{2r+2}(|Y|, X))$ does hold, and $|X| - |Y|$ is sufficiently large (larger than the LCMs of the cycle lengths in $G(\Theta)$, say), $\text{ext}^\Theta(Y)$ will necessarily contain a configuration of size $|X|$.

Define the function $O^*(B)$ by:

1. setting it to be O -minimal state s for which $\Gamma(B[-2r - 2 : -1] \circ s \circ B[1 : 2r + 2])$, if such s exists
2. setting it to be $O(B[-2r - 1 : -1], 1)$ otherwise.

Intuitively, $O^*(B)$ is the value that the right-moving head should take on as it moves across the space. If the gcd criterion indicates that a certain choice is better than the first O -minimal choice, it takes that; otherwise (when the gcd indicator is not present or no state will satisfy it), it simply takes the O -minimal choice.

Let $\widetilde{L}_O^P[B]$ be the boolean defined by

$$\widetilde{L}_O^P[B] = \Gamma(B) \Rightarrow (\exists s \succ_{O, b_1} B(0) \mid \Gamma(B[-2r - 2 : -1] \circ s \circ B[1 : 2r + 2])).$$

Moreover, when $\Gamma(B)$ holds but $\widetilde{L}_O^P(B)$ fails, let $s^*(B)$ be the O -minimal $s \succ_{O, b_1} B(0)$ for which $\Gamma(B[-2r - 2 : -1] \circ s \circ B[1 : 2r + 2])$.

Intuitively, L_O^P is the analog of the condition L used in defining F_Θ^O in the previous chapter. Essentially, it says “either 1) the gcd indicator is not present or is indicating a turn here is futile, or 2) a turn might not be futile, but previous choices here have maxed out the possible set of next choices that would lead to non-futile reversals.” Hence, if L_O^P holds, the head should continue on and NOT reverse. On the other hand, if a head turn would not be futile, and the choices have not been maxed out, $s^*(B)$ is the O -minimal such non-futile choice.

Now, define the local rule $\widetilde{F}(B)$ with radius $2r + 3$ by the following rules:

- **Rule 1** sets up the creation of a right-moving turing head when necessitated by a local mistake. Formally,

$$\widetilde{\alpha}_1[B] \triangleq \Theta_{-r-1}[B] \wedge (\Theta_{-r}^-[B]) \vee \eta[B] \wedge (B(0) \in S) \quad \Rightarrow \quad \widetilde{F}(B) = \triangleright.$$

²Technically, this means that b_1 is the r -ball whose underlying string is $B(-2r - 1 : -1)$ and for which $\star(b_1) = \max(0, |B(-2r - 1 : -1)| - r)$. If $|B(-2r - 1 : -1)| = 0$, b_1 is the empty left-ball. Similarly b_2 is the r -ball whose underlying string is $B(2 : 2r + 2)$ and for which $\star(b_2) = \min(r + 1, |B(2 : 2r + 2)| - r)$. If $|B(2 : 2r + 2)| = 0$, then b_2 is the empty right-ball.

- Otherwise, **Rule 2** propagates the \triangleright head to the right. Formally,

$$\left. \begin{array}{l} \tilde{\alpha}_2[B] \triangleq (B(-1) == \triangleright) \wedge \Theta_{-r-2}[B] \\ \tilde{\gamma}_1[B] \triangleq \Theta_{-r-1}[B] \wedge (B(1) = B(0) = \triangleright) \wedge (\exists j | \Theta[B[-2r : -1] \circ j]) \end{array} \right\} \Rightarrow \tilde{F}(B) = \left\{ \begin{array}{l} \triangleright \\ O^*(B) \end{array} \right.$$

- Otherwise, **Rule 3** halts the right-moving turing head by disappearing it off the right end, when a completion is possible. Formally,

$$\tilde{\gamma}_2[B] \triangleq (B(0) = \triangleright) \wedge \Theta_{-r-1}[B] \wedge (\exists j | b_j[B]) \quad \Rightarrow \quad \tilde{F}[B] = \min_O \{j \in [0, m-1] | b_j[B]\}.$$

- Otherwise, **Rule 4** reverses a right-moving turing head into a left-moving head, when the right-moving head is unable to move forward because a completion is impossible. Formally,

$$\left. \begin{array}{l} \tilde{\beta}_1 \triangleq \Theta_{-r-1}[B] \wedge (B(0) = \triangleright) \wedge (\wedge \Theta^-[B[-2r : -1] \circ j]) \\ \tilde{\beta}_2[B] \triangleq (B(0) = \triangleright) \wedge \Theta_{-r-1}[B] \wedge (\wedge_{j=0}^{m-1} b_j^-[B]) \end{array} \right\} \Rightarrow \tilde{F}(B) = \triangleleft.$$

- Otherwise, **Rule 5** propagates the \triangleleft head leftward, when a) the gcd indicator is not present, or b) when it is and indicates a reversal is futile or c) when the set of non-futile choices has already been maxed-out relative to O . Formally,

$$\left. \begin{array}{l} \tilde{\beta}_4[B] \triangleq \Theta_{r+1}[B] \wedge (B(-1) = B(0) = \triangleleft) \wedge (\exists j | \Theta[j \circ B[1 : 2r+1]]) \\ \tilde{\beta}_3[B] \triangleq (B(1) = \triangleleft) \wedge \tilde{L}_O^P(B) \wedge ((B(0) = \triangleleft) \vee (\tilde{\beta}_4[B^{+2}]^-)) \end{array} \right\} \Rightarrow \tilde{F}(B) = \left\{ \begin{array}{l} P(B[1 : 2r+1], 1) \\ \triangleleft \end{array} \right.$$

In case a), represented by $\tilde{\beta}_4[B]$ not holding, the left-moving head writes a string of \triangleleft 's. In b), or c), represented by $\tilde{\beta}_4[B]$ holding, the gcd indicator is propagated by the left-moving head writing Θ -consistent states from the right.

- Otherwise, **Rule 6** reverses the turing head at a choicepoint where a reversal is indicated to not be futile. Formally,

$$\left. \begin{array}{l} \tilde{\epsilon}[B] \triangleq ((B(0) = \Delta_{B(-1)}) \vee (B(0) = \triangleleft)) \wedge \Gamma(B^{-1}) \wedge (\tilde{L}_O^P)^-[B^{-1}] \\ \tilde{\delta}[B] \triangleq (B(1) = \Delta_{B(0)}) \wedge \Gamma(B) \wedge (\tilde{L}_O^P)^-(B) \\ \tilde{\alpha}_4[B] \triangleq (B(0) = \triangleleft) \vee ((\exists j | B(0) = \Delta_j)) \end{array} \right\} \Rightarrow \tilde{F}(B) = \left\{ \begin{array}{l} \Delta_{B(-1)} \\ s^*(B) \\ \triangleright \end{array} \right.$$

$\tilde{\epsilon}$ halts the left-moving turing head when reversal won't be futile; $\tilde{\delta}$ clicks up the choice to the next non-futile choice; and $\tilde{\alpha}_4$ completes the turn of the head, but only after $\tilde{\delta}$ has been acted upon. $\tilde{\alpha}_4$ also resets all partially formed heads that do not conform to the structure required in their other rules.

This local rule is summarized in the inset 2. In contrast to the Naive Backtracking algorithm, for this ‘‘Less Naive’’ Backtracking algorithm, 1) every reversal that would have been made by F_Θ^O but that \tilde{F}_Θ ignores would have been futile in the first place, and 2) \tilde{F}_Θ makes only a constant number of reversals that end up being futile, independently of the size of the system. As a result,

Proposition 18 *Suppose Θ is a local check scheme of radius r over m states, and $G(\Theta)$ is a single terminus graph. \tilde{F}_Θ is a solution to $\Theta(\mathbb{C})$ in an all live timing models, and in the synchronous timing model,*

$$TTS^{\text{worst}}(\tilde{F}_\Theta)(n) \leq 8 \cdot n + C$$

where C is a constant depending only on m and r . That is, the asymptotic runtime of \tilde{F} scales linearly in system size.

The proof of this result is found in an appendix to this chapter.

Algorithm 2: Less Naive Backtracking Algorithm for Single Terminus Graphs

```

if  $\widetilde{\alpha}_1[B] \vee \widetilde{\alpha}_2[B]$  then
  |  $\widetilde{F}(B) = \triangleright$ 
else if  $\widetilde{\beta}_1[B] \vee \widetilde{\beta}_2[B] \vee \widetilde{\beta}_3[B]$  then
  |  $\widetilde{F}(B) = \triangleleft$ 
else if  $\widetilde{\beta}_4[B]$  then
  |  $\widetilde{F}(B) = P(B[1 : 2r + 1], 1)$ 
else if  $\widetilde{\gamma}_1[B]$  then
  |  $\widetilde{F}(B) = O^*(B)$ 
else if  $\widetilde{\gamma}_2[B]$  then
  |  $\widetilde{F}(B) = \min_O \{j \in [0, m - 1] \mid b_j[B]\}$ 
else if  $\widetilde{\delta}[B]$  then
  |  $\widetilde{F}(B) = s^*(B)$ 
else if  $\widetilde{\epsilon}[B]$  then
  |  $\widetilde{F}(B) = \Delta_{B(-1)}$ 
else if  $\widetilde{\alpha}_4[B]$  then
  |  $\widetilde{F}(B) = \triangleright$ 
else
  |  $\widetilde{F}(B) = B(0)$ .
end

```

4.1.3 The General Case

In the previous section, I showed how to construct a linear-time algorithm for single terminus graphs. Here, I describe how to generalize this construction to all local check scheme graphs.

All finite directed graphs $G(\Theta)$ can be written as a finite union of (possibly overlapping) single terminus graphs. To see this, define the boolean

$$T_R(b) = 1 \text{ if } b \in G(\Theta) \text{ and } scc(p) = 0 \text{ for all paths } p \text{ with } p_0 = b.$$

In words, there is no path from b to a cycle in $G(\Theta)$, so b is a “right-terminal” node. Let $A_0 = \{b \mid T_R(b)\}$ denote the set of all right-terminal nodes in $G(\Theta)$. For instance, in figure 4.2a, A_0 is comprised of the nodes shaded red. The set

$$A_1 \triangleq \cup \{in(b, G(\Theta)) \mid b \in A_0\} - A_0$$

contains all the nodes that are “just prior” to the right-terminal nodes. These are the nodes shaded orange fig. 4.2a. Now for each $\gamma_0 \in A_1$, let $P(\gamma_0)$ denote the set of maximal paths p in A_0 for which $p_0 \in out(\gamma_0, G(\Theta))$, and let $p' = \gamma_0 \circ p$. Define

$$G_{\gamma_0, p} \triangleq G(\Theta)_{\leq \gamma_0} \cup p'.^3$$

For every $\gamma_0 \in A_1$ and $p \in P(\gamma_0)$, $G_{\gamma_0, p}$ is a distinct single terminus graph with γ_0 as the distinguished “gateway” node. The set of such $G_{\gamma_0, p}$ covers G , i.e.

$$G(\Theta) = \bigcup_{\substack{\gamma \in A_1 \\ p \in P(\gamma)}} G_{\gamma, p}.$$

The results of this decomposition process for the graph in fig. 4.2a are shown in fig. 4.2b.

Each $G_{\gamma_0, p}$ has a unique “most terminal cycle+line” associated with it. In particular: for each $\gamma_0 \in A_1$, make the inductive definition

$$\gamma_i \triangleq \mathbf{P}(\gamma_{i-1}, 1), \text{ and } A_{\gamma_0}^i = (\gamma_i, \gamma_{i-1}) \circ A_{\gamma_0}^{i-1}.$$

³Recall the definition used in chapter 3, that for any directed graph G and node $x \in G$, $G_{\leq x}$ is graph induced by G on the nodes $y \in G$ for which there is a path from y to x , including x itself.

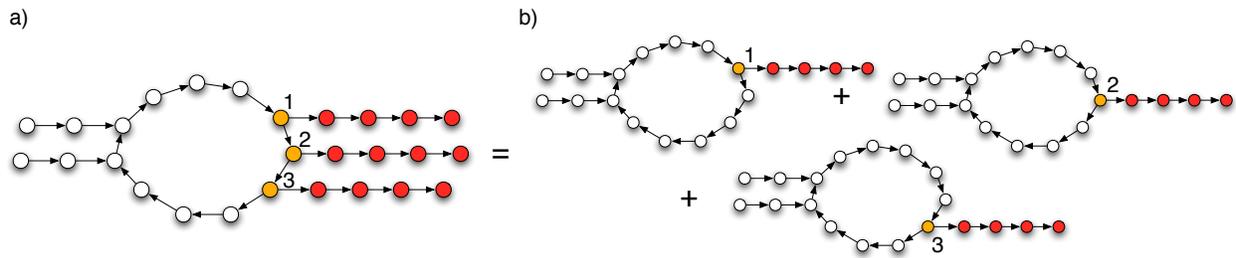


Figure 4.2: a) A graph with right-terminal nodes shaded red and “almost-terminal” nodes shaded orange. b) The decomposition of the graph from a) into three single-terminus components.

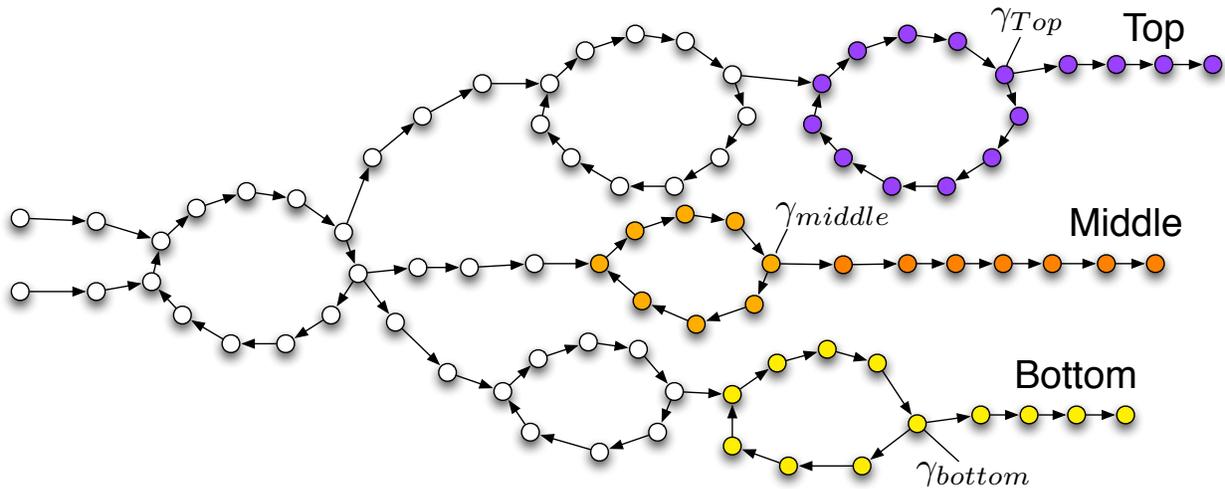


Figure 4.3:

Repeat this until such i as $\gamma_i = \gamma_j$ for some $j < i$. Let A_{γ_0} denote the resulting path. Each A_{γ_0} is of the form of a “cycle+path” graph; let C_{γ_0} denote the cyclic portion and L_{γ_0} denote the linear portion. For $p \in P(\gamma_0)$, $A_{\gamma_0,p} = A_{\gamma_0} \cup p'$ is the “most-terminal cycle + line” for $G_{\gamma_0,p}$, consisting of the cycle C_{γ_0} attached to the path $L_{\gamma_0} \circ p'$. In the algorithm $\widetilde{F}_{G_{\gamma_0,p}}$ as constructed in the previous section, the left-moving turing head will write states consistent with $A_{\gamma_0,p}$ for use as the “gcd indicator.”

The goal now is to combine the local rules \widetilde{F}_{G_i} for each single-terminus component G_i into a local rule for the overall pattern. To see how to do this, let’s pick a simple example. Consider a local check scheme whose graph is depicted in fig. 4.3. This graph has three single-terminus components: G_{Top} , G_{middle} , and G_{bottom} . The “most-terminal” portions of each are colored purple, orange, and yellow, respectively. A_1 consists of three nodes, γ_{Top} , γ_{middle} , and γ_{bottom} , one in each of the three “most-terminal” portions.

I’m now going to (informally) define a local rule \widetilde{F} that will solve this pattern. Let’s concentrate first on

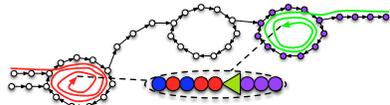
the single-terminus graph G_{top} (i.e. this one: ). Consider an initial configuration $X = Y \circ \triangleright \circ Z$, where where Y is a path in G_{Top} containing repeats of the “first” cycle, and Z is an arbitrary configuration.

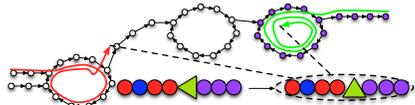


Viewed as a path in $G(\Theta)$, this looks like:

. Suppose we start applying $\widetilde{F}_{G_{top}}$ to this initial

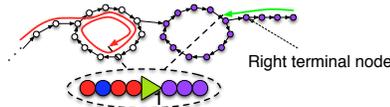
configuration. $\widetilde{F}_{G_{top}}$ will propagate the \triangleright head until it reaches the right end, in the configuration $Y \circ C^m \circ \triangleright$. The

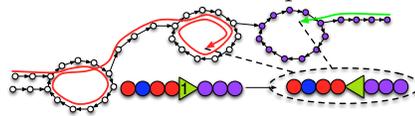
head will reverse then reverse and propagate leftward:  . As it propagates, the head leaves behind a trail of “purple states” to serve as the gcd indicator. If it reaches a choicepoint that satisfies the gcd criterion before hitting the left end, the head will take on the $\Delta_{B(-1)}$ state, and the agent to its

left will “click up” up to the next option:  . Define \tilde{F} , the algorithm that is supposed to solve the whole pattern, identically with $\tilde{F}_{G_{Top}}$ up until this point. Now, however, let’s define \tilde{F} slightly differently from $\tilde{F}_{G_{Top}}$. The latter will create \triangleright in place of the Δ state, and this would propagate all the way to the right end of the configuration. Define \tilde{F} instead so that it replaces \triangleright with a new state, call it

\triangleright_1 :  . This new state \triangleright_1 will operate like \triangleright , but instead of propagating all the way to the right end, \triangleright_1 will instead stop at the first right-terminal node in G_{Top} that it encounters. That is, the \triangleright_1 will stop propagating when the $2r + 1$ agents to its right, considered as an r -ball b , satisfy the boolean

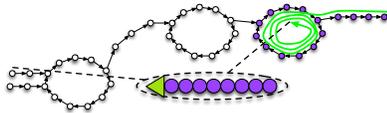
$T_r(b)$ defined above. The trajectory will look like:



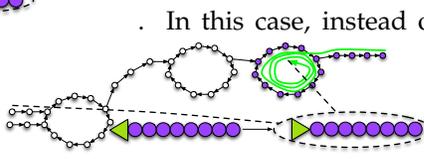
Upon encountering a right-terminal node, define \tilde{F} to have \triangleright_1 reverse into a \triangleleft head:  . This \triangleleft will then propagate as it did previously, i.e. until it encounters the first choicepoint where the gcd criterion holds. The above steps repeat: each time the \triangleleft reverses, it becomes a \triangleright_1 ; which then propagates up until the first right-terminal node in encounters and no farther; and reverses if need be. This continues until a solution is found.

If a solution is not found, however, the \triangleleft will eventually propagate all the way to the left end of the

configuration:



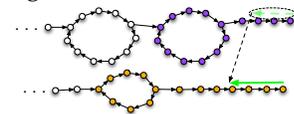
reverse back to the original \triangleright head:



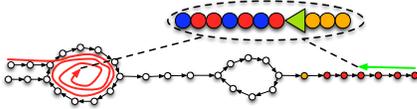
In this case, instead of reversing into \triangleright_1 , define \tilde{F} to Under the original algorithm $\tilde{F}_{G_{Top}}$, the \triangleright will propagate all the way to the left end; however, define \tilde{F} so that it propagates \triangleright to the first

left-choicepoint that is a terminal node, relative to the right choice function P :

There, the \triangleright will reverse, after the state of the agent to its right “clicks up” to the next choice, relative to P . That is, \triangleright propagates past the first right-terminal point, but stops at the first point thereafter which is a left-choicepoint and makes the “next choice”. For this graph $G(\Theta)$, that will mean that the path from the right terminal node will shift from the “top” branch to one of the other branches (say, the middle branch):



Now the algorithm operates as if it were on G_{middle} , and the configuration looks



like: $\tilde{F}_{G_{middle}}$. The \triangleleft head propagates leftward, propagating the “orange states” as the gcd indicator for this component; until it encounters the first choicepoint where the gcd criterion holds. It then reverses to \triangleright_1 , propagates right until the first right-terminal node (which is now on the terminus associated with G_{middle}); and repeats. If the configuration is still unsolved by the time the \triangleleft encounters the left end, the head reverses to the original \triangleright head, which travels rightward until it encounters the next right-terminal choicepoint. There, it moves the to the next choice relative to the P choice function, which will in this case be to shift the path from the right from the G_{middle} terminus to the G_{bottom} terminus; and the steps repeat.

So now we have a strategy that will work for all graphs:

- Decompose $G(\Theta) = \cup G_i$, where each G_i is single terminus.
- Define $\tilde{F}_{G(\Theta)}$ by combining the algorithms \tilde{F}_{G_i} on their domains of definition, with the modifications that:
 - When a \triangleleft head reverses as a result of a satisfied gcd criterion, the reversed head is in the new state \triangleright_1 .
 - When a \triangleright_1 head encounters a right-terminal node, it reverses to \triangleleft .
 - When a \triangleleft head reverses as a result of hitting the left end, it reverses to \triangleright .
 - When a \triangleright head encounters a left-choicepoint that is a (right) terminal node, it sets the next choice and reverses to \triangleleft .

The algorithm $\tilde{F}_{G(\Theta)}$ so defined will search “reverse lexicographically” through the single-terminus components, trying out all the gcd-criterion-approved completions of the i -th terminus become moving on to those of the $(i + 1)$ -st terminus. The runtime of $\tilde{F}_{G(\Theta)}$ on initial condition $|X|$ will simply be (at worst) the sum of the runtimes for each single-terminus component, so that

$$TTS^{worst}(\tilde{F}_{G(\Theta)})(n) \leq 8 \cdot |\mathcal{G}| \cdot n + A,$$

where $|\mathcal{G}|$ is the number of single-terminus component in $G(\Theta)$ and A is a constant depending only on m and r .

There is one small snag in implementing this strategy in general. The example in fig. 4.3 that we’ve just worked through is special in that there is only one “gateway node” γ_0 in any given terminal cycle C_γ . In contrast, the graph in fig. 4.2 has three γ_0 nodes attached to the same terminal cycle. This situation requires attention: if there is a cycle $C = C_{\gamma_0} = C_{\gamma'_0}$, then an agent trying to evaluate the “gcd criterion” in C will not be able to locally determine whether it “came from” γ_0 or γ'_0 . Of course, to accurately use the gcd criterion the agent will need to know this information. Hence, the head \triangleleft will have to carry with it an indicator of the identity of the node γ_0 . The simplest way to do this is to introduce states \triangleleft_{γ_0} for each $\gamma_0 \in A_1$.

Since A_1 is a subset of $G(\Theta)$, $|A_1| \leq m^{2r+1}$. Hence we need to add at most m^{2r+1} extra states. The details of implementing this algorithm are straightforward, and are given in an appendix §C. As a result, we have:

Theorem 3 *All locally checkable patterns T over m states are solvable by a local rule with radius $2r(\Theta) + 3$ using $m^{2r+1} + 2m + 2$ states, whose worst-case runtime in the synchronous timing model scales linearly with system size.*

4.2 Locally Patchable Patterns

A pattern is *locally patchable* (LP) if isolated local errors can always be fixed by local corrections. Formally,

Definition 26 [Locally Patchable Check Schemes] *A local check scheme Θ of radius r is k -locally patchable if for all configurations X such that $|X| \in \text{Sizes}(\Theta)$, and all $1 \leq i_1 \leq j_1 < i_2 \leq j_2 \leq j \leq |X|$, such that $i_2 > j_1 + 2k$, then whenever*

$$\Theta[B_r(l, X)] = 1 \text{ for all } l \notin [i_1, j_1] \cup [i_2, j_2],$$

there are subconfigurations Y_1 and Y_2 of size $\min(k, i_1) + k + j_1 - i_1 + 1$ and $k + \min(k, |X| - j_2) + j_2 - i_2 + 1$, respectively, such that

$$X' = X[1 : \max(1, i_1 - k)] \circ Y_1 \circ X[j_1 + k + 1 : i_2 - k - 1] \circ Y_2 \circ X[\min(|X|, j_2 + k) : |X|]$$

is a Θ -admissible configuration of size $|X|$. Θ is said to be locally patchable if it is k -locally patchable for some finite k .

In the above definition, the statement that $\Theta[B_r(l, X)] = 1$ for $k \notin [i_1, j_1] \cup [i_2, j_2]$ intuitively encodes the idea that all the errors in X are isolated to the regions $[i_1, j_1]$ and $[i_2, j_2]$. A k -locally patchable pattern will allow whatever errors there are to be fixed by replacing the regions $[i_1, j_1]$ and $[i_2, j_2]$ – together with buffer zones of size k on either side – with independent “local patches” Y_1 and Y_2 . Of course, the region $[i_2, j_2]$ can be taken to be empty, implying that single isolated errors in locally patchable patterns must be locally fixable. Moreover, the definition implies that patchability will hold for any number of regions $[i_1, j_1], \dots, [i_n, j_n]$, as long as $i_l > j_{l-1} + 2k$.

Example 26 Consider the pattern T_1 generated by freely alternating instances of the word 10 with the word 100, i.e.

$$T_1 = \bigcup_{k \in \mathbb{N}} \{(100)^{n_1} (10)^{n_2} \dots (100)^{n_k} \mid n_i \in \mathbb{N}\}.$$

Notice that if X and Y are T_1 -admissible configurations, and k is such that $X(k) = 1$, then $X[1 : k-1] \circ Y \circ X[k : |X|]$ is a T_1 -admissible configuration of size $|X| + |Y|$. Similarly, if k_1 is such that $X(k_1) = X(k_2) = 1$ then $X[1 : k_1 - 1] \circ X[k_2 - 1 : |X|]$ is T_1 -admissible.

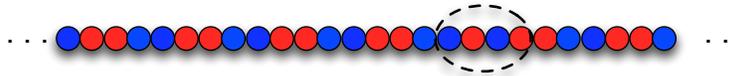
Now suppose X is a configuration and i, j are such that $\Theta[B_r(k, X)] = 1$ for $k \notin [i, j]$. Let i_1 be the maximal $l < i$ such that $X(l) = 0$. Evidently, $i - i_1 < 2$. Similarly, let j_1 be the minimal $l > j$ such that $X(l) = 1$; and $j_1 - j < 2$. Since $X(i_1 + 1) = X(j_1 + 1) = 1$, $X[1 : i_1] \circ X[j_1 : |X|]$ is T_1 -admissible. On the other hand, since T_1 contains configurations of all sizes larger than 1, pick be a T_1 -admissible configuration Y of size $j_1 - i_1$. Then $X[1 : i_1] \circ Y \circ X[j_1 : |X|]$ must be T_1 -admissible. Hence T_1 is 2-locally patchable.

Example 27 Consider the pattern T_2 generated by freely alternating combinations of the sequences 1001 and 100112001, discussed in example 19 in chapter 2. This pattern has a local check scheme Θ radius 3, whose graph $G(\Theta)$ has two irreducible cycles, one of length 4, denote C_1 , and other of length 9, denoted C_2 . Notice first that if X is a T_2 -admissible configuration, and k is such that $X[k-3 : k] = 1001$ or $X[k-8 : k] = 100112001$, and Y is T_2 -admissible, then $X[1 : k] \circ Y \circ X[k+1 : |X|]$ is T_2 -admissible. Similarly, if $k_1 < k_2$ are such that $X[k_1 : k_1 + 3] = 1000$ or $k[k : k + 8] = 100112001$ and $X[k_2 - 3 : k_2] = 1000$ or $X[k_2 - 8 : k_2] = 100112001$, then $X[1 : k_1 - 1] \circ X[k_2 + 1 : |X|]$ is T_2 -admissible. Now, as discussed in example 19, any size n greater than 36 is in $Sizes(T_2)$. Following the argument as in the previous example shows that T must be $(36/2 = 18)$ -locally patchable.

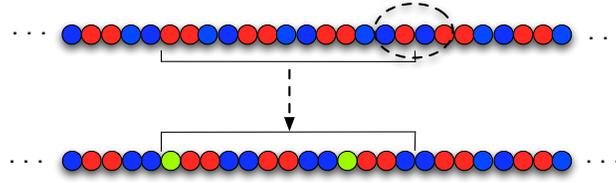
For instance, consider the subconfiguration

$$Z = \dots (1001)^4 10(1001)^2 \dots$$

Visually:



in which blue represents state 1 and red represents 0. This subconfiguration is a repetition of the 4-cycle C_1 several times – except for one “bad spot” at the location indicated by the dotted oval, where Θ is not satisfied. This mistake can be patched by replacing the states of the 18 agents to the left of the bad spot with 100112001100112001, that is, two copies of the cycle C_2 . Pictorially:



in which green color represents state 2.

Example 28 On the other hand, no non-trivial repeat pattern is locally patchable. For instance in the pattern $T_{10} = \{(10)^n | n \in \mathbb{N}\}$, the configuration

$$(10)^n 1 (10)^n 1$$

is of admissible size for every n . However no separate patches Y_1 and Y_2 of fixed size will allow for the correction of the error at positions $2n + 1$ and $4n + 2$ – all of the intervening states between $2n + 1$ and $4n + 2$ have to be modified to generate a T_{10} -admissible configuration.

What property do the patterns T_1 and T_2 share than T_{10} lacks? Evidently T_{10} only admits configurations of even sizes, whereas the other two patterns admit configurations of all sizes (past a certain fixed size). However, this is not enough to discriminate, since:

Example 29 The pattern

$$T_3 = \{(10)^n | n \in \mathbb{N}\} \cup \{(10)^n 1 | n \in \mathbb{N}\}$$

is not locally patchable, since there no fixed-size local patches that can correct the errors in the configurations $(10)^n 1 (10)^n$ in T_3 , even though T_3 admits configurations of all sizes.

Patterns T_1 and T_2 also have alternatable minimal segments, leading to exponential pattern growth, while T_{10} only has single minimal segment. This property is also not enough, since:

Example 30 The pattern T generated by freely alternating the segments 1000 and 10, considered in example 17 in chapter 2 is not locally patchable. It too is “foiled” by the configurations $(10)^n 1 (10)^n 1$.

The key difference between examples 26 and 27 and the others is that the former have both these two properties, possessing alternatable minimal segments whose sizes can be combined to form any (sufficiently large) integer. This fact is reflected in the graphs of associated local check schemes for these patterns. In both cases, the graphs contain a single non-trivial strongly connected component, each of which in turn contains two cycles whose lengths are relatively prime. A simple generalization of the above arguments shows that:

Proposition 19 A local check scheme Θ of radius r is k -locally patchable for some finite k if and only if (I) $G(\Theta)$ is acyclic, in which case $k < |G(\Theta)|$; or (II) for all strongly connected components C in $G(\Theta)$, $\gcd(C) = 1$, in which case, $k \leq \text{lcm}(C) \leq (1/2)m^{(2r+1)m^{2r}}$.

As a result, if a locally checkable pattern T locally patchable, then:

- Either $\text{Sizes}(T)$ is finite (when $G(\Theta)$ is acyclic) or its complement in \mathbb{N} is (when $G(\Theta)$ contains at least one cycle).
- $\text{Growth}_\Theta(n)$ is either eventually 0 (when $G(\Theta)$ is acyclic), linear (when $G(\Theta)$ contains only self-loops) or exponential (when $G(\Theta)$ contains nontrivial cycles).

The reason that I’ve introduced the notion of locally patchable patterns, is that we can improve significantly from linear runtime for solutions to these patterns. Call a pattern T *weakly locally patchable* if there is a local check scheme Θ for T such that $\Theta(C)$ is locally patchable.⁴

⁴It is easy to see by the same reasoning as is behind prop. 19 that a locally checkable pattern is weakly locally patchable if the graph associated with the pattern has at least one strongly connected component C with $\gcd(C) = 1$.

Proposition 20 *Every weakly locally patchable pattern admits a local rule solution F_T whose average-case runtime scales at worst $O(\log(n))$ in system size n , for all live uniform timing models; and the runtime improves to $O(1)$ in bounded asynchronous timing models.*

I will demonstrate this result by construction of a local rule that achieves the advertised bounds. The construction naturally splits into two steps, in which (I) arbitrary initial configurations are rendered “almost right” except for isolated errors; and then (II) in which those errors are patched. I will describe step (II) first.

4.2.1 Dynamic Local Patching

Let’s return to the pattern in example 27, the free alternation of the minimal segments 1001 and 100112001.

Example 31 Consider all subconfigurations of the form

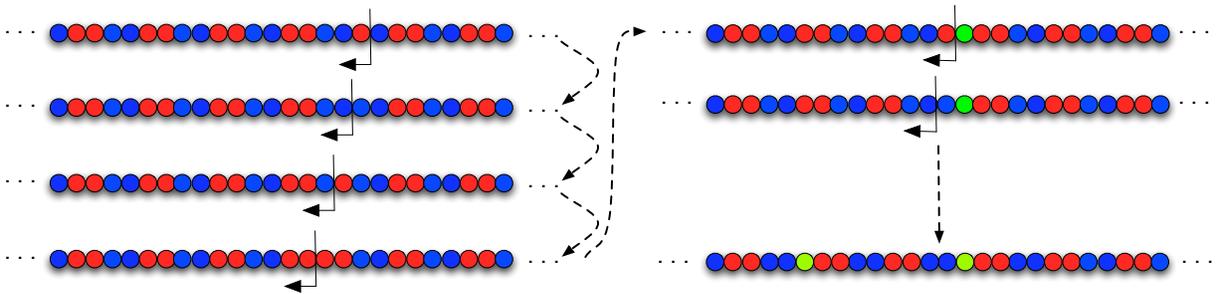
$$Z_i = \dots(1001)^n \circ 10^i \dots$$

These subconfigurations Z_i describe a set of “archetypal local errors” associated with the 1001 minimal repeat unit. As long as $n \geq 9$, so that there is a buffer zone of size 36, such configurations will be locally patchable. Moreover, there are only really 3 classes of patchings: $i = 0, 1, 2$ – all the other situations have patches equal to one of these three forms. Denote the patch for Z_i by P_i .

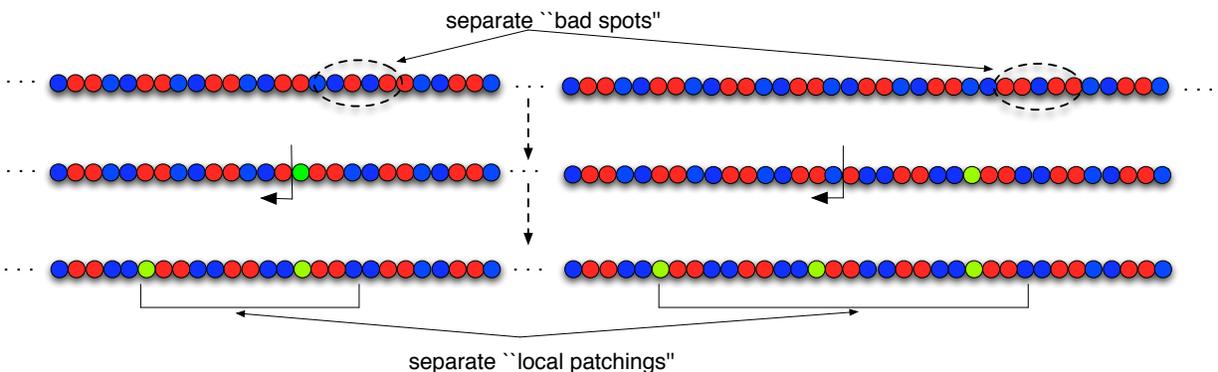
Form all complete configurations composed by concatenating copies of these Z_i ’s, i.e.

$$X = Z_{i_1} \circ Z_{i_2} \circ \dots \circ Z_{i_k}$$

where $i_j \in \{0, 1, 2\}$, $n \geq 9$, and the first and last Z_{i_1} , and Z_{i_k} are considered as left- and right-end configurations respectively. Denote the set of all configurations of this form by \mathcal{Z} , and call these the “almost good” configurations. Now, we want to define dynamic local patching process will drive every almost good configuration $X \in \mathcal{Z}$ into a completely solved state. (Notice that \mathcal{Z} itself defines an LP pattern, so what we want to do is find a rule that transforms one LP pattern to another.) This is very simple to do: we simply “apply” the patches P_i one step at a time:



Because each of the bad spots Z_{i_j} can be patched independently, the dynamic patching process happens independently for each bad spot in the concatenation:



Let's formalize and generalize this construction. Suppose T is a given pattern and $T' \subset T$ is a locally patchable subpattern. Suppose T' is locally checkable by check scheme Θ with radius r , and without loss of generality, let's assume $G(\Theta)$ contains one strongly connected component C . Let $L = lcm(C)$, the least common multiple of the lengths of cycles in C . Now, pick a cycle C in C – it doesn't matter which one. Let C also denote the repeatable sequence of states corresponding to the cycle (for instance the state sequence 1001 corresponds to cycle C_1 in the example above). Now, consider the subconfigurations

$$Z_i = \dots C^{L/|C|+1} \circ C(1:i) \circ C^{L/|C|+1} \dots$$

where $1 \leq i \leq |C|$. Each Z_i has a corresponding ‘‘patch’’ W_i , a subconfiguration of size $2(L + |C|) + i$ of the form $C \circ W'_i \circ C$ which is Θ -admissible. Notice that the concatenation of any two such patchings W_i and W_j is a Θ -admissible configuration of the form $C \circ W' \circ C$.

Let $z_i = |Z_i|$. For $v = 0, \dots, z_i$, let

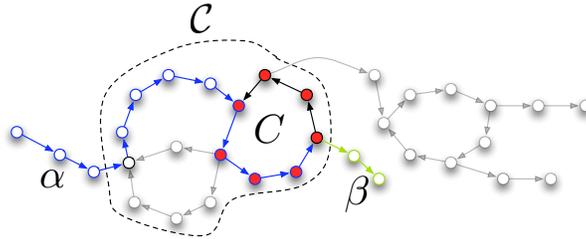
$$W_i^v = Z_i[1 : z_i - v] \circ W_i[z_i - v + 1 : z_i].$$

In words, the sequence

$$W_i^0 \rightarrow W_i^1 \dots \rightarrow W_i^{z_i}$$

is the local trajectory from $Z_i = W_i^0$ to $W_i = W_i^{z_i}$ in which the ‘‘patch is applied one step at a time.’’ Of course, there might be some v such that $W_i^v = W_i^{v+1}$ as defined; in this case, the local trajectory appears to ‘‘skip’’ at least one step. Remove the repeated elements from the sequence, denote the numbering of remaining elements n_i , and renumber these so that $Z_i = W_i^0, \dots, W_i^{n_i} = W_i$. For each $v \in [0, n_i]$, there is a unique position a_i^v in $[1, z_i]$ and state $s_i^v \in S$ such that changing the state of agent a_i^v in W_i^v from whatever it is to s_i^v produces W_i^{v+1} .

Let P be a maximal acyclic path in $G(\Theta)$ intersecting the cycle C such that the only nodes in the strongly connected component of the graph $P \cup C$ are contained C . Evidently $P \cup C$ is single terminus; let γ be the unique gateway node. Let α denote the portion of P up to γ , and β denote the portion after γ . Pictorially, the situation is:



in which the blue path is α , the green path is β , and the nodes shaded red comprise the cycle C . Use the notation α and β for the state subconfigurations corresponding to these paths.

Now, consider the configurations X of the forms

$$X = \alpha \circ C^{l_1} \circ Z_{i_1} \circ C^{l_2} \circ Z_{i_2} \circ \dots \circ C^{l_k} \circ \dots Z_{i_k} \circ C^{l_{k+1}} \beta \quad (4.1)$$

and

$$X = \alpha \circ C^{l_1} \circ W_{i_1}^{v_1} \circ C^{l_2} \circ W_{i_2}^{v_2} \circ \dots \circ C^{l_k} \circ W_{i_k}^{v_k} \circ C^{l_{k+1}} \circ \beta \quad (4.2)$$

where k is any integer, l_i are integers, and in the second form, $v_l \leq n_{i_l}$ for all $l \leq k$. Denote the set of configurations of the form in 4.1 by \mathcal{Z} and those of the form in eq. 4.2 by \mathcal{W} . In words, \mathcal{Z} consists of the ‘‘almost correct’’ configurations composed of repetitions of the cycle C , with ‘‘bad spots’’ interspersed at intervals of at least $2L$ is size. \mathcal{W} then consists of the configurations that lie along trajectories between the ‘‘almost correct configurations’’ in \mathcal{Z} and the completely correct configurations in which the patch W_{i_l} has been applied to bad spot Z_{i_l} .

It is very simple to encode these trajectories with a local rule. Consider the set of radius $R = 2L + |C|$ balls in the configurations in \mathcal{W} , and call this set $\mathcal{B}_{\mathcal{W}}$. Given $B \in \mathcal{B}_{\mathcal{W}}$, the central agent $\star(B)$ will be able to locally determine whether it is in one of the $W_{i_l}^{v_l}$'s – and more particularly, whether it is the unique (within the

$4L + 2|C|$ region around it) agent of the form a_i^v defined above. Because this local determination is possible, we are able to define the local rule:

$$\widehat{F}_1(B) = \begin{cases} s_i^v, & \text{if } B = B_R(a_i^v, W_i^v) \\ B(0), & \text{otherwise} \end{cases}.$$

In words, this rule simply enacts the local patching step if it is the unique agent within the patching area that is supposed to act; otherwise, it does nothing. This rule is evidently the embodiment of “each patch being applied independently.” Under any live timing model \mathcal{S} , \widehat{F}_1 will drive configurations in \mathcal{Z} along a path through \mathcal{W} into T , the locally patchable subpattern of T chosen above. Having originally started out with repeats of the cycle C with interspersed errors, F_1 patches these errors by using other copies of other cycles in the strongly connected component C .

\widehat{F}_1 is thus a solution to T that works on “almost correct” configurations. (Of course, since above definition of \widehat{F}_1 is only made on a subset of all balls of radius R (those in $\mathcal{B}_{\mathcal{W}}$), \widehat{F}_1 is only partially specified.) It’s very simple to compute how long it takes from \widehat{F}_1 to converge to a solution in its region of definition. Suppose $X \in \mathcal{W}$. Then it is of the form

$$X = \alpha \circ \left(\bigcirc_{h=1}^K C^{l_h} \circ W_{i_h}^{v_h} \right) \circ C^{l_{k+1}} \circ \beta,$$

for some K . Hence, for any call sequence s ,

$$(\widehat{F}_1)_s^k(X) = \alpha \circ \left(\bigcirc_{h=1}^K C^{l_h} \circ W_{i_h}^{v_h + s_k(v_h)} \right) \circ C^{l_{k+1}} \circ \beta,$$

where $s_k(v_h)$ is the number of calls to agent v_h by the k -th step of s . In words: in any timing model, during every round (i.e. period in which each agent is called at least once), at least one step of each independent patching is made. Now, within each “unit” $C^{l_h} \circ W_{i_h}^{v_h + k}$, one agent is active when $v_h + s_k(v_h) \leq n_{i_h}$, but once $s_k(v_h) = n_{i_h} - v_h$, this unit is solved and becomes inactive. Thus, in any uniform timing model \mathcal{S} , and $X \in \mathcal{W}$,

$$\frac{|\mathcal{S}_n|}{|X|} \tau_{\mathcal{S}}(|X|, K, \min_h n_{i_h}) \leq TTS(\widehat{F}_1, X) \leq \frac{|\mathcal{S}_n|}{|X|} \tau_{\mathcal{S}}(|X|, K, \min_h n_{i_h})$$

where $|\mathcal{S}_n|$ is the average number of agents in a size n configuration called per timestep in \mathcal{S} , and $\tau_{\mathcal{S}}(n, k, l)$ is the average time it takes for k agents each to be called l times, in a configuration of size n . Now, $\min_h n_{i_h} = 1$ and $\max_h n_{i_h}$ is at most the least common multiple of the cycle lengths in C , which is bounded by a constant $A(m, r)$ independent of system size.

At this point, the answer actually begins to depend somewhat on the timing model (unlike the previous algorithms), so we’ll compute it for three specific cases: 1) the totally synchronous model \mathcal{S}^s , 2) the k -bounded asynchronous model \mathcal{S}_k , and 3) the totally asynchronous model \mathcal{S}^a . In the synchronous model, $|\mathcal{S}^s| = n$, and $\tau_{\mathcal{S}^s}(n, K, l) = l$, so $TTS_{\mathcal{S}^s}(\widehat{F}_1, X) \leq A(m, r)$, i.e. it scales as $O(1)$ with system size. In the k -bounded asynchronous model, $|\mathcal{S}_k| = 1$, and $\tau_{\mathcal{S}_k}(n, K, l) = C \cdot k \cdot n$ where $C \in (0, 1)$. Hence $C \cdot k \leq TTS_{\mathcal{S}_k}(\widehat{F}_1, X) \leq C \cdot k \cdot A(m, r)$, so here too the runtime scales as $O(1)$ with system size. In the totally asynchronous model, $|\mathcal{S}^a| = 1$, and $nH_K - K \leq \tau_{\mathcal{S}^a}(n, K, l) \leq l(nH_K - K)$, where H_K is the k -th harmonic number. Hence for some constants c_1, c_2 , $c_1 \cdot \log(K) \leq TTS_{\mathcal{S}^a}(\widehat{F}_1, X) \leq c_2 \cdot A(m, r) \cdot \log(K)$. Since K can scale linearly with system size, the worst runtime of \widehat{F}_1 in the asynchronous model scales logarithmically with system size.

4.2.2 Getting Almost Correct

Now, we have to extend the definition of \widehat{F}_1 to $\mathcal{B}_R - \mathcal{B}_{\mathcal{W}}$, the set of all R -balls besides those on which \widehat{F}_1 is already defined. The goal is of this part is to force arbitrary initial conditions into \mathcal{Z} , making them “almost correct”. To see how to do this, let’s return to the setup in example 31.

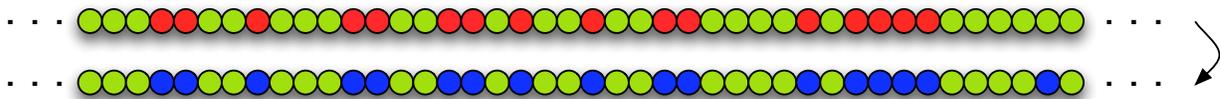
Example 32 Suppose our goal is to drive arbitrary initial conditions into configurations composed of concatenations of 1001s, with local errors of the form 1, 10, 100 interspersed at most once every $2^*36 = 72$ agents. The basic answer to how to do this is very simple: 1) have agents that are part of a region that consists of

repeats of the segment 1001 extend the region by appropriate choice of state, and 2) have agents that aren't already part of any such subconfiguration "seed" their own region starting at the beginning of the segment.

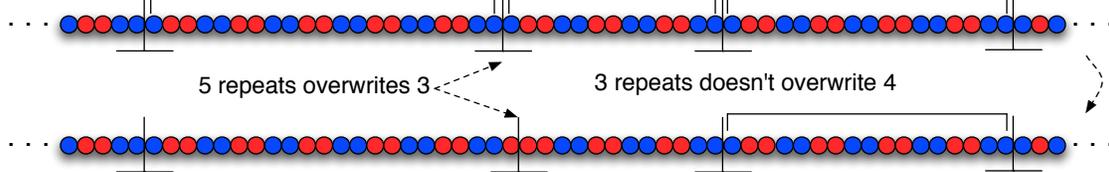
A seeding will be necessary when an agent at position j sees no other agents within a region of size 4 on either side that are in the "first state" of the segment 1001. For example, suppose we start out with a configuration made up entirely 0s and 2s, like



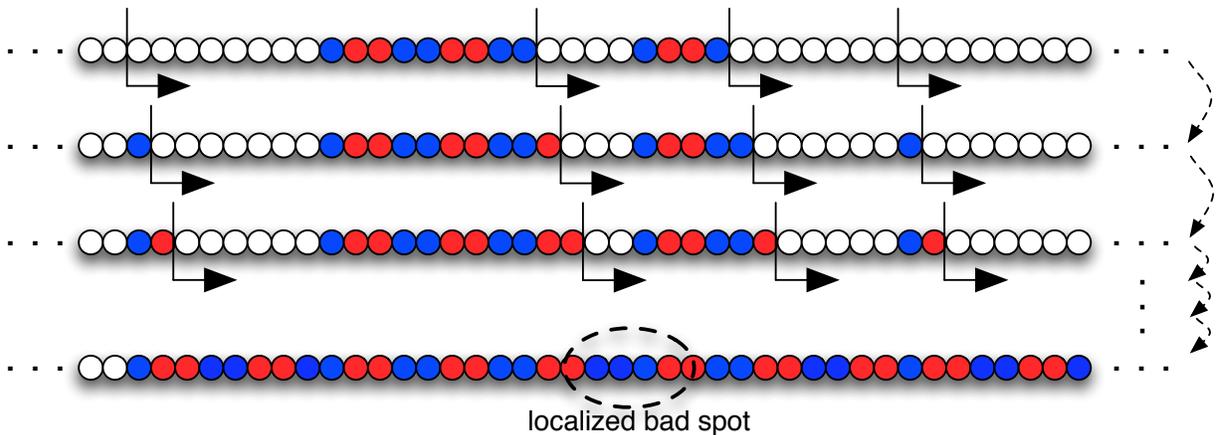
in which red represents 0 and green represents 2. Now, to seed 1001 segments, let's have all agents which do not see a 1 within radius 4 on either side, and whose own state is a minimum state within that radius-4 ball, turn their state to 1:



Now, agents that did not act as seeds will join regions of correctness "seeded" by other agents, so that those regions should grow larger and larger. To ensure growth, the rule will allow a region of size k to overwrite (from the left) a region of size l only when $k \leq l$:



In other words, larger regions "win out" over smaller regions. If $k = l$, we can default to having the left region win the competition and overwrite. Once any such region is larger than $2 \cdot lcm(|C_1|, |C_2|) / |C_1| = 72$, it needn't grow any longer. Eventually, the only remaining "bad spots" will be localized and isolated within a region at this large:



This process eventually drives any initial configuration to one which is consistent with C_1 at every point except at localized "bad parts", i.e. a configuration in \mathcal{Z} .

Let's generalize and formalize this idea, by making the following definitions:

- Given a ball B of radius $R = 2L + |C|$, let

$$L(B) = \max\{K \mid B[\star(B) - K : \star(B) - 1] = C^m \circ C[1 : b]\}$$

where $|C| + b + m|C| = K$ and

$$R'(B) = \max\{J \mid B[\star(B) : \star(B) + J] = C^m \circ C[1 : b]\}$$

where again $|C| + b + m|C| = J$. Let

$$R(B) = \max\{R'(B^{-i}) \mid 0 \leq i \leq |C| - 1\}.$$

$L(B)$ represents the size the growing C -repeat region just bordering on $\star(B)$ from the left. $R(B)$ represents the size of the C -repeat region to which $\star(B)$ belongs.

- If $L(B) > 0$ let $s(B)$ be the unique state such that

$$B[\star(B) - L(B) : \star(B)] = C^m \circ C[1 : b]$$

where $|C| + b + m|C| = L(B) + 1$. In words, if $\star(B)$ is to the left of a growing C -repeat region, $s(B)$ is the state to which $\star(B)$ should switch, to extend that region.

- The initial and terminal paths α and β are both less than L in length. Hence, an agent can determine from its L -ball if its position $pos(B)$ is $\leq |\alpha|$ or $\geq |X| - |\beta|$; and obviously can determine. Moreover, if $pos(B) < |\alpha|$, the agent can determine the value of $pos(B)$ – and therefore determine its unique α -consistent state, namely $\alpha(pos(B))$. Similarly if $pos(B) \geq |X| - |\beta|$, the agent can determine $pos(B) - |X| + |\beta|$, and therefore determine its unique β -consistent state, namely $\beta(pos(B) - |X| + |\beta|)$.
- Choose a numbering function $h : \{1, \dots, m\} \rightarrow S$ so that $h(1) = C(1)$ (where $C(1)$ is the first state in the repeatable segment C). Let $D[B]$ be the boolean function which is TRUE if $B[-|C| : |C|]$ does not contain the state $h(1)$, and if

$$h^{-1}(B(0)) = \min_{-|C| \leq i \leq |C|} h^{-1}(B(i)),$$

that is, if the state of the $\star(B)$ agent in B is a minimum relative to the ordering h , of all states present within a radius of $|C|$ on either side.

Now, define the local rule \widehat{F}_2 with radius $R = 2L + |C|$ by:

```

if  $\omega_1[B] \triangleq (pos(B) < |\alpha|)$  then
  |  $\widehat{F}_2(B) = \alpha(pos(B))$ 
else if  $\omega_2[B] \triangleq (pos(B) > |X| - |\beta|)$  then
  |  $\widehat{F}_2(B) = \beta(pos(B) - |X| + |\beta|)$ 
else if  $\omega_3[B] \triangleq ((L(B) > 0) \wedge (L(B) \geq R(B)) \wedge (R(B) < 2L + |C|))$  then
  |  $\widehat{F}_2(B) = s(B)$ 
else if  $\omega_4[B] \triangleq (L(B) = J(B) = 0) \wedge D[B]$  then
  |  $\widehat{F}_2(B) = C(1)$ 
else
  |  $\widehat{F}_2(B) = B(0)$ 
end

```

In words, what this local rule does is: 1) if $\star(B)$ sees that it is just to the right of a left-terminal segment, it takes on the appropriate state to extend that segment; and similarly for the right. Otherwise, 2) if $\star(B)$ is on the border a growing C -repeat region to its left whose size is as large as the C -repeat region to which it already belongs, but hasn't already reached $2L + 2|C|$ in length, then $\star(B)$ extends the region from the left. Otherwise, 3) if $\star(B)$ is not on the border of, or already part of, a C -repeat, it seeds a new C -repeat region; but it only does this if it is a minimal state-agent relative to the ordering h , within a radius- $|C|$ ball. Otherwise, 4) it does nothing.

Proposition 21 *The local rule \widehat{F}_2 drives arbitrary initial conditions into the set \mathcal{Z} .*

Proof: Suppose that we start with an initial condition W_0 is a configuration of the form $\alpha[1 : j] \circ Y \circ \beta[k : |\beta|]$ where $j < |\alpha|$ and $k > 1$. For any live call sequence, after the first timesteps in which the agents at positions j and $|X| - |\beta| + k$ are called, then due to the applications of predicates ω_1 and ω_2 , the configuration becomes $\alpha[1 : j + 1] \circ Y' \circ \beta[k - 1 : |\beta|]$, where Y' is some configuration of size $|Y| - 2$. Thus we can suppose the initial condition W is of the form

$$W = \alpha \circ X \circ \beta.$$

Let $j(W)$ be the maximal l for which $X[1 : l]$ is a subconfiguration of a configuration in \mathcal{Z} . Then for some $h \geq j(W) - 2L - |C|$, $X[h : j(W)] = C^{2L}C[1 : k]$. Now let $B = B_R(j(W) + 1, W)$. Evidently, $L(B) \geq 2L + k$; on the other hand, $R(B) < 2L + k$, since otherwise the maximality assumption about $j(W)$ would be contradicted. But then ω_3 applies, and after the first call to the agent $j(W)$ in call sequence s , say at time t , the configuration is for the form

$$W_t \triangleq (\widehat{F}')_s^t = \alpha \circ X[1 : j] \circ C(k + 1) \circ Y' \circ \beta,$$

for some configuration Y' of size $|Y| - 1$. But then $j(W_1) = j(W) + 1$. Hence, by induction eventually for some t^* , $j(W_{t^*}) = |X| - |\beta|$ and the configuration must therefore be in \mathcal{Z} . ■

Since \widehat{F}_2 drives configurations to \mathcal{B}_W , and \widehat{F}_1 as defined in the previous section drives configurations from \mathcal{B}_W to T' , we immediately have that the local rule applying \widehat{F}_1 on its domain of definition, and \widehat{F}_2 elsewhere, is a solution to T' . That is,

Corollary 2 Define \widehat{F} by

$$\widehat{F}(B) = \begin{cases} \widehat{F}_1(B), & \text{if } B \in \mathcal{B}_W \\ \widehat{F}_2(B), & \text{otherwise} \end{cases}.$$

Then \widehat{F} is a solution to T' . I will call \widehat{F} the Local Patching Algorithm.

Moreover, for all call sequences s ,

$$TTS(\widehat{F}, X, s) \leq TTS(\widehat{F}_2, X, s) + TTS(\widehat{F}_1, Z, s)$$

where $Z = \lim_n (\widehat{F}_1)_s^n(X) \in \mathcal{Z}$. Since we've already computed $TTS(\widehat{F}_1, Z, s)$ for the three timing models we care about, and found it to be fast, we need only compute $TTS(\widehat{F}_2, X, s)$. Now, notice that the proof of proposition 21 does not make use of the ω_4 predicate – that is, whether or not that predicate at its subsequent action were included, the rule would still drive configurations to \mathcal{Z} . On the other hand, the proof implies a linear upper bound for the scaling of $TTS(\widehat{F}_2, X, s)$, which is not good enough to show that \widehat{F} is fast. \widehat{F}_2 turns out, however, on average to be faster than indicated by the proof of prop. 21, as we will now see (and which involves the ω_4 predicate).

In appendix §C.3, it is shown that:

Proposition 22 The runtime of \widehat{F} scales as

- $O(1)$, in the worst-case, for the k -bounded asynchronous timing model or the synchronous timing model if $|C| = 1$ for some cycle $C \in G(\Theta)$.
- $O(\log(n))$ in the worst and average-case, for the totally asynchronous timing model
- $O(\log(n))$ in the average and $O(n)$ in the worst case for the synchronous timing model if $|C_i| > 1$ for all cycles C_i in $G(\Theta)$.

which establishes prop. 19.

The Trivial Case

There is one corner case in which an even faster algorithm is possible. Define a pattern T to be *trivial* if there is an N such that T contains all configurations of size N or larger, i.e. $\cup_{i \geq N} C_i \subset T$. T is locally checkable: choose $r = \lceil N/2 \rceil$ and let Θ be the check of radius r that accepts all radius r -balls b of size $2r + 1$, and accepts balls of size less than $2r + 1$ iff they correct to configurations in T of size N or less. For each size $n \leq 2r$ for which T contains a solution, choose a unique X_n of size n such that $X \in T$. Now, define a local rule F^* of radius $R = 2r + 1$ by

$$F^*(B) = \begin{cases} B(0), & |B| \geq R \\ X_{|B|}(\star(B)), & \text{otherwise} \end{cases}$$

For all $n \geq N$, F^* “solves” any configuration of size n in zero timesteps, since any such configuration is already solved and on these configurations F^* simply is the identity. Hence $TTS(F^*)(n)$ approaches 0 as $n \rightarrow \infty$.

4.3 Pattern Complexity Classes

At this point is worth stepping back and consolidating what we’ve learned about *classes* of locally checkable patterns. Let \mathbb{L} denote the set of all locally generated patterns over state set S . To understand pattern classes is to understand how to partition \mathbb{L} naturally into subset that share certain generic properties. At a very detailed level, we can partition \mathbb{L} by directly classifying the topology or geometry invariants of the associated graphs. This detailed level of analysis is important, and will be discussed in Chapter 7. However, here I want to focus briefly on much coarser-grained pattern classes, distinguishing patterns by the complexity properties of the local rules that can form them.

The results of §3.3 show that we can separate locally checkable patterns into a natural Exponential/Polynomial hierarchy, based on the runtime of the Naive Backtracking algorithm on the various patterns. That is, we can write

$$\mathbb{L} = \bigcup_{j \in \mathbb{N}} \mathbb{L}_p^j \cup \bigcup_{\substack{\lambda \in \mathbb{R} \\ \lambda > 1}} \mathbb{L}_E^\lambda$$

where \mathbb{L}_p^j and \mathbb{L}_E^λ is the set of locally checkable patterns on which the naive backtracking algorithm has average case runtime scaling as $\sim O(n^j)$ and $\sim O(\lambda^n)$, respectively, for configurations of size n . In light of proposition 17, this classification essentially well-defined, being mostly independent of exactly how the backtracking is done, and thus basically a function of the graph structure. Note that the polynomial part of this hierarchy is evidently discrete, as j can only take on positive integral values, while λ can *a priori* be any real number larger than 1 so the exponential portion potentially has a continuous structure.⁵

The results of this chapter show that the Exponential/Polynomial hierarchy identified in §3.3 collapses into a Linear-or-Below class structure. At this level the structure becomes a little subtle.

The Naive Backtracking rule F_Θ and Less Naive Backtracking rule \tilde{F}_Θ work for all locally checkable patterns, while the Local Patching algorithm \widehat{F} works only for locally patchable patterns. Moreover, the first two local rules share a common strategy: distributed virtual turing machine heads emerge at various places in the multiagent system, and create increasingly large substructures that satisfy the pattern; each time distinct heads collide, only one wins, and its pattern substructure engulfs the other’s; eventually a unique head wins out. The Local Patching algorithm can also be seen as a form of distributed virtual turing machine, but the heads created by \widehat{F} seem to work in a different way. Instead of competing to win control and therefore needing to travel throughout the system, each \tilde{F} -created head is able to remain in roughly localized position, and through a bounded number of negotiations with nearby heads is able to concurrently resolve the pattern. The result of these negotiations can be thought of as a form of averaging, in which local pattern choices that don’t quite mesh are blurred at their boundaries until they do mesh.

The ability to sustain this averaging is an inherent property of the pattern – locally patchable patterns “admit compromise” between concurrently operating local decision makers in a way that other patterns do

⁵The specific possible values of λ that can arise as a function of radius r and state m is an important question in the theory of graph eigenvalues.

not. The distinction between these two classes of patterns – and the corresponding strategies for solving them – may exemplify a more general classification of types of self-organization. On the one hand, we have those problems which require information to travel through the system, becoming amplified and elaborated as it propagates. These problems imply long range correlations and do not tolerate error very easily. On the other hand, we have those problems in which a “cleverly chosen” local interaction rule causes many simultaneous local processes to generate a statistically-defined pattern. These problems do not have long range correlations, and are naturally error-tolerant.

Distinguishing the two types of patterns also raises two specific questions within our model: a) Can we show that the two classes are real complexity classes, with a lower bound on possible solution times separating them? and b) Having found one fast algorithm (\widehat{F}), and then improved it in subclass of problems (to \widehat{F}), we want to know whether there are any other natural classes of patterns “between” the two, or whether we’ve done the best we can. Both of these questions call for demonstrations that the two algorithms developed in this chapter are *optimal* within their respective classes. The next chapter provides this demonstration.

One final pattern class distinction that is worth remembering is the Single-Choice/Multi-Choice difference that arose at the beginning of chapter 3. While this can’t be a complexity class distinction (since both are linear), it is a resource usage distinction: single choice check schemes Θ can be solved with radius $2r(\Theta)$ and no extra states, while multi-choice patterns may require radius $2r(\Theta) + 2$ with extra states, or up to $4r(\Theta)$ to be solved with no extra state (as is shown in §6.2). This difference can be interpreted “self-organized-Turing-theoretically” as the difference between those patterns which require Backtracking (multi-choice) and those which do not (single-choice).

Chapter 5

Lower Bounds and Measure of Global Order

In the previous chapter, we were able to find clever algorithms that completely collapsed the polynomial-and-exponential hierarchy associated with the Naive Backtracking algorithm for locally checkable patterns in one dimension. In all cases, we could achieve linear scaling, and in some cases, logarithmic or constant. We therefore naturally would like to know: have we done the best we can? Or, could we by some further cleverness find faster algorithms for at least some classes of patterns? In other words, are the fast algorithms we described in the previous two sections optimal?

Let's define what we mean by optimality.

Definition 27 [Optimality] Suppose f is a solution to a pattern T in timing model \mathcal{S} . Then f is worst-case (resp. average case) optimal if for all solutions g to T in timing model \mathcal{S} , there is a constant $K(g)$ such that for all n , $TTS_{\mathcal{S}}^{\text{worst}}(f)(n) \leq K(g)TTS_{\mathcal{S}}^{\text{worst}}(g)(n)$ (resp. $TTS_{\mathcal{S}}^{\text{avg}}(f)(n) \leq K(g)TTS_{\mathcal{S}}^{\text{avg}}(g)(n)$).

In words, a local rule is optimal if its runtime is less than that of any other rule, up to a multiplicative constant. It turns out that the two fast algorithms defined in chapter 4 – the Less Naive backtracking rule, and the Local Patching rule – are in fact optimal.

Proposition 23 Suppose T is a nontrivial locally checkable pattern (where triviality is meant as in §4.2.2). Then, in the synchronous, k -bounded asynchronous, and totally asynchronous timing models,

1. If T does not have a locally patchable check scheme, the local rule \tilde{F}_{Θ} defined in §4.1 is A) worst- and B) average-case optimal for any check scheme Θ ; while,
2. If T does have a locally patchable check scheme Θ , the local rule \hat{F}_{Θ} defined in §4.2 is A) worst- and B) average-case optimal.

The import of this proposition is not simply to show that the two algorithms described in the previous chapter are, in their respective domains of definition, optimal. It also implies that the two classes of patterns (locally patchable and not locally patchable) are natural complexity classes – and in fact the *only* two natural complexity classes – of locally solvable one-dimensional patterns.

This chapter is devoted to proving proposition 23. Most of this result – everything but the claim of average-case optimality for non-locally patchable patterns – is very simple. In §5.1, I lay out the elementary arguments that cover all the cases except 1 B). Understanding average-case optimality, however, requires the development of a more sophisticated set of techniques. My basic strategy is:

- First, I define a “measure of order” that will distinguish “random” initial conditions as having a small amount of order, from “highly ordered” final states. This measure is based on fourier analysis of patterns.

- Then, I prove that any local rule can only increase this measure of order slowly. At each time step, the amount of order present in a system can only, on average, increase a small amount. As a result, final patterns that have large amounts of order should take, on average, a long time to solve because the local rules take time to build up that order.
- Finally, I compute the measure of order as a function of the graph structure of the locally checkable pattern, and show this measure precisely distinguishes locally patchable from non-locally patchable patterns.

The technique developed here is a “bigger hammer” than the “nail” it drives in may warrant (the nail being claim 1B) in proposition 23). My ulterior motive in this chapter is to introduce a nontrivial measure of order for its own sake, and also to conceptually indicate how more sophisticated lower bound arguments might in the future be constructed by generalizing the ideas developed here.

5.1 Elementary Arguments

In this section I present proofs of the claims 1A), and 2, in proposition 23.

Claim 1A: Suppose Θ is a local check scheme which for every strongly connected component C in $G(\Theta)$, the greatest common divisor of the lengths of the induced cycles is greater than 1. The goal is to show that linear-time scaling is worst-case optimal for possible solution f to $\Theta(CC)$. Suppose such an f is given, and let C_1 be a cycle in $G(\Theta(\text{fix}(f)))$. Choose y such that

$$\dots C_1^{m(r)} y C_1^{m(r)} \dots$$

is not Θ -consistent and there is no z and $m < m(r)$ such that $|z| = (m(r) - m)|C_1| + |y|$ and $\dots C_1^{m'(r)} z C_1^{m'(r)} \dots$ is Θ -consistent. Such a y can be chosen because Θ does not admit patching. Now, let Y and Z be such that $Y \circ C_1^{m(r)} \dots$ is Θ consistent and $\dots C_1^{m(r)} \circ C_1^{m(r)}$ is Θ -consistent, and let

$$X_n = Y \circ (C_1^{m(r)+n} \circ y \circ C_1^{m(r)+n})^{a_n} Z$$

where $m(r) = \lceil r(f)/|C_1| \rceil$, and a_n is chosen such that $|X_n| = |Y| + |Z| + a_n(2m(r)|C_1| + 2n + |y|)$ is a Θ -admissible size. By the Chinese remainder theorem, a_n can always be chosen to be less than the least common multiple of the lengths of induced cycles in the strongly connected component of C_1 . Thus in all live timing models $TTS(f, X_n) > n|C_1|/(2r(f) + 1)$. Hence

$$TTS^{\text{worst}}(f)(n) > ((n - |Y| - |Z|)/a_n - |y| - 2m(r)|C_1|)/2 \geq k(r)n$$

for some constant $k(r)$.

Claim 2: I'll consider each of the three synchrony models, one at a time. Let's start with the k -bounded asynchronous timing model. Because in this case, the average-case and worst-case scaling of \widehat{F} , is identical, we can simply treat the average case. For any nontrivial local check scheme Θ that admits local patching, we want to show that constant time scaling is average-case optimal. To establish this we simply need to show that when T is nontrivial, for any solution g to any local check scheme Θ for T , there is a constant $K(g, k)$ such that $TTS_{S(k)}^{\text{avg}}(g)(n) > K(g, k)$ for all n . Now, let B be any ball in $\mathcal{B}_{r(\Theta)}$, and let $\mathcal{X}_B = \{X \in \mathcal{C} | B \in B_r(X)\}$. It is clear that \mathcal{X}_B in \mathcal{C} is a full-measure subset of \mathcal{C} . On the other hand, since by assumption in proposition 23, T is nontrivial. This means there is B^* such that $\Theta(B^*) = 0$, and either $|B^*| - \star(B^*) > r$ or $\star(B^*) > r$. Suppose $X \in \mathcal{X}_B$, and that $B_r(j, X) = B^*$. Then any solution g to X must cause some agent i such that $|i - j| \leq 2r$ to change state; hence $TTS(g, X) \geq \tau(k, |X|, 2r + 1)$, where $\tau(k, n, m)$ is the expected number of timesteps in the $S(k)$ timing model before at least one agent out of a given group of m agents is called, in a size- n configuration. It is easy to see that $\tau(k, n, m) \geq kn/4m$ so since \mathcal{X}_B is full measure, $TTS(g)(n) \geq k/(8r(g) + 4)$.

Next let's look into the totally asynchronous model. Again, because the average- and worst-case scaling of \widehat{F} is the same in this case, proving the result for the average case is sufficient. Since $\Theta(\mathcal{C})$ is assumed to be nontrivial, the probability that in any given $2r(\Theta) + 1$ window at least one agent will need to change states so that any configuration containing it will be correct is at least $1/m^{2r+1}$. Hence, in any initial condition X

of size n , the expected number of agents that must change state to achieve a T -consistent configuration is at least

$$\sum_{i=0}^{n/(2r+1)} i \binom{n/2r+1}{i} (1/m^{2r+1})^i (1 - (1/m^{2r+1}))^{n/(2r+1)-i} = \frac{n}{(2r+1)m^{2r+1}}.$$

In the totally asynchronous model, the expected number of timesteps for k distinct agents to be called once is about $Ck \log(n)$ where C is a constant $> 1/2$, so any local will have $TTS^{avg}(f)(n) \geq \frac{1}{2(2r+1)m^{2r+1}} \log(n)$.

Finally, let's look at the synchronous model. There are two cases: a) when there is a cycle $C \in G(\Theta)$ with $|C| = 1$ and b) when there is not. In the case of a), then as long as T is nontrivial, let B^* be as in the previous argument. For any $X \in \mathcal{X}_{B^*}$, X is not already solved; so for any solution g , at least one time-step is required before $g(X)$ can be solved. Hence by definition of the synchronous model, $TTS(g, X) \geq 1$ for all $X \in \mathcal{X}_{B^*}$. In the case of b), the tandem repeat argument encapsulated in eq. C.5 of appendix §4.2 holds for *any* algorithm, not just \widehat{F} . Any algorithm g will have $TTS(g, X) \geq (1/(2r(g) + 1)) \cdot L(X)$, where $L(X)$ is the length of the maximum tandem repeat of a sequence shorter than the shortest cycle in $G(\Theta)$. In the worst case $L(X) \sim |X|$ while on average over all X , $L(X) \sim O(\log(|X|))$, yielding the result.

5.2 Discrete Fourier Analysis of Configurations

Now let's turn our attention to Claim 1 B), that when T has no locally patchable check scheme, the Less Naive Backtracking algorithm \widehat{F}_Θ is average-case optimal.¹ My demonstration of this uses techniques from Fourier analysis. In this section, I will introduce the basis of Fourier analysis as applied to the 1-D configurations in our model.

Suppose that S is a set of states of size m . Let V be the complex vector space of dimension m with orthonormal basis $\epsilon_1, \dots, \epsilon_m$. For $v = \sum v_i \epsilon_i \in V$, let

$$\|v\| = \left(\sum |v_i|^2 \right)^{1/2},$$

where $|\cdot|$ denotes absolute value of a complex number. Consider a configuration X of size N with states in S . We have often been thinking of X as a by treating $X(n)$ as an integer $\{1, \dots, m\}$. For the present application, we now think of X as a function $X: \{1, \dots, N\} \rightarrow V$ to the complex vector space defined by $X(n) = \epsilon_{X(n)}$.

Definition 28 [Invariant Discrete Fourier Transform] Given a function $X: \{1, \dots, N\} \rightarrow V$, let the invariant discrete Fourier Transform of X , denoted by $\mathcal{F}[X]$, be the complex-vector-valued function defined by

$$\mathcal{F}[X](\omega = k/N) = \frac{1}{|X|} \sum_{n=1}^{|X|} \epsilon_{X(n)} e^{2\pi i n k / |X|}$$

where $k \in \{0, \dots, N-1\}$.² The power spectrum of X is the norm-square of \mathcal{F} ; that is,

$$\mathbb{P}[X](\omega = k/N) = \|\mathcal{F}[X](\omega)\|^2.$$

Example 33 For the simple locally checkable repeat pattern T_{1234} ,

$$\mathcal{F}[(1234)^n](\omega) = \begin{cases} \frac{1}{4}(\epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4) & \text{for } \omega = 0 \\ \frac{1}{4}(\epsilon_1 + i\epsilon_2 - \epsilon_3 - i\epsilon_4) & \text{for } \omega = 1/4 \\ \frac{1}{4}(\epsilon_1 - \epsilon_2 + \epsilon_3 - \epsilon_4) & \text{for } \omega = 1/2 \\ \frac{1}{4}(\epsilon_1 - i\epsilon_2 - \epsilon_3 + i\epsilon_4) & \text{for } \omega = 3/4 \\ 0 & \text{otherwise} \end{cases}$$

¹In the rest of this chapter I will assume that we're working with the synchronous update model for notational simplicity, but the arguments go through in the more general case.

²This definition of the Discrete Fourier Transform is slightly different from that used in standard signal processing. The normal transform associates each $s \in S$ with a complex number; then considers each X as a complex-valued function; and so the DFT would be a complex valued function. Here, we associate each s with an element of an orthonormal basis of \mathbb{C}^m ; X is then a complex-vector valued function; and so therefore is \mathcal{F} . The reason this is done is to make it "invariant" under permutations in the values assigned to the states in S .

so that

$$\mathcal{P}[(1234)^n](\omega) = \frac{1}{4} \sum_{i=0}^3 \delta_{i/4}(\omega)$$

in which $\delta_f(\omega)$ is the impulse function that is 1 at $\omega = f$ and 0 elsewhere (see fig. 5.1a). In words, $(1234)^n$ has uniformly-high peaks in its power spectrum at frequencies that are multiples of $1/4$.

Example 34 For the non-locally checkable $1/2$ -way pattern $T_{1/2}$,

$$\mathcal{F}[0^n 1^n](\omega = k/2n) = \frac{\epsilon_0}{2n} \cdot \left(\sum_{t=0}^{n-1} e^{\pi i k t / n} \right) + \frac{\epsilon_1}{2n} \cdot \left(\sum_{t=n}^{2n-1} e^{\pi i k t / n} \right) = \frac{\sin^2(k/2)}{2n} \left(1 + i \cdot \cot\left(\frac{k\pi}{2n}\right) \right) \cdot (\epsilon_0 - \epsilon_1)$$

so that

$$\mathcal{P}[0^n 1^n](\omega = k/2n) = \begin{cases} \frac{1}{n^2} \csc^2\left(\frac{k\pi}{2n}\right), & k \text{ odd} \\ 0, & k \text{ even} \end{cases}$$

for $\omega > 0$. At $\omega = 0$, $\mathcal{P}(0) = 1/2$. Fig. 5.1 b) displays the square-root of this, to make the oscillations more evident.

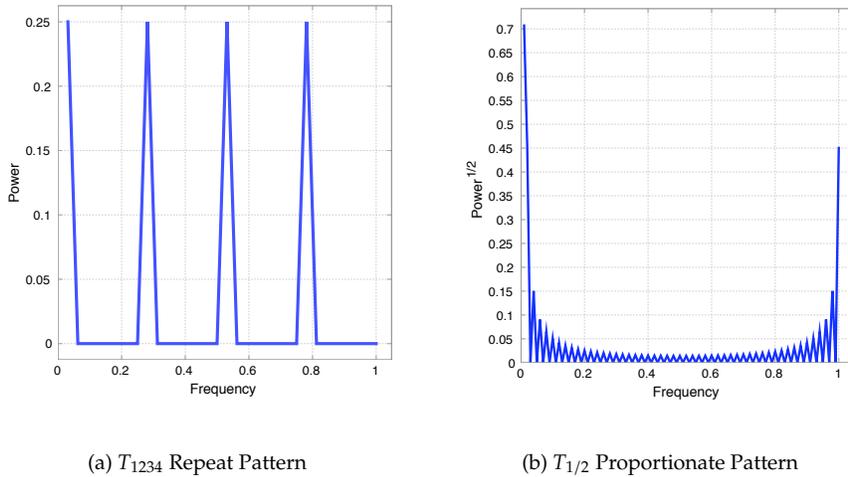


Figure 5.1: A) The power spectrum of configuration $(1234)^n$. B) The square-root of the power spectrum of $0^n 1^n$.

To aid in the computation of the power spectrum, it is useful to introduce another function of X .

Definition 29 [Autocorrelation Function] The autocorrelation function of X , denoted $\sigma[X]$, is the real-valued function of $\tau \in \{0, \dots, N-1\}$ defined by

$$\sigma[X](\tau) = \frac{1}{|X|} \left(\sum_{t=1}^{|X|} (\epsilon_X(t) - \bar{X}) \otimes (\epsilon_X(t + \tau) - \bar{X}) \right)^\dagger$$

where the \dagger superscript indicates contraction of the rank-two tensor with respect to the $\{\epsilon_i \otimes \epsilon_j\}$ basis, the indexing is circular, and $\bar{X} = (1/|X|) \sum_n \epsilon_X(n)$.

Because of the circular indexing and the definition of tensor contraction, a simple calculation shows that

$$\sigma[X](\tau) = \left(\frac{1}{|X|} \sum_t \epsilon_X(t) \otimes \epsilon_X(t + \tau) - \bar{X}^{\otimes 2} \right)^\dagger = \frac{1}{|X|} N_\tau[X] - \frac{1}{|X|^2} \sum_{j=1}^m N_j^2[X]$$

where

$$N_\tau(X) = |\{t | X(t) = X(t + \tau)\}|$$

is the number of agents whose state is equal to the state of the agent τ places to its right, and

$$N_j[X] = \{t | X(t) = j\}$$

is the number of agents in state j .

Example 35 For the T_{1234} patterns, $\sigma[(1234)^n](\tau) = \begin{cases} \frac{3}{4}, & \text{for } \tau \equiv 0(4) \\ -\frac{1}{4}, & \text{otherwise} \end{cases}$. For the T_{10100} repeat pattern,

$$\sigma[(10100)^n](\tau) = \begin{cases} \frac{12}{25}, & \text{for } \tau \equiv 0(5) \\ -\frac{8}{25}, & \text{for } \tau \equiv 1, 4(5) \\ \frac{2}{25}, & \text{for } \tau \equiv 2, 3(5) \end{cases}$$

. These calculations show that the autocorrelation function is periodic when the configuration contains a periodic long-range correlation. On the other hand, $\sigma[0^{n/2}1^{n/2}](\tau) = \frac{1}{2} \left(1 - \frac{4\tau\sigma(\tau)}{n} \right)$, where $\sigma(\tau)$ is 1 when $\tau \leq n/2$ and -1 when $\tau > n/2$. This shows a long-range correlation that falls off with the length of the configuration.

The importance of the autocorrelation for our purposes is its relation to the power spectrum.³

Proposition 24 When $\omega = 0$,

$$\mathbb{P}[X](\omega) = \frac{1}{|X|^2} \sum_j N_j[X]^2.$$

For $0 < \omega < 1$,

$$\mathbb{P}[X](\omega) = \frac{1}{|X|} \sum_{\tau=1}^{|X|} \cos(2\pi\omega\tau) \sigma[X](\tau). \quad (5.1)$$

Proof: The $\omega = 0$ case follows immediately from the definition of the Fourier transform. For $\omega > 0$, given configuration X , let $N_\tau(j)$ denote the number of indices i such that $X(i) = X(i + \tau) = j$. By definition of the power spectrum,

$$\begin{aligned} \|\mathcal{F}[X](\omega)\|^2 &= \frac{1}{|X|^2} \sum_{j=1}^m \left| \sum_{t \in S_j} e^{2\pi i \omega t} \right|^2 \\ &= \frac{1}{|X|^2} \sum_j \left(N_j(X) + \sum_{\substack{t, t' \in S_j(X) \\ t \neq t'}} e^{2\pi i \omega (t - t')} \right). \end{aligned}$$

Symmetrizing gives

$$\|\mathcal{F}[X](\omega)\|^2 = \frac{1}{|X|} + \frac{1}{2|X|^2} \sum_j \sum_{\substack{t, t' \in S_j(X) \\ t \neq t'}} e^{2\pi i \omega (t - t')} + e^{2\pi i \omega (t' - t)}.$$

By definition of the cosine function the RHS becomes

$$\frac{1}{|X|} + \frac{1}{|X|^2} \sum_j \sum_{0 < \tau \leq N-1} \cos(2\pi\omega\tau) N_\tau(j).$$

³Though it may really be the other way around: the autocorrelation is perhaps more fundamental the power spectrum.

Reversing the order of summation and using the fact that $\sum_{t=0}^{N-1} \cos(2\pi k/Nt) = 0$ when $0 < k < N$, we have

$$\begin{aligned} \|\mathcal{F}[X](\omega)\|^2 &= \frac{1}{|X|} \sum_{0 \leq \tau \leq N-1} \cos(2\pi\omega\tau) \frac{1}{|X|} \sum_j N_\tau(j) \\ &= \frac{1}{|X|} \sum_\tau \cos(2\pi\omega\tau) \frac{1}{|X|} \left(\sum_j N_\tau(j) - \frac{1}{|X|^2} \sum_t N_j^2(t) \right) \\ &= \frac{1}{|X|} \sum_\tau \cos(2\pi\omega\tau) \frac{1}{|X|} \left(\sum_t X(t) \otimes X(t+\tau) - \bar{X} \right)^\dagger \end{aligned}$$

and the RHS of the above is $f1|X| \sum_\tau \cos(2\pi\omega\tau) \sigma_X(\tau)$ by definition of σ_X . ■

Sometimes it is convenient to work with the function

$$\sigma'[X](\tau) = \left(\frac{1}{|X|} \sum_t \epsilon_X(t) \otimes \epsilon_X(t+\tau) \right)^\dagger,$$

which I call the *modified* autocorrelation function. For example, the proof of proposition 24 shows that $\mathbb{P}[X](\omega) = \frac{1}{|X|} \sum_\tau \cos(2\pi\omega\tau) \sigma'_X(\tau)$ for all ω , including the $\omega = 0$ case.

Eq. 5.1 is essentially a discrete version of what is known in continuous Fourier analysis as the Wiener-Khinchin theorem, and is useful for conceptual and computational purposes. What it says is that: the power spectrum measures something about the spatial predictive information. It makes it easy to see that

$$\sum_{k=0}^{n-1} \mathbb{P}[X](k) = 1$$

since $\sigma'[X](0) = 1$ for all X .

5.3 \mathcal{P} as a Slowly-Growing Order Measure

The reason that Fourier analysis is useful for proving lower bounds can be seen intuitively by following the progress of the power spectrum at various points along the trajectory of a local rule while it is constructing a pattern.

Consider, for instance, the repeat pattern T_{1234} , and let Θ be its associated radius 1 check scheme. Suppose we pick a random initial condition X_0 over 4 states, with 800 agents, and iterate F_Θ on X_0 until it converges to a solved state. Looking at $\mathcal{P}(F^t(X_0))$ for various t , we see that, to begin with, there are no peaks at non-zero frequencies (fig. 5.2a). As we apply the rule F_Θ to X_0 , the configuration will evolve toward a solved state. As it moves along its trajectory, the power spectrum reflects the growing amount of order present by developing peaks (fig. 5.2b-j). At first, these peaks are low and spread out. As time goes by, the peaks increase in height and decrease in spread. Eventually the power spectrum is identical to that computed in figure 5.1a).

Suppose we now select one specific frequency for which the power spectrum of the final configuration has a non-zero peak – say, $\omega = 1/2$ – and plot the value of $\mathcal{P}(1/2)$ throughout the trajectory:

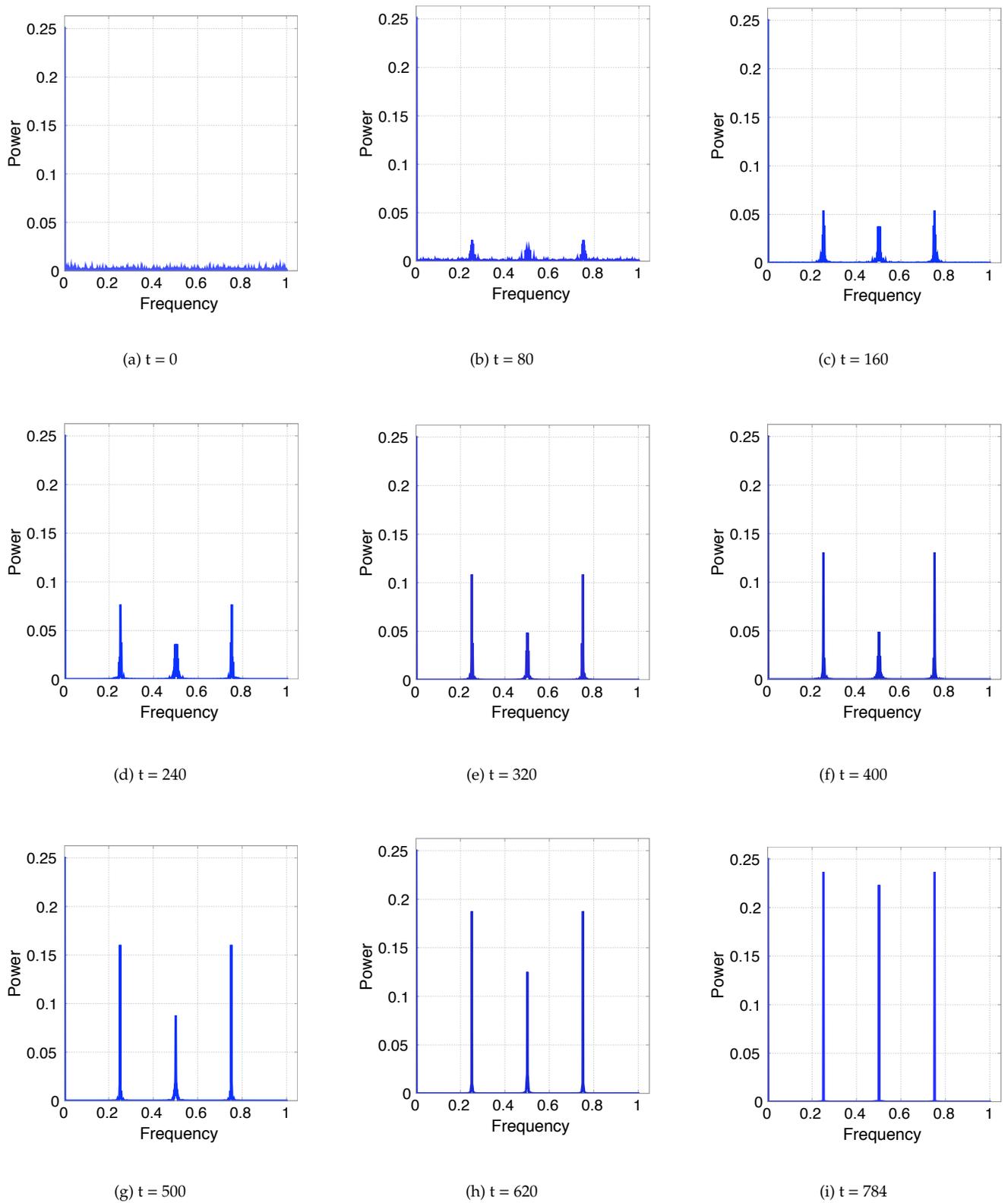
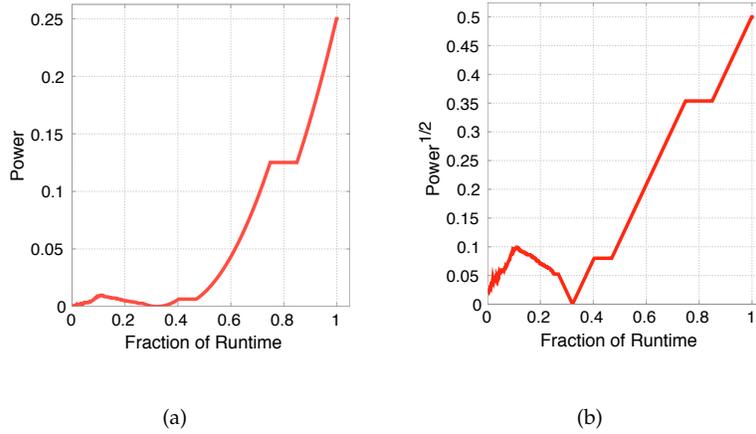


Figure 5.2: The power spectrum along a trajectory from an 800-agent randomly chosen initial condition at $t = 0$ in a), to a solved state of the form $(1234)^{200}$, under the operation of a local rule solution to T_{1234} .



For some period of time it remains low; then, it increases monotonically toward its final value, getting larger relatively slowly and continuously. The square-root of the power spectrum, show to the right in the above figure, in fact appears to (eventually) increase linearly with time.

The key point is that slow, continuous increase of the the power spectrum from 0 to a comparatively large peak in specific frequencies is not specific to our choice of local rule. In fact, as we will show:

1. Random initial conditions typically have no power in non-zero frequencies.
2. But for *any* local rule f , the increase of $\mathcal{P}(\omega)$ during one iteration of f must, on average, *always* be small. In fact, there is a constant $C(r)$ depending only on $r = r(f)$, such that the difference

$$\Delta\mathcal{P} \triangleq \mathcal{P}[f(X)] - \mathcal{P}[X]$$

is, when averaged over all X with size n , no bigger than $C(r)/n$.

3. Hence, if for a selected frequency ω the power spectrum in the final state must be high, a comparatively large number of iterations must have occurred for local rule f to build up the power in that frequency. In fact,

$$TTS^{avg}(f)(n) \propto Kn,$$

where K is the (average) power $\mathcal{P}(\omega)$ of the final pattern in frequency ω .

4. Finally, non-locally patchable patterns always have $K > 0$, so the previous item yields the relevant lower bound.

In the remainder of this section, I will demonstrate items 1 and 2 above. In §5.4, I formulate a proper method for computing the power spectrum over an entire pattern (not just a single configuration), so that the average mentioned in item 3 above makes sense. Finally, in §5.5, I compute the power spectrum of a pattern in terms of the pattern's graph $G(\Theta)$, allowing us to prove item 4 and thus, proposition 23.

5.3.1

As a warm up, we prove that the average configuration typically has very little power in non-zero frequencies. In particular:

Proposition 25 For $\omega > 0$,

$$\langle \mathcal{P}[X](\omega) \rangle_{X \in \mathcal{C}_{n,m}} = \frac{1}{n} \left(1 - \frac{1}{m} \right) = O\left(\frac{1}{n}\right).$$

Proof: We will compute $\langle \sigma'[X](\tau) \rangle_{X \in \mathbb{C}_n}$ and then use prop. 24. Recall that $\sigma'[X] = \frac{1}{n} N_\tau[X]$ where $N_\tau[X]$ is the number of t for which $X(t) = X(t + \tau)$, with circular indexing. Evidently $\sigma'[X](0) = 1$. For $\tau \geq 1$, notice that

$$\text{Prob}(X(1) = X(1 + \tau)) = \frac{1}{m}.$$

Extending this,

$$\text{Prob}(X(2) = X(2 + \tau) | X(1), X(1 + \tau)) = \frac{1}{m}$$

and

$$\text{Prob}(X(3) = X(3 + \tau) | X(1 : 2), X(1 + \tau : 2 + \tau)) = \frac{1}{m},$$

and &c. Hence $\langle N_\tau[X] \rangle_{X \in \mathbb{C}_n} = \frac{n}{m}$, so that for $\tau > 0$, $\langle \sigma'[X] \rangle_{X \in \mathbb{C}_n} = \frac{1}{m}$. Plugging this computation into proposition 24, we have

$$\begin{aligned} \langle \mathcal{P}[X](\omega) \rangle_{X \in \mathbb{C}_n} &= \frac{1}{n} \sum_{\tau=0}^{n-1} \cos(2\pi\omega\tau) \langle \sigma'[X](\tau) \rangle_{X \in \mathbb{C}_{n,m}} \\ &= \frac{1}{mn} \left(\sum_{\tau=1}^{n-1} \cos(2\pi\omega\tau) \right) + \frac{1}{n} \\ &= \frac{1}{mn} \cdot (-1) + \frac{1}{n} = \frac{1}{n} \cdot \frac{m-1}{m} \end{aligned}$$

as claimed. ■

We can think of 25 as computing a bound on $\langle \mathcal{P}[F(X)](\omega) \rangle_{X \in \mathbb{C}_n}$ where F is the identity local rule. Our goal now is to extend prop. 25 to all local rules F . The proof of prop. 25 has two steps: first, we compute $\langle \sigma'[X] \rangle$, and then we plug the result of that computation into prop. 24. It turns out to be possible to compute $\langle \sigma'[F(X)] \rangle$ explicitly (and doing this is handy later on in §5.6). For the present purposes, however, only need:

Proposition 26 *Let F be a radius- r local rule over m states. Then for $n > 2r + 1$ and $\tau, \tau' \in [2r + 2, n - 2r - 1]$,*

$$\langle \sigma'[X](\tau) \rangle_{X \in \mathbb{C}_n} = \langle \sigma'[X](\tau') \rangle_{X \in \mathbb{C}_n}.$$

Proof: Let \mathcal{B}_n denote the set of r -balls arising in configurations in \mathbb{C}_n , and

$$N_{b_1, b_2}^\tau(X) = |\{t | B_r(t, X) = b_1 \text{ and } B_r(t + \tau, X) = b_2\}|.$$

Suppose that $n > 2r + 1$ and $\tau \in [2r + 2, n - 2r - 1]$. Then if both b_1 and b_2 are right- or left-end balls, $N_{b_1, b_2}^\tau(X) = 0$, so $\langle N_{b_1, b_2}^\tau(X) \rangle_{X \in \mathbb{C}_n} = 0$. If b_1 is a end-ball $l < r$ steps from the right- or left-end and b_2 is a central ball – or vice versa – then

$$\langle N_{b_1, b_2}^\tau(X) \rangle_{X \in \mathbb{C}_n} = \frac{1}{m^{r+l+1}} \frac{1}{m^{2r+1}} = \frac{1}{m^{3r+l}}.$$

Finally, if b_1 and b_2 are both central balls, then for all $t \notin [n - r - \tau, n + r - \tau]$ (using circular indexing), the probability $\text{Pr}_\tau(b_1, b_2)$ given by

$$\text{Prob}((B_r(t, X) = b_1) \text{ and } (B_r(t + \tau, X) = b_2)) = \frac{1}{m^{2(2r+1)}},$$

is independent of τ as long as $\tau \in [2r + 2, n - 2r - 1]$. Similarly for A, B of length k , the probability $\text{Pr}_\tau(b_1, b_2 | A, B)$ given by

$$\text{Prob}((B_r(t, X) = b_1), (B_r(t + \tau, X) = b_2) | (X(t - k : t - 1) = b_3), (X(t - k + \tau : t + \tau) = b_4)),$$

is also independent of τ as long as $\tau \in [2r + 2, n - 2r - 1]$ and $t \notin [n - r - \tau, n + r - \tau]$ again. Putting all this together, we have for all b_1, b_2 and $\tau \in [2r + 2, n - 2r - 1]$, that $\langle N_{b_1, b_2}^\tau(X) \rangle_{X \in \mathbb{C}_n} = \langle N_{b_1, b_2}^{\tau'}(X) \rangle_{X \in \mathbb{C}_n}$. Thus letting $\delta(a = b)$

denote the Kronecker delta, we have

$$\begin{aligned} \langle \sigma'[X](\tau) \rangle_{X \in \mathbb{C}_n} &= \sum_{b_1, b_2 \in \mathcal{B}_n} \langle N_{b_1, b_2}^\tau(X) \rangle_{X \in \mathbb{C}_n} \cdot \delta(F(b_1) = F(b_2)) \\ &= \sum_{b_1, b_2 \in \mathcal{B}_n} \langle N_{b_1, b_2}^{\tau'}(X) \rangle_{X \in \mathbb{C}_n} \cdot \delta(F(b_1) = F(b_2)) = \langle \sigma'[X](\tau') \rangle_{X \in \mathbb{C}_n}. \end{aligned}$$

■

To obtain the proper generalization of prop. 25 we combine prop. 26 with prop. 24:

Proposition 27 *Suppose F is a radius r local rule. Then for $\omega > 0$, For any $\omega > 0$,*

$$\langle \mathcal{P}[F(X)](\omega) \rangle_{X \in \mathbb{C}_n} \leq \frac{4(2r+1)}{n}.$$

Proof: The case $n \leq 2r+1$ is trivial, so suppose $n > 2r+1$. Applying prop. 24 gives

$$\langle \|\mathcal{F}[\phi(X)](\omega)\|^2 \rangle_{X \in \mathbb{C}_n} = \frac{1}{n} \sum_{\tau=0}^{n-1} \cos(2\pi\omega\tau) \langle \sigma[\phi(X)](\tau) \rangle_{X \in \mathbb{C}_n}.$$

Now, split this sum into two parts, one for

$$\tau \in A \triangleq [0, 2r] \cup [n-2r-1, n-1]$$

and for

$$\tau \in B \triangleq [2r+1, n-2r-2].$$

Applying the Cauchy-Schwarz inequality to the first sum and prop. 26 to the second sum gives

$$\begin{aligned} \langle \|\mathcal{F}[F(X)](\omega)\|^2 \rangle_{X \in \mathbb{C}_n} &\leq \frac{1}{n} \sqrt{\left(\sum_{\tau \in A} \cos^2(2\pi\omega\tau) \right)} \sqrt{\left(\sum_{\tau \in A} \langle \sigma'[F(X)](\tau) \rangle_{X \in \mathbb{C}_n}^2 \right)} \\ &\quad + \frac{1}{n} \cdot \langle \sigma'[X](2r+2) \rangle_{X \in \mathbb{C}_n} \cdot \sum_{\tau \in B} \cos(2\pi\omega\tau). \end{aligned} \tag{5.2}$$

Now,

$$\sum_{\tau \in A} \cos^2(2\pi\omega\tau) = \alpha(n, r, \omega) \triangleq (2r+1) \left(1 + \frac{\sin(2\pi\omega(4r+1)) + \sin(2\pi\omega(4r+3))}{(8r+4)\sin(2\pi\omega)} \right),$$

and

$$\sum_{\tau \in B} \cos(2\pi\omega\tau) = \beta(n, r, \omega) \triangleq - \left(1 + \frac{\sin(\pi(1+4r)\omega)}{\sin(\pi\omega)} \right).$$

It is easy to see that $|\alpha(n, r, \omega)|, |\beta(n, r, \omega)| \leq 2(2r+1)$. Plugging all this into eq. 5.2 and using $\sigma'[X](\tau) \in [0, 1]$ gives

$$\begin{aligned} \langle \|\mathcal{F}[\phi(X)](\omega)\|^2 \rangle_{X \in \mathbb{C}_n} &\leq \frac{1}{n} \sqrt{\alpha(n, r, \omega)} \cdot \sqrt{2(2r+1)} + \frac{1}{n} |\beta(n, r, \omega)| \\ &\leq \frac{1}{n} \sqrt{4(2r+1)^2} + \frac{2(2r+1)}{n} \\ &\leq \frac{4(2r+1)}{n}. \end{aligned}$$

■

For any X and rule f of radius r , it follows immediately from the definition that

$$\langle \|\mathcal{F}[f^k(X)](\omega)\|^2 \rangle_{X \in \mathbb{C}_n} \leq 4 \frac{2(2r+1)k+1}{n} \tag{5.3}$$

since f^k can be treated as a radius $(2r + 1)k$ rule. Denoting the LHS of eq. 5.3 by $P_n^k(\omega)$ we have

$$\Delta\mathcal{P}(\omega) = P_n^{k+1}(\omega) - P_n^k(\omega) \leq \frac{8(2r + 1)}{n}$$

establishing the "slowing growing" property as described previously.

Denote $T_n = \langle TTS(f, X) \rangle_{X \in \mathbb{C}_n}$. Fix δ a positive integer and let $A = \{X | TTS(f, X) \leq \delta \cdot T_{|X|}\}$. In words, A is the set of configurations on which the time-to-solution of f is at worst δ time the average for configurations of equal size. Let $A_n = \mathbb{C}_n \cap A$. Let $\mu_n(A) = |A_n|/|\mathbb{C}_n| = |A_n|/m^n$, the measure of A relative to the number of configurations of that size. Evidently

$$T_n = \mu_n(A) \cdot \langle TTS(f, X) \rangle_{X \in A_n} + (1 - \mu_n(A)) \cdot \langle TTS(f, X) \rangle_{X \in \mathbb{C}_n - A_n}.$$

By definition $TTS(f, X) > \delta \cdot T_n$ for all $X \in \mathbb{C} - A_n$, and obviously $\langle TTS(f, X) \rangle_{X \in A_n} > 0$. Hence, $T_n \geq (1 - \mu_n(A)) \cdot \delta \cdot T_n$, whence $\mu(A_n) \geq 1 - \frac{1}{\delta}$ for all n .

Next, let's denote $\mathcal{G}_n = \langle \|\mathcal{F}[f^{TTS(f, X)}(X)](\omega)\|^2 \rangle_{X \in \mathbb{C}_n}$. Let $\epsilon \in (0, 1)$ and let

$$B = \{X | \|\mathcal{F}[f^{TTS(f, X)}(X)](\omega)\|^2 \geq \epsilon \cdot \mathcal{G}_n\}.$$

Analogously to above, let $B_n = \mathbb{C}_n \cap B$ and $\mu_n(B) = |B_n|/|\mathbb{C}_n| = |B_n|/m^n$. Evidently

$$\mathcal{G}_n = \mu_n(B) \cdot \langle \|\mathcal{F}[f^{TTS(f, X)}(X)](\omega)\|^2 \rangle_{X \in B_n} + (1 - \mu_n(B)) \cdot \langle \|\mathcal{F}[f^{TTS(f, X)}(X)](\omega)\|^2 \rangle_{X \in \mathbb{C}_n - B_n}.$$

By definition $\|\mathcal{F}[f^{TTS(f, X)}(X)](\omega)\|^2 < \epsilon \cdot \mathcal{G}_n$ for all $X \in \mathbb{C} - B_n$, and since $\|\mathcal{F}[f^{TTS(f, X)}(X)](\omega)\|^2 \leq 1$ for $X \in B_n$. This yields

$$\mu(B_n) \geq \frac{(1 - \epsilon)\mathcal{G}_n}{1 - \epsilon\mathcal{G}_n}$$

for all n .

Let $C = A \cap B$. For any non-negative real-valued function H of configurations X , assuming $\mu(\mathbb{C}_n) > 0$, we obviously have

$$\langle H(X) \rangle_{X \in \mathbb{C}_n} \leq \frac{1}{\mu(\mathbb{C}_n)} \langle H(X) \rangle_{X \in \mathbb{C}_n}. \quad (5.4)$$

Moreover, $\mu_n(C) \geq \max(0, \mu(A_n \cup B_n) - 1)$, so choosing $f \geq (2(1 - \epsilon\mathcal{G}_n))/((1 - \epsilon)\mathcal{G}_n)$ yields

$$\mu_n(C) \geq \frac{1}{2} \mu_n(B) = \frac{(1 - \epsilon)\mathcal{G}_n}{2(1 - \epsilon\mathcal{G}_n)} \quad (5.5)$$

which is positive when \mathcal{G}_n is. We thus have

$$\begin{aligned} \mathcal{G}_n &\leq \frac{1}{\epsilon} \langle \|\mathcal{F}[f^{TTS(f, X)}(X)](\omega)\|^2 \rangle_{X \in \mathbb{C}_n} \\ &\leq \frac{1}{\epsilon} \langle \|\mathcal{F}[f^{\delta T_n}(X)](\omega)\|^2 \rangle_{X \in \mathbb{C}_n} \\ &\leq \frac{1}{\epsilon \mu(\mathbb{C}_n)} P_n^{\delta T_n}(\omega) \\ &\leq \frac{1}{\epsilon \mu(\mathbb{C}_n)} \frac{4}{n} (2((2r + 1)\delta \cdot T_n) + 1) \\ &\leq \frac{32(2r + 1)(1 - \epsilon\mathcal{G}_n)^2 T_n}{\epsilon(1 - \epsilon)^2 \mathcal{G}_n^2} + \frac{K}{n} \end{aligned} \quad (5.6)$$

where K is a constant smaller than $32(2r + 1)$. The first inequality is due to the definition of B_n , the second to the definition of A_n , the third to eq. 5.4 and the fact that P_n^k is a positive real-valued function, the fourth to eq. 5.3, and the fifth to eq. 5.5. Rearranging and maximizing with respect to ϵ ,⁴ we have that for $n > 32(2r + 1)$,

⁴This maximization can be done by choosing $\epsilon = \frac{3 - \mathcal{G}_n - \sqrt{9 - 10\mathcal{G}_n + \mathcal{G}_n^2}}{2\mathcal{G}_n}$.

$$\langle TTS(f, X) \rangle_{X \in \mathbb{C}_n} \geq \frac{\mathcal{G}_n^3(\omega)}{864(2r+1)} \cdot n \quad (5.7)$$

Now, notice that

$$\mathcal{G}_n(\omega) = \langle \|\mathcal{F}[f^{TTS(f,X)}(X)](\omega)\|^2 \rangle_{X \in \mathbb{C}_n} = \sum_{Y \in \mathbb{C}_n} \|\mathcal{F}[Y](\omega)\|^2 \rho_f(Y)$$

where $\rho_f(Y)$ is the proportion of initial conditions in \mathbb{C}_n which converge to Y (that is, ρ_f is the distribution on \mathbb{C}_n induced by f). Hence, $\mathcal{G}_n(\omega)$ is the expected value of $\mathcal{P}[Y](\omega)$, over \mathbb{C}_n , taken with respect to the distribution ρ_f . In words, eq. 5.7 shows that the runtime of any local rule f is bounded below by the system size, times a constant that is:

- directly proportional to the cube of the power spectrum at any fixed frequency ω , averaged against the distribution that f induces over final states, and
- inversely proportional to the radius of information of f .

This holds for each frequency ω , separately. Now, some local rules f might have $G_n(\omega) = 0$ for some (or all) non-zero frequencies, so this bound may not always be non-trivial. The goal of sections 5.4 and 5.5 is to show that for all non-locally patchable patterns, there are some frequencies ω for which the average of $\mathcal{G}_n(\omega)$ over all the admissible sizes of the pattern must be nonzero.

5.4 A More Robust Fourier Transform

The computation culminating in eq. 5.7 applies to configurations of a fixed size, yielding a connection between the average runtime of a local rule on configurations of given size and the value of the power spectrum (at any given frequency) averaged over the same set. However, to complete our demonstration of prop. 23, we have to be able to extend this connection to averages over the whole of a pattern, which will include configurations of various sizes. We will need to combine the spectral densities of all the elements of a pattern, producing the an “average spectral density” function $\bar{\mathbb{P}}[T]$. $\bar{\mathbb{P}}[T]$ should have a peak at frequency ω if some non-negligible fraction of the elements of the pattern T have a peak at that frequency, and the height of the peak should be related to the average heights of the constituent elements, weighted relative to their frequency in the pattern as a whole.

Example 36 Consider the pattern

$$T = T_{10} \cup T_{100} = \{(10)^n, (100)^n \mid n \in \mathbb{N}\},$$

the union of two repeat patterns. Fixing N and taking configurations of size N or less, T contains approximately $N/3$ configurations of the form $(100)^n$ for $n < N/3$, and approximately $N/2$ configurations of the form $(10)^n$ for $n < N/2$. Thus, elements of the size- $3n$ configurations comprise about 40% of the total pattern, and elements of the size- $2n$ configurations about 60%, so according to the informal definition of $\bar{\mathbb{P}}[T]$ given in the previous paragraph, we should have

$$\begin{aligned} \bar{\mathbb{P}}[T] &= \frac{2}{5} \left(\frac{5}{9} \delta_0 + \frac{2}{9} \delta_{1/3} + \frac{2}{9} \delta_{2/3} \right) + \frac{3}{5} \left(\frac{1}{2} (\delta_0 + \delta_{1/2}) \right) \\ &= \frac{47}{90} \delta_0 + \frac{4}{45} \delta_{1/3} + \frac{3}{10} \delta_{1/2} + \frac{4}{45} \delta_{2/3}. \end{aligned} \quad (5.8)$$

In words, $\bar{\mathbb{P}}[T](\omega)$ has peaks at $\omega = 0, 1/3, 1/2$, and $2/3$ – and the non-zero frequencies of T_{10} and T_{100} ; and the weighting reflects the fact that the 10 repeats occur somewhat more frequently.

But wait. There’s an issue being swept under the carpet here, viz.: What is the domain of definition of $\bar{\mathbb{P}}[T]$? As given in def. 28, the discrete Fourier transform for size- n configurations only takes on values for frequencies $\omega = k/n$. Thus the domain of definition of \mathcal{F} (and therefore \mathbb{P}) is different for each size. The two

terms in the first equality in expression 5.8 are actually defined for different sets of frequencies ω , the first term for $k/3n$ frequencies and the second for $k/2n$ frequencies. By combining them, we are tacitly assigning zero power to frequencies for which each is not defined. This means that $\bar{P}[T]$ as written will be defined by frequencies that are *either* of the form $k/2n$ or the form $k/3n$. Thinking about this issue more generally, we realize that as we average spectra of configurations of divers sizes, we will be implicitly increasing the fine-ness of resolution of the definition of the combined spectrum. Continuing this process to configurations of unbounded size, every rational number will eventually belong to the domain of \bar{P} .

This suggests $\bar{P}[T]$ should therefore be defined in the first place to have domain of definition containing *all* frequencies $\omega \in [0, 1)$, i.e. $\bar{P}[T]$ will be function on a *continuous* space, or at any rate all of $\mathbb{Q} \cap [0, 1]$. To formalize this, one might be tempted to define $\bar{P}[T](\omega)$ for any ω by taking the limit of the (average of) $\mathbb{P}[X](\omega_n)$ as ω_n approaches ω ; or to take the limit of linear interpolations. For example, for each ω define $\eta(n, \omega)$ to be that i which minimizes $|(i/n) - \omega|$, and then let

$$\bar{P}[T](\omega) = \lim_{n \rightarrow \infty} \frac{1}{|T_{i \leq n}|} \sum_{x \in T_{i \leq n}} \mathbb{P}[X](\eta(|X|, \omega)).$$

If T contains infinitely many elements, then $\eta(n, \omega)$ will converge to ω , so one might expect this to provide a good limiting definition.

However, this definition has a subtle problem. Taking $T = \{(123)^n | n \in \mathbb{N}\}$ under this definition gives $\bar{P}[T] = \frac{1}{3}\delta_0 + \frac{1}{3}\delta_{1/3} + \frac{1}{3}\delta_{2/3}$; that is, the aggregate simply is the original, reflecting the fact that all instances of the pattern have the same underlying repeated segment of length 3, and a natural harmonic at each integral multiple of the fundamental frequency. Each of these frequencies has the same strength. Now consider the closely related pattern T' made by adding a '1' to the end of each instance of T , i.e. $T' = \{(123)^n 1 | n \in \mathbb{N}\}$. In this case, simple computation shows that for all instances X of this pattern, while $\mathbb{P}[T'](0) \rightarrow 1/3$ as $n \rightarrow \infty$, $\mathbb{P}[X](\eta(n, 1/3))$ and $\mathbb{P}[X](\eta(n, 2/3))$ are strictly less than $1/3$. In fact, they are bounded below .23 for all n , so that in the limit definition, while $\bar{P}[T'](0) = 1/3$, $\bar{P}[T'](1/3) = \bar{P}[T'](2/3) < .23$. Somehow, the addition of something that intuitively shouldn't make a difference (because its relative size gets smaller as $n \rightarrow \infty$) nonetheless does – some of the power that “should” have been assigned to the $1/3$ and $2/3$ frequencies is missing. Compare figures 5.3a) and 5.3b).

The main underlying problem here is that the “real” fundamental frequency of T' (which is $1/3$) is not an integral multiple of $1/|X|$ for any instance X , and if we take different sequences of points converging to $1/3$ besides $\eta(n, 1/3)$, we get different results in the limit. Another way to phrase this is: *the DFT is not “robust” under finite perturbations of infinite patterns*. But it really needs to be if we're going to take aggregates of the spectral density over such patterns. We somehow have to modify the definition of the DFT, making a more robust transform $\tilde{P}[X]$. This transform will still just apply to individual configurations X , but will be invariant enough under small changes in configuration size so that it can support averaging over configurations of many sizes.

I can see two paths toward this end. One rests on the following observation: even though the modified pattern T' above has less power at frequency $1/3$ than it ought, the power spectrum does look essentially correct: there are peaks at $1/3$ and $2/3$, but a bit blurred in comparison to the “crisp” peaks in the spectrum of T . It is easy to see that the spread of the peak scales as a constant number of terms independent of the size of the configuration. Moreover, the area under the peak, given by the sum of $\mathbb{P}[X](\eta(|X|, \omega) \pm k)$ for some small number of terms at distances k or less, DOES approach $1/3$, as one increases k , as long as it's small enough not to start getting contributions from the $2/3$ peak. This suggests we could get at the “true” value of the power spectrum at frequencies ω that are not integral multiples of $1/|X|$ by convolving $\mathbb{P}[X]$ with a delta function at ω . The choice of delta-function would need to be made carefully, since if the support converges too quickly it might “miss” some of the mass under the peak, while if it converges too slowly, it capture too much from surrounding peaks.

Another approach is simply to correct for the issue of needing to get non-integral frequencies ω by taking a subsequence for which ω is an integral frequency, i.e. the first or last $3n$ places in each $(123)^n 1$. To compute the value of $\mathbb{P}[X]$ at $\omega = p/q$, written in lowest terms, one would compute $\mathbb{P}[X'](\omega)$ for substrings X' whose sizes are divisible by q . But which subsequence X' should be taken? One obvious choice is simply to

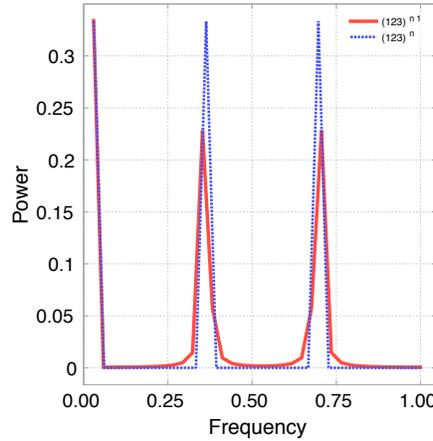


Figure 5.3: Superposition of the power spectra of the patterns $(123)^n$ (in blue) and $(123)^{n,1}$ (in red). The latter has blurred peaks around frequencies $1/3$ and $2/3$, with maximum height less than $.23$, in comparison to the sharp peaks in the former with height $1/3$. However, the area under the red curve local to each peak sums up to $1/3$.

average all possible such X' . In equations, this is:

$$\tilde{\mathbb{P}}[X](p/q) \triangleq \frac{2}{q \lfloor N/q \rfloor (1 + \lfloor N/q \rfloor)} \sum_{m=1}^{\lfloor N/q \rfloor} \sum_{s=0}^{N-mq-1} \mathbb{P}[X(s : s + mq - 1)](pm) \quad (5.9)$$

where $N = |X|$.

In appendix §D, I show that:

Proposition 28 *Let T be any infinite pattern and y, z be any finite configurations, and define $T' = z \circ T \circ y = \{z \circ X \circ y \mid X \in T\}$. Then $\tilde{\mathbb{P}}[T'] = \tilde{\mathbb{P}}[T]$. In words, the new definition of $\tilde{\mathbb{P}}$ is in fact more “robust” than the original, in that its value is not affected by appending finite portions at the ends of patterns.*

and that the “integral subsequence” approach reflected in eq. 5.9 is actually *equivalent* to the delta-function approach described previously. That is, I prove that $\tilde{\mathbb{P}}[X]$ is, essentially, a smoothing of $\mathbb{P}[X]$ obtained by convolving it with a delta function that is independent of X :

Proposition 29 *There is a discrete delta-like function $f_{N,\omega}(a, b)$, independent of X , such that for all ω on which $\tilde{\mathbb{P}}[X](\omega)$ is defined,*

$$\tilde{\mathbb{P}}[X](\omega) = \sum_{l,k} (\mathcal{F}_k[X] \overline{\mathcal{F}[X]}_l + \mathcal{F}_l \overline{\mathcal{F}[X]}_k) f_{N,\omega}(l/N, k/N).$$

This delta-like function has very specific asymptotics that allow it to avoid capturing nearby peaks but still be wide enough to restore the power blurred out of the peaks due to non-integrality, and is useful for computational purposes.

Having defined $\tilde{\mathbb{P}}$, we now need to find an analogous version of eq. 5.7 that applies to $\tilde{\mathbb{P}}$. Denote

$$\tilde{\mathcal{G}}_n = \langle \tilde{\mathbb{P}}[f^{TTS(f,X)}(X)] \rangle_{X \in \mathcal{C}_n}.$$

Now, with the sets A, B , and C defined analogously to the computation in §5.3.1, let

$$K_n = 8(2r + 1)\delta \cdot T_n + 4.$$

Then we have for $\omega = p/q$ in lowest terms,

$$\begin{aligned}
\widetilde{\mathcal{G}}_n(p/q) &\leq \frac{1}{\epsilon} \sum_{m=1}^{\lfloor n/q \rfloor} \sum_{s=0}^{n-mq-1} \frac{2}{q \lfloor n/q \rfloor (1 + \lfloor n/q \rfloor)} \langle \mathbb{P}[f^{\delta \cdot T_n}[X](s : s + mq - 1)](pm) \rangle_{X \in C_n} \\
&\leq \frac{1}{\epsilon \mu(C_n)} \sum_{m=1}^{\lfloor n/q \rfloor} \sum_{s=0}^{n-mq-1} \frac{2q}{n(1+n)} \min\left(\frac{4}{qs} (2((2r+1)\delta T_n + 1)), 1\right) \\
&\leq \frac{q}{\epsilon \mu(C_n)} \left(\sum_{s=0}^{K_n} \frac{2(n-qs+1)}{n(1+n)} + \sum_{s=K_n+1}^{n/q} \frac{2(n-qs+1)}{n(1+n)} \frac{K_n}{qs} \right) \\
&\leq \frac{1}{\epsilon \mu(C_n)} \left(\left(\frac{K_n}{n}\right)^2 + 2 \frac{K_n}{n} \log\left(\frac{n}{K_n}\right) \right) \\
&\leq \frac{2}{\epsilon \mu(C_n)} \sqrt{\frac{K_n}{n}}
\end{aligned} \tag{5.10}$$

where in the fourth inequality we use simple facts about the asymptotics of the harmonic numbers. Rearranging, and optimizing for choice of ϵ^5 , we find for every frequency $\omega > 0$,

$$\langle TTS(f, X) \rangle_{X \in C_n} \geq C \cdot \frac{\widetilde{C}_n^5(\omega)}{2r+1} \cdot n \tag{5.11}$$

where C is a constant larger than .0002. Surely a better constant could be found by more careful accounting throughout, but for our purposes this result is strong enough.

5.5 Graph-Based Computation of \overline{P}

$\widetilde{\mathbb{P}}$ extends immediately from individual configurations on whole patterns by simple averaging. For a configuration X of size n , $\widetilde{P}[X]$ is defined for $\omega = p/q$, where p/q is a rational number in lowest terms with $q \leq n$. For any $\omega \in (0, 1)$ let $\eta'(n, \omega)$ be the closest such p/q to ω , and define

Definition 30 *Define*

$$\overline{\mathbb{P}}_n[T](\omega) = \frac{1}{|T_n|} \sum_{x \in T_n} \widetilde{P}[X](\eta(n, \omega))$$

and

$$\overline{P}[T](\omega) = \lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{|T_k|}{|T_{k \leq n}|} \overline{\mathbb{P}}_k[X](\omega).$$

$\overline{\mathbb{P}}[T]$ is called the spectrum of pattern T .

Propositions 28 and 29 make it easy to compute $\overline{\mathbb{P}}[T]$ – or at least the locations of the non-zero peaks – when T is a locally checkable pattern.⁶ Moreover, this computation is naturally structured in terms of the graph structure of the local check scheme $G(\Theta)$.

- First, consider the patterns $T_1^m = \{(1234 \dots m)^n \mid n \in \mathbb{N}\}$, a repeat pattern with m distinct states. Computing according to props. 28 and 29 we find that:

$$\overline{P}[T_1^m] = \frac{1}{m} (\delta_0 + \delta_{1/m} + \dots + \delta_{(m-1)/m}),$$

⁵Which is done by taking $\epsilon = \frac{5 - \widetilde{C}_n - \sqrt{25 - 26\widetilde{C}_n + \widetilde{C}_n^2}}{4\widetilde{C}_n}$.

⁶It is also possible to compute $\overline{P}[T]$ for a large range of non-locally checkable patterns. It is shown in an appendix to this chapter that in the half proportion case $T_{1/2} = \{0^n 1^n \mid n \in \mathbb{N}\}$, $\overline{P}[T_{1/2}] = \left(\frac{3}{2} - \log(2)\right) \delta_0$.

with equal peaks at each frequency that is a multiple of $1/m$. In the case, the graph associated with the radius-1 check scheme Θ for T_m contains a single m -cycle c . Hence, $\bar{P}[T_m]$ is measuring the “fundamental” frequency $1/m$, and all the higher harmonics, resonant with the cycle c .

- Now consider $T_2 = \{(1001)^n \mid n \in \mathbb{N}\}$, the length-4 repeat pattern with two distinct states. In this case

$$\bar{P}[T_2] = \frac{1}{2}\delta_0 + \frac{1}{4}\delta_{1/4} + \frac{1}{4}\delta_{3/4}.$$

The graph $G(\Theta)$ is a 4-cycle, and $\bar{P}[T_2]$ picks out the “fundamental frequency” of the cycle at $\omega = 1/2$, but due the specifics of the labeling the frequency $1/2$ peak is cancelled out.

- For the generic repeat pattern T_q with minimal segment q , we have

$$\bar{P}[T_q] = \sum_{i=0}^{|q|-1} \alpha_i \delta_{i/|q|},$$

where the $\alpha_i \in [0, 1)$. The coefficient $\alpha_0 = \frac{1}{|q|^2} \sum_{j=1}^m N_j^2(q)$, where $N_j(x)$ is the number of states in x with state j . Hence $\alpha_0 \geq \frac{1}{m^2}$. The coefficient

$$\alpha_1 = \frac{1}{|q|^2} \sum_{j=1}^m \left| \sum_{t \in S_j(q)} e^{2\pi i t / q} \right|^2,$$

where $S_j(x)$ is the set of agents t with state j . α_1 must therefore be positive, since $\alpha_1 = 0$ would imply that for all j , $\sum_{t \in S_j(q)} e^{2\pi i t / q} = 0$, which would contradict the minimality of q . In fact, it is easy to see that $\alpha_1 > \frac{1}{(|q|/2)^{2|q|}}$.⁷

- Generally, suppose Θ is a local check scheme of radius r over m states whose graph $G(\Theta)$ contains a single cycle C . Then $\bar{P}[\Theta(C)]$ has peaks of height α_i at frequencies of the form $i/|C|$ for $i = 0, \dots, |C| - 1$. Since $|C| \leq m^{2r+1}$, $\alpha_1 \geq \Gamma_{m,r} \triangleq \left(\frac{1}{m^{2r+1/2}}\right)^{2m^{2r+1}}$. Thus, the “fundamental frequency” $1/|C|$ always has a non-zero peak, even if the higher harmonics are cancelled out (as with T_2 above).
- Now consider $T_3 = T_{100} \cup T_{10}$, the two-segment pattern discussed above in ex. 5.8, and $T_4 = A \cup B$, where A consists of (consecutive) substrings of elements of $\{(100)^n \mid n \in \mathbb{N}\}$ and B consists of (consecutive) substrings of elements of $\{(10)^n \mid n \in \mathbb{N}\}$. In these cases, we see that

$$\bar{P}[T_3] = \frac{47}{90}\delta_0 + \frac{4}{45}\delta_{1/3} + \frac{3}{10}\delta_{1/2} + \frac{4}{45}\delta_{2/3}$$

as desired above, and

$$\tilde{P}[T_4] = \frac{8}{15}\delta_0 + \frac{2}{15}\delta_{1/3} + \frac{1}{5}\delta_{1/2} + \frac{2}{15}\delta_{2/3}.$$

The graphs of T_3 and T_4 both contain two connected components. The spectrum of the whole pattern is in both cases a weighted average of the spectra of the two components, and the more balanced weighting of the T_4 pattern reflects the existence of more maximal acyclic paths in its graph relative to that of the T_3 pattern.

⁷That is, the smallest in absolute value of a sum of $|q|$ -th roots of unity is at least $(2/|q|)^{|q|}$. To see this, following [28] let $f(k, N)$ denote the smallest non-zero sum of k N -th roots of unity. Then $f(k, N) = f(N - k, N)$, so we can assume $k \leq N/2$. Letting $g(k, N)$ as sum of k N -th roots such that $|g(k, N)| = f(k, N)$, note that $g(k, N)$ is an algebraic number. Hence $|\prod_{i=1}^l g_i(k, N)| > 1$, where the g_i range over the conjugates of $g(k, N)$. Now, $l < N$ and $g_i(k, N) \leq k$, so the smallest such $g_i(k, N)$ is at least k^{-N} . Computational evidence suggests a *much* better lower bound exists, but proving it seems hard. See [28].

- Now consider the pattern $T_5 = \{(123)^n(1234)^m \mid n, m \geq 1\}$. In this case, we can compute

$$\bar{P}[T_5] = (\alpha_{1,0} \cdot \delta_0 + \alpha_{1,1} \cdot \delta_{1/4} + \alpha_{1,2} \cdot \delta_{1/2} + \alpha_{1,3} \cdot \delta_{3/4}) + (\alpha_{2,0} \cdot \delta_0 + \alpha_{2,1} \cdot \delta_{1/3} + \alpha_{2,2} \cdot \delta_{2/3})$$

where $\alpha_i > 0$ for all i . The graph associated with T_5 contains two cycles that are connected but not part of the same strongly connected component. The first cycle, of size 3, generates the three peaks in the second term in the expression above, while the second cycle generates the four peaks in the first term. Both cycles contribute to the δ_0 0 frequency. Similarly, consider

$$T_6 = \{(10^5)^m(10^9)^n(10^{14})^o, m, n, o \in \mathbb{N}\}.$$

Here

$$\begin{aligned} \bar{P}[T_6] &= (\alpha_{1,0} \cdot \delta_0 + \alpha_{1,1} \cdot \delta_{1/6} + \dots + \alpha_{1,5} \cdot \delta_{5/6}) \\ &\quad + (\alpha_{2,0} \cdot \delta_0 + \alpha_{2,1} \cdot \delta_{1/10} + \dots + \alpha_{2,9} \cdot \delta_{9/10}) \\ &\quad + (\alpha_{3,0} \cdot \delta_0 + \alpha_{3,1} \cdot \delta_{1/15} + \dots + \alpha_{3,14} \cdot \delta_{14/15}). \end{aligned}$$

The graph associated with T_6 contains three cycles, all connected but none in the same strongly connected component. The three cycles contribute independently to the existence of the peaks in each of the three terms above. The examples T_5 and T_6 indicate that, just like in the cases of the disconnected cycles in examples T_1 through T_4 , the spectrum of the whole pattern is a weighted average of the spectrum of the patterns associated with the cycles considered separately. Now let $T_7 = T_5 \cup T_6$. Because T_6 has asymptotically n^3 configurations versus n^2 in T_5 , the components in T_5 are completely drowned out by those of T_6 . Hence,

$$\bar{P}[T_7] = \bar{P}[T_6].$$

- Summarizing what we've seen so far, if $G(\Theta) = A \cup B$, where the cycles of A are disjoint from the cycles of B and each cycle is in its own strongly connected component, then

$$\bar{P}_n[G(\Theta)] = \frac{|A_n|}{|A_n \cup B_n|} \bar{P}[A] + \frac{|B_n|}{|A_n \cup B_n|} \bar{P}[B].$$

If A_n and B_n are asymptotically comparable then both the peaks in A_n and B_n survive in the limit; if not, only the peaks of the larger one survive. Hence, for patterns whose graph only contain cycles in separate strongly connected components, the power spectrum only has peaks at multiples of the fundamental frequencies of each cycle separately, but only the cycles in weakly connected component with the largest number of cycles – and thus the largest number of configurations – survive.

- Now let T_8 be the pattern generated by freely alternating the words 1234 and 123456. Then

$$\bar{P}[T_8] = \alpha_0 \delta_0 + \alpha_1 \delta_{1/2},$$

where α_0 and $\alpha_1 > 0$. The graph of associated with T_8 contains one strongly connected component, with two component irreducible cycles, the greatest common divisor of whose lengths is 2. Similarly, let T_9 the pattern generated by freely alternating the words 10^5 , 10^9 , and 10^{14} . In this case,

$$\bar{P}[T_9] = \alpha'_0 \delta_0,$$

that is, T_9 has no non-zero peaks. The graph associated with T_9 also contains only one strongly connected component, with three component irreducible cycles, the gcd of whose lengths is 1. Examples T_8 and T_9 indicate that a general strongly connected component in $G(\Theta)$ contributes to $\bar{P}[T]$ only at frequencies that are multiples of the greatest common divisor of the lengths of its cycles, so that a component C with $\gcd(C) > 1$ contributes nonzero peaks but one with $\gcd(C) = 1$ does not. Again, only the peaks corresponding to the strongly connected components with the largest λ_{max} , contributing the asymptotically largest number of configurations, survive. As a result, $\bar{P}[T_8 \cup T_9] = \bar{P}[T_8]$.

These computations can be summarized as:

Proposition 30 *Suppose Θ is a local check scheme with radius r over m states. Then:*

If $G(\Theta)$ contains no two alternatable segments, then using the notation of sections 2.3.2 and 2.3.3,

$$\bar{P}[\Theta(\mathbb{C})] = \sum_P \sum_{c \in \text{SCC}(P)} \sum_{i=0}^{|c|-1} \alpha_{i,c} \delta_{i/|c|}$$

where the outer sum is taken over all maximal acyclic paths in $G(\Theta)$ for which $n(P) = n(\Theta)$, where $\alpha_{i,c} \in [0, 1]$ with $\alpha_{0,c} > 0$ always, and $\alpha_{1,c} > (2/m)^{m^{2r+1}}$ if $|c| > 1$.

If $G(\Theta)$ contains alternatable segments, then

$$\bar{P}[\Theta(\mathbb{C})] = \sum_{C \in \text{SCC}(G(\Theta))} \sum_{i=0}^{\text{gcd}(C)-1} \alpha_{i,C} \delta_{\frac{i}{\text{gcd}(C)}}$$

where the sum is taken over all strongly connected components C with $\lambda_{\max}(C) = \lambda_{\Theta}$. Again, $\alpha_{0,C} > 0$ always, and $\alpha_{1,C} > (2/m)^{m^{2r+1}}$ if $\text{gcd}(C) > 1$.

This computation emphasizes the importance of the “fundamental frequencies” of a pattern.

Definition 31 [Fundamental Frequencies] *Suppose T is a locally checkable pattern, and $T = \Theta(\mathbb{C})$ for a local check scheme Θ . Then the fundamental frequencies of T are numbers $\eta \in [0, 1)$ of the form*

$$\eta = \frac{1}{\text{gcd}(C)}$$

for all strongly connected components C in $G(\Theta)$, when $\text{gcd}(C) > 1$. If $\text{gcd}(C) = 1$, then we assign to C the fundamental frequency 0. Let $\omega(T)$ denote the set of fundamental frequencies.

Combining prop. 11 and prop. 30 yields that for an infinite pattern $\Theta(\mathbb{C})$, $\omega(T)$ contains at least one element, and that any $\omega \in \omega(T)$ is either 0 or $1/i$ for an integer $i \leq m^{2r+1}$. Now, recall in §4.2 I introduced the idea of a weakly locally patchable pattern T as one in which that admits a local checks scheme Θ for which $\Theta(\mathbb{C})$ is locally patchable. In terms of the graph structure, proposition 19 implies that if T is not weakly locally patchable, all the strongly connected components $C \in \text{SCC}(G(\Theta))$ have $\text{gcd}(C) > 1$. Hence, a (strongly) locally patchable pattern has all elements of $\omega(T)$ being 0; a weakly locally patchable pattern has at least one 0 fundamental frequency; and a pattern T that is not weakly locally patchable has $\omega > 0$ for all $\omega \in \omega(T)$.

We can now complete the proof of prop. 23.

Corollary 3 *Suppose T is a non-weakly locally patchable pattern. There is a constant $C(r)$ such that for any local rule of radius r that is a solution to T and any T -admissible size n ,*

$$TTS^{\text{avg}}(f)(n) \geq C(r) \cdot n.$$

Proof: Suppose $T = \Theta(\mathbb{C})$ for some local check scheme of radius r . Suppose f is a solution to T with radius R . Suppose n is a T -admissible size. For all $X \in \mathbb{C}_n$, $\bar{P}[X]$ contains a peak of height at least $\Gamma_{m,r}/m^{2r+1}$ at one of the fundamental frequencies of a cycle in $G(\Theta)$. Since there are most m^{2r+1} possible frequencies, this implies that for some $\omega_n > 0$, a fraction of at least $1/m^{2r+1}$ of $X \in \mathbb{C}_n$ have $\bar{P}[X](\omega_n) \geq \Gamma_{m,r}/m^{2r+1}$. Hence $\bar{\mathcal{G}}_n(\omega_n) \geq \frac{\Gamma_{m,r}}{m^{2(2r+1)}}$. Since T is assumed to not be locally patchable, $\omega_n \neq 0$, so we can apply eq. 5.11 which yields

$$T_n \geq \frac{C\Gamma_{m,r}^5}{(2R+1)m^{2(2r+1)}} n$$

for some $C > .0002$. ■

5.6 A Frequency Independent Approach

Equation 5.7 applies for each frequency ω separately. It would also be desirable to have a measure that is frequency-independent. A simple way to do this is to compute the *sum* the frequency-dependent order measure $\mathcal{P}(\omega)$ over all frequencies ω . Of course, we'd only want to sum $S(X) \triangleq \sum_{\omega>0} \mathbb{P}[X](\omega)$ for ω ranging only over non-zero frequencies, because otherwise we'd be unable to distinguish random configurations from highly ordered ones – after all, $\sum_{\omega \geq 0} \mathbb{P}[X](\omega) = 1$ for all configurations X .

In fact, even this restriction to $\omega > 0$ isn't quite good enough. It is easy to see that $\langle S(X) \rangle_{X \in \mathbb{C}_n} \sim (1 - \frac{1}{n})$. Hence, even for random configurations X the value of $S(X)$ is typically so large that it looks no different from its value on highly ordered configurations. Thus, $S(X)$ cannot be used as a slowly-growing measure of order. Conceptually, what's happening is that, since for each ω separately, $\langle \mathbb{P}[X](\omega) \rangle_{X \in \mathbb{C}_n} \sim O(1/n)$, by the time we've summed up $\sim O(n)$ different frequencies, $\langle S(X) \rangle$ is $\sim O(1)$. To put it another way: the peaks of the power spectrum are distinguishable from noise to order $O(1/n)$ for each frequency ω ; but we need a measure that distinguishes peaks from noise to order at least $O(1/n^{1+\epsilon})$ for $\epsilon > 0$ if we want the distinction to survive a sum over a large number of frequencies.

There is an obvious way to achieve this. Since $\mathcal{P}[X](\omega) \sim O(1/n)$ for random configurations, $(\mathbb{P}[X](\omega))^2$ will typically be $O(1/n^2)$. Hence, the sum of the *squares* of the power spectrum on non-zero frequencies might be a good measure of order:

Definition 32 [*Fourth-Power Sum*] Let J be the function defined on configuration X of size N by

$$J[X] = \sum_{k=1}^{N-1} \|\mathcal{F}[X](\omega = k/|X|)\|^4.$$

That is, $J[X]$ is the sum of the fourth-powers $\|\mathcal{F}[X](\omega)\|^4$ for non-zero frequencies.

It turns out that J is indeed a slowly-growing order measure. To see this, first note that, like \mathcal{P} , J can also be computed in terms of the of the autocorrelation function:

Proposition 31 For all X of size N ,

$$J[X] = \text{Var}(\sigma[X])$$

where $\text{Var}(\sigma[X])$ is the variance of $\sigma[X]$ considered as a vector in \mathbb{R}^N .

Proof: Obviously

$$J[X] = \sum_{k=0}^{N-1} \|\mathcal{F}[X](k/|X|)\|^4 - \|\mathcal{F}[X](0)\|^4.$$

Now,

$$\begin{aligned} \sum_{k=0}^{N-1} \|\mathcal{F}[X](k/N)\|^4 &= \left(\frac{1}{N} \sum_{\tau=0}^{N-1} \cos(2\pi k\tau/N) \sigma'[X](\tau) \right)^2 \\ &= \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{\tau} \sum_{\tau'} \cos(2\pi k\tau/N) \cos(2\pi k\tau'/N) \sigma'[X](\tau) \sigma'[X](\tau') \\ &= \frac{1}{N^2} \sum_{\tau} \sum_{\tau'} \sigma'[X](\tau) \sigma'[X](\tau') \left(\sum_{k=0}^{N-1} \cos(2\pi k\tau/N) \cos(2\pi k\tau'/N) \right) \\ &= \frac{1}{N^2} \sum_{\tau} \sum_{\tau'} \sigma'[X](\tau) \sigma'[X](\tau') \cdot \frac{N}{2} (\delta_{\tau,\tau'} + \delta_{\tau,N-\tau'}) \\ &= \frac{1}{N} \sum_{\tau} (\sigma'[X](\tau))^2 \end{aligned} \tag{5.12}$$

where $\delta_{a,b}$ is the Krocker delta. In the fourth equality I used the orthogonality of cosines of different frequencies. On the other hand,

$$\|\mathcal{F}[X](0)\|^4 = \frac{1}{N^2} \left(\sum \sigma'[X](\tau) \right)^2$$

so

$$J[X] = \frac{1}{N} \sum_{\tau} (\sigma'[X](\tau))^2 - \frac{1}{N^2} \left(\sum \sigma'[X](\tau) \right)^2 = \frac{1}{N^2} \sum_{\tau} \sum_{\tau'} (\sigma'[X](\tau) - \sigma'[X](\tau'))^2.$$

Since $\sigma[X](\tau) = \sigma'[X](\tau) - (X \otimes X)^\dagger$, we have $\sigma[X](\tau) - \sigma[X](\tau') = \sigma'[X](\tau) - \sigma'[X](\tau')$, yielding the result. ■

The key point is that:

Proposition 32 *Suppose F is a local rule of radius r . Then for $n > 4r + 2$,*

$$\langle J(F(X)) \rangle_{X \in \mathbb{C}_n} \leq \frac{8(r+1)}{n}.$$

To prove this, it will be useful to have the following

Lemma 1 *Suppose F is a local rule of radius r . If $n > 4r + 2$ and $\tau \in [2r + 2, n - 2r - 1]$,*

$$\left| \langle \sigma(F(X)) \rangle_{X \in \mathbb{C}_n} \right| \leq \frac{4r + 2}{n - 4r + 2}.$$

Proof: Recall that by definition $\sigma[X](\tau) = \sigma'[X](\tau) - (\bar{X} \otimes \bar{X})^\dagger$. On the other hand, it's easy to see that

$$\sum_{\tau} \sigma'[X](\tau) = (\bar{X} \otimes \bar{X})^\dagger \cdot |X| \quad (5.13)$$

which implies

$$\sum_{\tau} \sigma[X](\tau) = 0. \quad (5.14)$$

Now, adopt the notation $A \triangleq [0, 2r] \cup [n - 2r - 1, n - 1]$ as in a previous section and $\sigma_n \triangleq \langle \sigma[F(X)](2r + 1) \rangle_{X \in \mathbb{C}_n}$. Then combining eq. 5.14 and prop. 26, we have

$$0 = \sum_{\tau} \langle \sigma[F(X)](\tau) \rangle_{X \in \mathbb{C}_n} = \sum_{\tau \in A} \langle \sigma[F(X)](\tau) \rangle_{\mathbb{C}_n} + (n - 4r - 2) \cdot \sigma_n$$

so that

$$|\sigma| = \frac{1}{n - 4r - 2} \left| \sum_{\tau \in A} \langle \sigma[F(X)](\tau) \rangle_{\mathbb{C}_n} \right| \leq \frac{4r + 2}{n - 4r - 2}$$

as claimed. ■

Proof: (Of prop. 32) Let

$$A_n \triangleq \frac{1}{n^2} \langle \left(\sum \sigma'[F(X)](\tau) \right)^2 \rangle_{X \in \mathbb{C}_n}$$

and

$$B_n \triangleq \frac{1}{n} \sum_{\tau} \langle (\sigma'[F(X)](\tau))^2 \rangle_{X \in \mathbb{C}_n}.$$

We will compute A_n and B_n and then take their difference. First, to compute B_n : note that

$$\langle \sigma'[F(X)]^2(\tau) \rangle_{X \in \mathbb{C}_n} = \frac{1}{n^2} \sum_{\substack{a,b,a',b' \\ F(a)=F(b) \\ F(a')=F(b')}} \langle N_{a,b}^\tau(X) N_{a',b'}^\tau(X) \rangle_{X \in \mathbb{C}_n} \quad (5.15)$$

where $N_{a,b}^\tau(X)$ is the number of t such that $B_r(t, X) = a$ and $B_r(t + \tau, X) = b$. Now, if $\tau \in A = [2r + 1, n - 2r - 1]$, then for all $a, a', b, b' \in \mathcal{B}_{r,m}$,

$$\left| \langle N_{a,b}^\tau(X) N_{a',b'}^\tau(X) \rangle - \langle N_{a,b}^\tau(X) \rangle \langle N_{a',b'}^\tau(X) \rangle \right| \leq \frac{n}{m^{4(2r+1)}}. \quad (5.16)$$

Plugging eq. 5.16 into eq. 5.15, we get

$$\left| \langle \sigma'[F(X)]^2(\tau) \rangle - (\langle \sigma'[X](\tau) \rangle)^2 \right| \leq \frac{1}{n}.$$

Thus applying the lemma, we have for $\tau \in A$ that

$$\left| \langle \sigma'[X]^2(\tau) \rangle - \Gamma^2 \frac{n}{n - 4r - 2} \right| \leq \frac{4r + 2}{n - 4r - 2}$$

where

$$\Gamma = \left\langle \sum_j \left(\sum_{a|F(a)=j} \frac{N_a[X]}{n} \right)^2 \right\rangle.$$

Summing this over $\tau \in A$ gives

$$\left| \frac{1}{n} \sum_{\tau \in A} \langle \sigma'[X]^2(\tau) \rangle - \Gamma^2 \right| \leq \frac{4r + 3}{n}$$

so that

$$|B_n - \Gamma^2| \leq \frac{8r + 5}{n}. \quad (5.17)$$

To compute A_n : note that due to eq. 5.13,

$$A_n = \frac{1}{n^4} \sum_{j,j'} \sum_{\substack{a,b,a',b' \\ F(a)=F(b)=j \\ F(a')=F(b')=j'}} \langle N_a[X] N_b[X] N_{a'}[X] N_{b'}[X] \rangle \quad (5.18)$$

where $N_a[X]$ is the number of t such that $B_r(t, X) = a$. But for all $a, b, c, d \in \mathcal{B}_{r,m}$,

$$\left| \langle N_a[X] N_b[X] N_{a'}[X] N_{b'}[X] \rangle - \langle N_a[X] N_b[X] \rangle \langle N_{a'}[X] N_{b'}[X] \rangle \right| \leq \frac{3n^3}{m^{4(2r+1)}} \quad (5.19)$$

and plugging eq. 5.19 into eq. 5.18 gives

$$|A_n - \Gamma^2| \leq \frac{3}{n}. \quad (5.20)$$

Finally, combining eqs. 5.17 and 5.20 gives

$$\langle J(F(X)) \rangle = B_n - A_n \leq \frac{8(r+1)}{n}$$

as desired. ■

As a result of proposition 32, we can go through each of the steps of the steps leading to the lower bound eq. 5.7, with $J(X)$ in place of $\mathcal{P}(\omega)$. This is useful because $J(X)$ will typically be larger than $\mathcal{P}(\omega)$, since the former takes into account many frequencies at once, and therefore provides sharper lower bounds. In future work, I intend to develop the theory of $J(X)$ – and other more sophisticated measures of order, more thoroughly.

Related Work

The use of Fourier analysis in theoretical computer science has gained some popularity in the past fifteen years (see e.g. [21]), though I am not aware of a usage quite like the one here. I have found a paper in which Fourier analysis is used to explore features of regular languages, in the context of cellular automata theory, although the motivation, techniques, and results of the paper seem fairly different from those developed here [19]. The power spectrum, considered as an invariant of regular languages, is like several of the properties described in previous chapters, in that it is “more detailed” in scope than traditional measures of structure in regular languages. It is, for instance, impossible to compute the Fourier coefficients of a language from the algebraic generating function representations as described in [8] or [9].

Chapter 6

The Radius-State Tradeoff

The multi-agent systems model being used in this work has two parameters, the radius r that an agent can see and the amount of state m that it can store. In practical implementations of an actual multi-agent system, these parameters will often be resource-limited, since agents have limits on either their internal memory and communications bandwidth. Up to this point, we have assumed that the amount of state is implicitly given by the pattern and then determined the best (smallest) radius at which the problem was solvable.¹ In this chapter, I show that there is a *radius-state resource tradeoff*: systems that have a large amount of available state can achieve with a reduced radius the same goals achieved by a system with a small amount of state and a larger radius. More precisely, I discuss two forms of radius-state tradeoff, presenting:

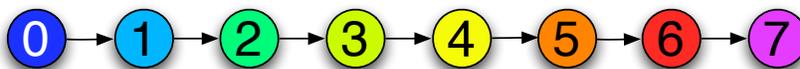
- A “static tradeoff” that applies to local check schemes, establishing that any local check scheme lies along a continuum between low-radius/high-state and low-state/high-radius implementations, and exhibiting algorithms for “tuning” along this continuum.
- A “dynamic tradeoff” that applies directly to local rules, showing that for certain forms of local rule, extra states used to aid the dynamics (like the \triangleright , \triangleleft , and Δ_i states used in chapters 3 and 4) can be removed by encoding them with larger-radius dynamical ensembles.

6.1 The Static Tradeoff

I introduce the static tradeoff via its most important example: the *coordinate problem*.

6.1.1 Example: The Coordinate Pattern

Suppose our task is to self-organize a coordinate system within the one-dimensional model system. Let’s make the assumption that the number of possible agent internal states, $|S| = m$, is large compared to the size of a configuration X . A “coordinatization” of a configuration X with of size 8, with agents that had at least 8 states available, would look something like:



One simple way to achieve this state is via the nearest-neighbor local rule F defined on the local ball b by

$$F(b) = \begin{cases} 0, & \text{if } b \text{ is the left-most agent} \\ 1 + \min(b), & \text{otherwise} \end{cases} \quad (6.1)$$

in which $\min(b)$ denotes the minimum state value seen in the local ball b . In this formation, the left-most agent is the *anchor* or *source* of the gradient. As long as $|X| < m$, this rule will generate a state in which

¹Though occasionally we added some hidden state for ease of explication, and as we discuss, this state can be removed.

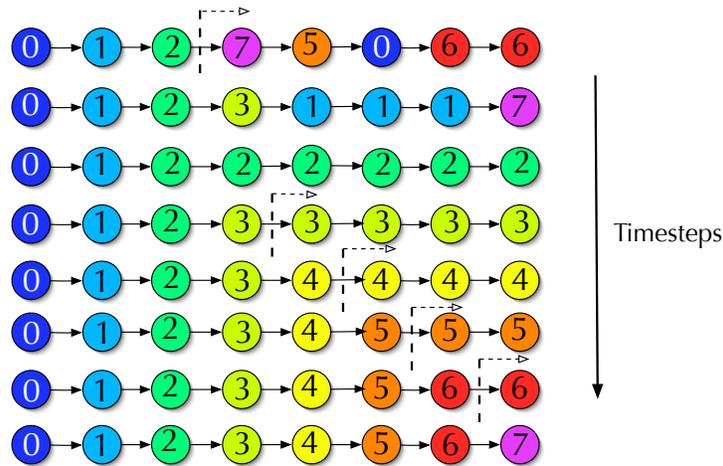


Figure 6.1: The gradient propagation process according to eq. 6.1. Each horizontal line corresponds to a new timestep.

each agent’s position is reflected in its internal state value. The “way that it works” is that – regardless of initial state – the left-most agent serves as an anchor for the origin position, and the correct coordinate value propagates through the system, as seen in figure 6.1.

Hence the “coordinate pattern” – in which each agent’s state reflects relative position in configuration – is solvable with a radius 1 algorithm, as long as the amount of amount of state available to each agent can grow with the number of agents present. This algorithm, with slight modifications, works in much more general contexts, and is known as a *discrete gradient* [1]. Because coordinate systems are basic to very many tasks, the gradient is a tool of great importance in local-to-global programming.

Though the coordinate problem may look different from the locally checkable problems we’ve seen so far, the discrete gradient algorithm is *almost* implementing a local check scheme. Specifically, let

$$\Theta(b) = \begin{cases} 1 & , \text{ if } b(-1) = b(0) - 1 \\ 0 & , \text{ otherwise} \end{cases} .$$

This check scheme has radius of 1/2, meaning that the local check scheme only makes use of information from the agent to the left. This formulation is somewhat problematic because it relies on each agent having more and more state as the system size increases. One way to resolve this apparent problem is to allow infinite state sets, like the whole set \mathbb{N} of natural numbers, assuming that memory is cheap. Another resolution is to consider for each k the k -coordinatization problem – that is, the ability to produce a coordinate modulo k . In this case, the local rule:

$$F(b) = 1 + b(-1) \pmod k$$

produces the correct result. For each k , the k -coordinatization problem is locally checkable with k states, with radius 1/2.

But what if we want to achieve k -coordinatization with fewer than k states? On the one hand, it is impossible to find a local check scheme for k -coordinatization with radius 1/2 and fewer than k states: to see this, simply note that the local check scheme Θ for a k -coordinatization corresponds to $G(\Theta)$ being containing a single irreducible cycle of length k in $ID(2, k)$; and such cycles are size at most k .

However, if we allow a somewhat larger radius, can we lower the required amount of state? Yes. A well-known idea in computer science is the idea of the *DeBruijn sequence*: for m states and window-length n , a DeBruijn sequence $B(n, m)$ is an m -ary sequence of length- m^n in which each length- n m -ary sequence arises exactly once as a contiguous subsequence in $B(n, m)$, wrapping around at the end. For example, a DeBruijn sequence with $n = 3$ and $m = 2$ is 10111000. It is a classic and simple result that such sequences

exist for all m and n ([7])² The key realization is to think of the view within each length- n window in $B(n, m)$ as encoding a unique position along a line up to m^n positions in length.

Example 37 A coordinate gradient of length 8 can be encoded by a DeBruijn sequence with $n = 3$ and $m = 2$ (figure 6.2). In particular, we have: $101 \rightarrow 0, 010 \rightarrow 1, 100 \rightarrow 2, 000 \rightarrow 3, 001 \rightarrow 4, 011 \rightarrow 5, 111 \rightarrow 6$, and $110 \rightarrow 7$.

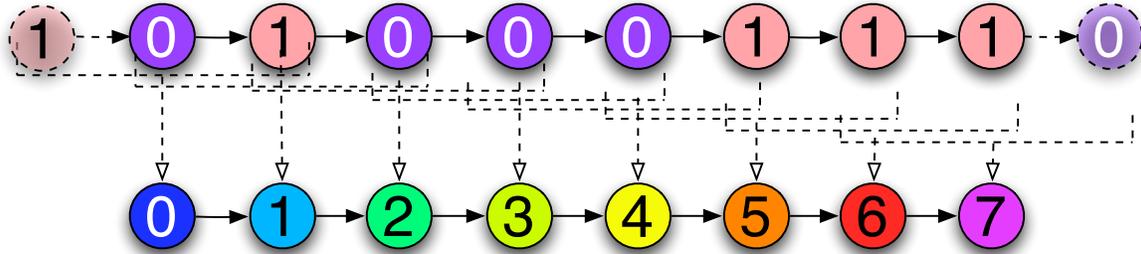


Figure 6.2: DeBruijn sequence (low state, high radius) encoding of gradient (low radius, high state).

It turns out that there is a radius $\lceil (n+1)/2 \rceil$ check scheme for the DeBruijn sequence $B(n, m)$, corresponding to an induced cycle of length m^n in $\mathbb{D}(n+1, m)$. In fact, it can be achieved simply by reading off the length- $n+1$ windows in $B(n, m)$.

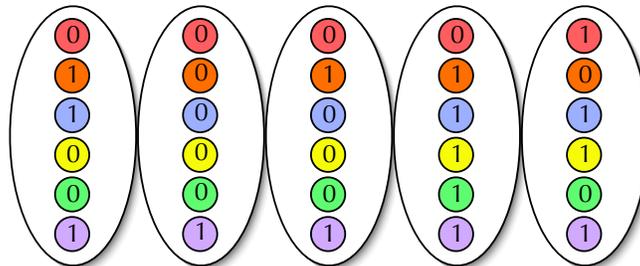
Example 38 A local check scheme of radius $3/2$ for $B(3, 2)$ is given by:

$$\Theta(b) = \begin{cases} 1 & , \text{ for } b = 1010, 0100, 1000, 0001, 0011, 0111, 1110, \text{ and } 1101 \\ 0 & , \text{ otherwise} \end{cases}$$

The relationship between discrete gradients (with fixed amounts of state) and DeBruijn sequences is an example of a more general idea: the *radius-state tradeoff*, in which patterns that are locally checkable with a given amount of state and radius can be written as encodings of patterns that are checkable with more states and less radius or vice versa. To indicate how this tradeoff might be implemented in general, I now discuss the translation of local check schemes of a given radius and amount of state to one with low radius but higher state, and the reverse.

6.1.2 Trading Radius for State

A simple procedure for trading radius for state can be described if we first add a little more structure to our model of agents. Instead of modeling the states of the agents as an unstructured set S , suppose internal state is now specified as a length- k m -sequence. For instance, if $k = 6$, then the model will look something like this:



²They correspond to hamiltonian cycles in the strongly connected component of the graph $\mathbb{D}((n-1)/2, m)$, an idea I'll explore in greater detail in chapter 7.1.

in which the large ovals correspond to agents and each of the slots is assigned a unique color. Each ball $b \in \mathcal{B}_r$ in this model is like a k -tuple of balls in a single-slot model, so we write $b = (b_1, b_2, \dots, b_k)$. Formally, each configuration in this new model is a function

$$X : \{1, \dots, |X|\} \rightarrow [m]^k$$

where $[m] = \{1, \dots, m\}$. Let $b_{j,r}(i, X)$ denote the r -ball in X at agent i , for slot j —formally, $b_j = (X(i-r)_j, \dots, X(i+r)_j)$ where a_j denotes the j -th coordinate in a . In the above figure, the radius 2 ball around the third agent $B_2(3, X)$, consists of $(b_1, b_2, b_3, b_4, b_5, b_6)$, in which $b_1 = (0, 0, 0, 0, 1)$, $b_2 = (1, 0, 1, 1, 0)$, $b_3 = (1, 0, 0, 1, 1)$, $b_4 = (0, 0, 0, 1, 1)$, $b_5 = (0, 0, 0, 1, 0)$, and $b_6 = (1, 1, 1, 1, 1)$.

Since each of the k “ m -ary register” slots can take on m possible states, the k -slot model is effectively like choosing S to have m^k states. Hence, any check scheme or local rule in multi-slot model can be translated into a corresponding check scheme or local rule in the original model, and vice-versa.

Now, suppose we’re given a radius r local check scheme Θ with m states, and our goal is to get a radius l local check scheme for the pattern $\Theta(\mathbb{C})$ generated by Θ , for any $l < r$. We will associated to Θ into a new local check scheme Θ^l with radius l :

$$\Theta = \text{Radius } r \text{ check scheme} \longrightarrow \Theta^l = \text{radius } l \text{ check scheme, using more state.} \quad (6.2)$$

Specifically, let Θ^l be the radius- l local check scheme on $M = 2\lceil(r-l)/(2l+1)\rceil + 1$ slots defined by the criterion that:

$$\Theta^l(b = (b_1, \dots, b_l)) = 1$$

if and only if there is an $X \in \Theta(\mathbb{C})$ and $i \leq |X|$ such that

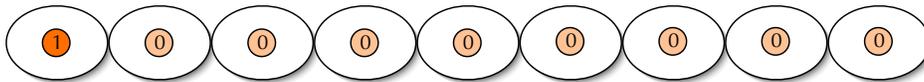
$$b_j = B_l(i + j(2l + 1), X) \text{ for } j = -(M - 1)/2, \dots, (M - 1)/2.$$

Evidently

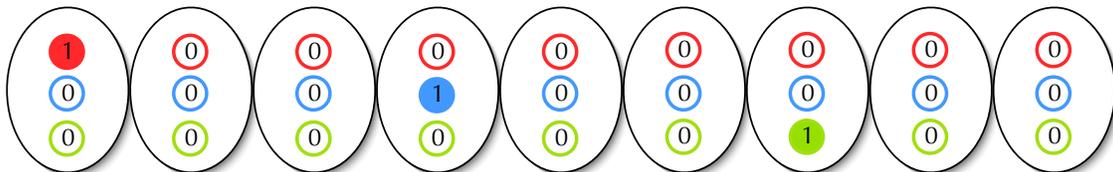
$$\Theta^l(b) = 1 \iff \Theta(b_1 \circ b_2 \circ \dots \circ b_l) = 1.$$

Hence in configuration X satisfying Θ^l , the original pattern can be read off as the states in the “first register” b_1 , or in a shifted version in any other register. This construction can be thought of as a “cut and shift” construction, in which the original local check scheme is segmented into a number of separate parts, which are then shifted through the slots. Evidently this procedure can be translated to the original “one-slot” agent model through simple “decoding” function.

Example 39 Consider the $T_{100000000}$ pattern in the 1-slot model.



This pattern has local check radius of 4. Now suppose we wish to get a radius-1 local check. The “cut and shift” scheme applied to this problem yields a 3-slot local check that looks like:



Let’s now make the formal definition:

Definition 33 [Local Encodings] A pattern T' over state set S' is a k -encoding of a pattern T over state set S if there is a function

$$\phi : \mathcal{B}_{k,S'} \rightarrow S$$

such that

$$T = \{\phi(X) | X \in T'\}$$

where $\phi(X)$ denotes the simultaneous action of ϕ on every k -ball in X . In case this holds, ϕ is called a decoding function.³

In terms of the definition of this definition, the cut-and-shift procedure defined by the construction of Θ' above shows that: any local check scheme over m states with radius r is the 1-encoding of a radius-1 check scheme with m^{2r+1} states.

6.1.3 Trading State for Radius

Now, we'd like to make the reverse of the construction described in the previous section. This turns out to be somewhat more complicated.

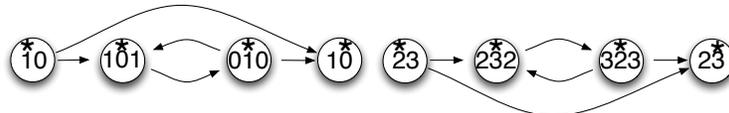
The most obvious thing to look for would be a procedure that, given local check scheme Θ with radius r and m states, finds a local check scheme Θ' with 2 states and a radius $R \geq r$ such that $\Theta(\mathbb{C})$ is the R -encoding of $\Theta'(\mathbb{C})$. However, this is impossible, for a very simple reason. Suppose we're given the trivial local check scheme Θ on r states – that is, Θ that accepts all r -balls. Then $Growth_{\Theta(\mathbb{C})}(n) \sim m^n$; that is, the number of possible Θ -admissible structures of size at most n grows exponential in n , with growth rate m . On the other hand, if Θ' is a supposed local check scheme in $m' < m$ states, then $Growth_{\Theta'}(n)$ grows no faster than $O((m')^n)$. But then no decoding function ϕ of any radius k can possibly produce $\Theta(\mathbb{C})$ as the image $\phi(\Theta'(\mathbb{C}))$.

Another way to view this problem is to think about the graphs associated with the local check schemes. The graph of the trivial pattern over m states is the complete graph with degree m ; but all graphs of a local check schemes with fewer than m states have degree less than m , and there is no way that any could represent a degree- m graph. Generically, if we define a “degree- k ” pattern as one whose graph $G(\Theta)$ has maximum in- or out-degree of k , then evidently no degree- k pattern can be encoded by a degree k' pattern if $k' < k$, regardless of additional radius.

Example 40 Conversely, let's take the pattern over 4 states given by

$$T = \{10, 1010, 101010, \dots\} \cup \{23, 2323, 232323, \dots\} = T_{10} \cup T_{23}.$$

This pattern possesses a radius-1 check scheme Θ whose graph $G(\Theta)$ is

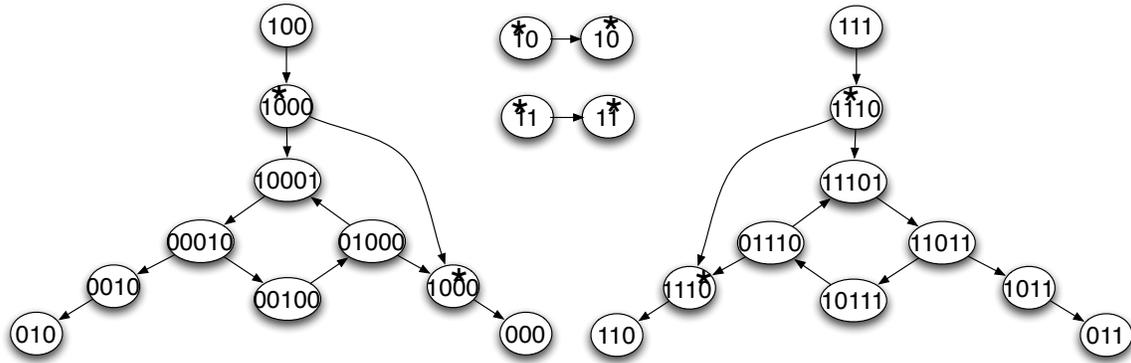


This pattern is “degree 2”, since its graph has 2 as its maximal in or out degree. On the one hand, there is no local check scheme over 2 states, of any radius, whose graph is isomorphic to the above graph. That is because it has two 2-cycles, while there is a unique length-2 periodic minimal segment in two states (namely 10), corresponding to the single length-2 cycle in $ID(2r + 1, 2)$. Nonetheless, the pattern T can be viewed as the encoding of a radius-2 local check scheme in 2 states. In particular, consider the pattern

$$T' = \{10, 1000, 100010, 10001000, 1000100010, \dots\} \cup \{11, 1110, 111011, 11101110, 1110111011, \dots\}.$$

This pattern has a radius-2 local check Θ' whose graph is:

³The notion of encoding makes another connection to the “language recognition” point of view. By induction on the basic closure operations of regular languages, it is easy to prove that all regular languages are local encodings of (one-dimensional) locally generated languages, and all local encodings of locally generated languages are regular. In other words, the regular languages are, in language-theoretic terms, the “homomorphic images” of locally generated languages (see [26]). In a sense, local checkability therefore captures the “the most natural physical model” of regular languages, at least for the purposes of the construction problem.



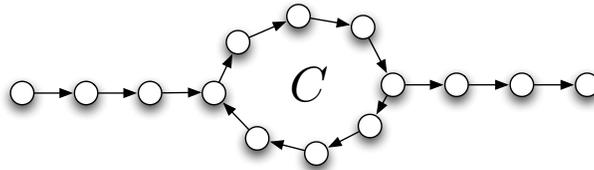
Now consider the radius 1 decoding function ϕ defined on the 1-ball (x, y^{star}, z) by the formula

$$\begin{aligned} \phi((x, y, z)) &= \delta_{x,0}\delta_{z,0} \cdot \mathbf{1} + (1 - \delta_{x,z})(\delta_{y,0} \cdot \mathbf{0} + \delta_{y,1} \cdot \mathbf{2}) + \delta_{x,1}\delta_{z,1} \cdot \mathbf{3} \\ &= xz + 3(x-1)^2(z-1)^2 + 2(1-(x-z)^2)(y-1)^2 \end{aligned} \quad (6.3)$$

where $\delta_{a,b}$ is the Kronecker delta. The key point is that ϕ generates T from T' , i.e. $\phi(T') = T$. Hence, this degree-2 pattern radius 1 pattern over 4 states can in fact be encoded as a 2-state pattern with radius 2. Intuitively, what is happening is that the small cycles and their single terminal paths in the original pattern are expanded into larger cycles with multiple terminal paths in the encoded pattern, which are then collapsed by the decoding function. In other mathematical terms, the pair $(G(\Theta'), \phi)$ is a covering of $G(\Theta)$ with multiplicity 2. Since all the local neighborhoods in the original graph are degree 2, these can be reproduced in the encoded graph.

Whether a generalization of this graph covering procedure can be constructed to obtain from all degree- k patterns an exact encoding with k states is an open question. However, it is possible to construct an *approximate* encoding – an encoding of a *subpattern* of $\Theta(\mathbb{C})$ that has a valid configuration in all but finitely many Θ -admissible sizes. To see how this works, I will sketch the construction process in several steps:

Step 1: Suppose we're given a repeat pattern T_q , in m states which has a local check scheme Θ of radius r . The graph $G(\Theta)$ contains a single cycle C corresponding to the repeated unit q , and length- r initial and terminal paths:



Since C is an irreducible cycle in $\mathbb{D}(r, m)$, $|C| \leq m^{2r}$. Hence, to solve our problem, we need to show that for any $l \leq m^{2r}$, there is some R and a 2-state local check scheme with radius R whose graph $G(\Theta)$ consists of a single cycle of length l . In fact, take $R = 2M + 1$, where

$$M = \lceil r \log_2(m) \rceil.$$

Now, let $k = \text{mod}(l, R)$. Since $l \leq m^{2r}$, $L \triangleq \lceil l/(2M+1) \rceil \leq 2^M/M$. Thus, there are at least $L+1$ unique length- M binary sequences, so at least L that are not the all-ones sequence 1^M . Enumerate a size- L selection of non- 1^M binary length- M binary sequences as b_1, \dots, b_L . Now, consider the binary sequence

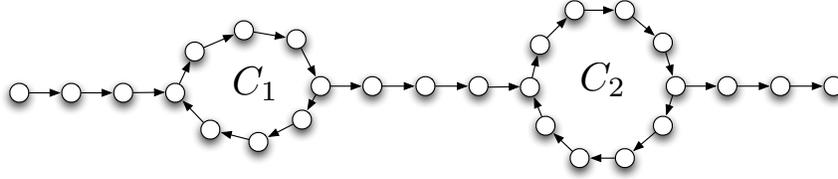
$$S \triangleq 1^M 0 \circ b_1 \circ 1^M 0 \circ b_2 \circ \dots \circ 1^M 0 \circ b_L \circ 1^{k-1} 0.$$

The repeat pattern with unit S , i.e. T_S , is locally checkable with radius R – and the graph of its local check scheme is a cycle of length $|S| = l$. Hence,

$$\phi(B_R(i, S)) = C(i) \text{ for } i = 1, \dots, l$$

is the desired encoding.

Step 2: Now suppose T is the concatenation of two repeat patterns, i.e. $T = \{x \circ y\}$, where $x \in T_q$ and $y \in T_{q'}$. Suppose that the overall local check scheme has radius r . It's graph looks like:



where the cycles C_1 and C_2 correspond to the minimal repeatable segments q and q' . As long as $|C_1| + |C_2| \leq \lceil r \log_2(m) \rceil$, then we can repeat the above process to construct sequences

$$S_1 = 1^M 0 \circ b_1^1 \circ 1^M 0 \circ b_2^1 \circ \dots \circ 1^M 0 \circ b_l^1 \circ 1^{k_1-1} 0$$

and

$$S_2 = 1^M 0 \circ b_1^2 \circ 1^M 0 \circ b_2^2 \circ \dots \circ 1^M 0 \circ b_l^2 \circ 1^{k_2-1} 0$$

of lengths $|C_1|$ and $|C_2|$ respectively, where $k_1 = \text{mod}(|C_1|, R)$ and $k_2 = \text{mod}(|C_2|, R)$, such that

$$b_k^i = b_l^j \text{ if and only if } i = j \text{ and } k = l.$$

Now consider all the configurations

$$T' = \{(S_1)^n \circ (S_2)^m \mid m, n \in \mathbb{N}\}.$$

In the case that either (i) $|S_1| \geq M$ or $|S_2| \geq M$, or (ii) $|S_1| \neq |S_2|$, any $R = 2M + 1$ window B in X can determine whether it is part of S_1 or S_2 , and what position p_B it has within S_1 or S_2 . The decoding function ϕ defined by

$$\phi(B) = \begin{cases} C_1(p_B), & \text{if } B \in S_1 \\ C_2(p_B), & \text{if } B \in S_2 \end{cases}$$

is therefore well-defined and $T = \phi(T')$. Hence T' is an exact 2-state M -encoding of T . If on the other hand $|S_1| = |S_2| < M$, then ϕ will not be well-defined, and the process we've defined won't yield an encoding of T . However, an agent that is the \star -agent of a ball B of radius M that is within M states of the right end of a configuration can its $|X| - \text{pos}(B, X)$, and an agent not within M states of the right end can determine that $|X| - \text{pos}(B, X) > M$. Hence, the decoding function ϕ defined by

$$\phi(B) = \begin{cases} C_1(p_B), & \text{if } |X| - \text{pos}(B) > M \\ C_2(p_B), & \text{if } |X| - \text{pos}(B) \leq M \end{cases}$$

is well-defined, and $\phi(T') \subset T$. This subset inclusion is proper, because the configurations $q^m (q')^n$ with $n > M/|q'|$ will never arise. On the other hand, since $|q| = |q'|$, $\text{Sizes}(T') = \text{Sizes}(T)$, so ϕ is an approximate M -encoding. Intuitively, the problem encountered here is the same as in example 40 above, in which there are "too many small cycles" to be encoded by a 2-state check scheme of any radius. However, as opposed to the strategy taken in the encoding in eq. 6.3, in which we produce an exact encoding by quotienting a high-multiplicity covering, here we simply ignore one of the small cycles in $G(\Theta)$ – at the expense of producing an approximate encoding.

Step 3: Repeating the above reasoning, we are able to generate approximate 2-state radius $2 \lceil r \log_2(m) \rceil + 1$ encodings of all local check schemes Θ of radius r in m states for which $G(\Theta)$ has maximum in- and out-degree of 2.

Step 4: The final step is to show that any degree- k pattern T in m states with local check scheme of radius r has an approximate encoding (also in m -states) by a degree-2 subpattern T' with check scheme of radius

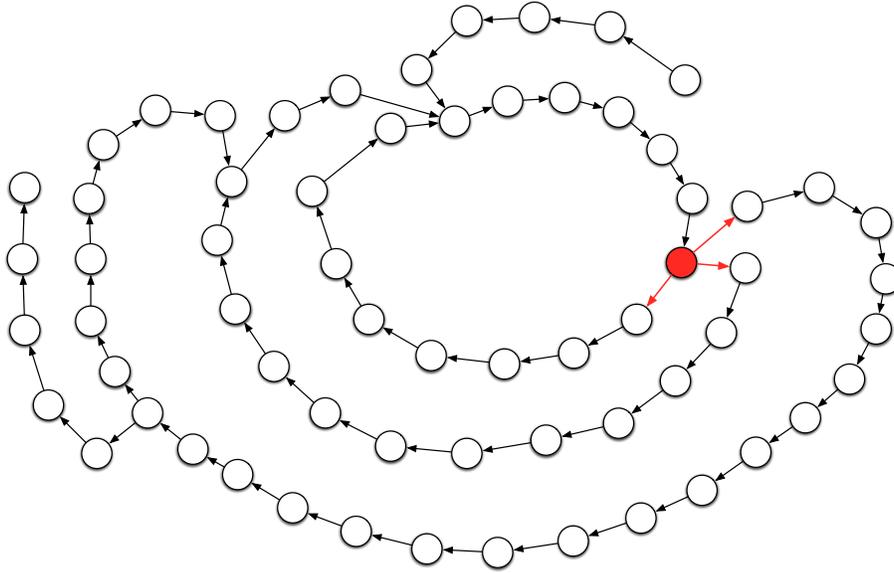
$2r + 1$. To do this, one first picks disjoint linear suborderings of the graph $G(\Theta)$, each component of which contains a cycle of each size to which it is connected, and then applies the same procedure as in step 3 above to that ordering. For instance, suppose we take the pattern

$$S = \{A^m \circ 11 \circ B^m \circ 1000111000011 \circ C^o \mid m, o \geq 1, n \in \mathbb{N}\}$$

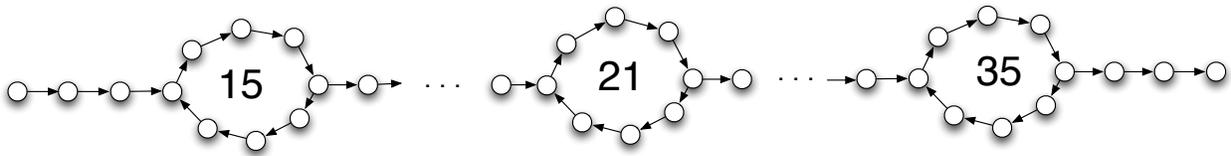
where

$$A = 11100011100011, \quad B = 210011100001110001111, \quad C = 11100001110001111001011100111110100.$$

The pattern T generated by the radius-5 windows in S has local check scheme Θ whose graph contains a degree-3 node:



in which the offending degree-3 node and out-edges is highlighted in red. The single large strongly connected component in this graph has irreducible cycles of sizes 15, 21, and 35 and thus defines a linear ordering with three cycles which is locally checkable with radius 11. The rough shape of this graph is:



in which the number in the cycles represent their size – I’ve not included the actual graph because it is very large and unwieldy. Applying the procedure in step 3 to this construction yields the pattern

$$T' = \{\tilde{A}^n \circ 11 \circ \tilde{B}^m \circ 1111000100111 \circ \tilde{C}^o \mid n, o \geq 1, m \in \mathbb{N}\}$$

where

$$\tilde{A} = 1111000011111100, \quad \tilde{B} = 111100010111100011111,$$

and

$$\tilde{C} = 1111001011111001111110100011101001.$$

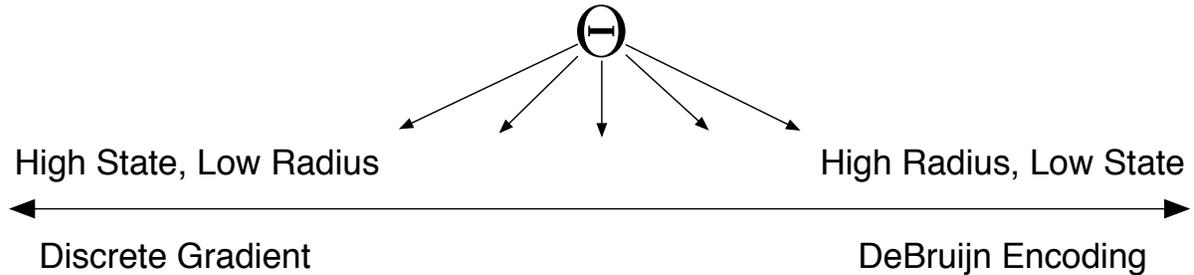
The formula for the decoding function ϕ is determined by mapping the positions in \tilde{A} , \tilde{B} , and \tilde{C} to the corresponding positions in A , B , and C , respectively.

The Radius-State Continuum

The two tradeoff procedures described in the previous sections combine to show that:

Proposition 33 [The Static Radius-State Trade-Off] Any pattern T over m states with $\text{LCR}(T) \leq r$ has a local check Θ such that $\Theta(\mathbb{C})$ is the $r \log_2(m)$ -encoding of a pattern in 2 states that is locally checkable with radius $2\lceil r \log_2(m) \rceil + 1$, and the 2-encoding of a pattern in m^{2r+1} states that is locally checkable with radius 1.

The conceptual import of proposition 33 is to establish that for every pattern there is a continuum along the radius-state tradeoff between high-state/low-radius implementations and high-radius/low-state implementations:



The two ends of the continuum are represented by the solutions of coordinatization, with the Discrete Gradient algorithm as the high-state/low-radius implementation and the DeBruijn Sequence algorithm as the high-radius/low-state implementation.

6.2 The Dynamic Trade-Off

In the previous section, we discussed a “static” radius/state trade-off, one that applied to patterns and their local check schemes, not to the local rules that created those patterns. However, we need to understand the latter as well.

Suppose we’re given a state set S_1 , with size m . Let $T \subset \mathbb{C}_{S_1}$ be a pattern over those m states. Now suppose we add states to the state set, so that $S = S_1 \cup S_2$, where $|S_2| = m' > 0$. Evidently T is also a pattern in \mathbb{C}_S , since $\mathbb{C}_{S_1} \subset \mathbb{C}_{S_2}$; T is simply a pattern that “doesn’t mention” the states in S_2 . Now, suppose we’re given a rule F that solves T in \mathbb{C}_S . This rule is robust to the presence of the extra m' states, in that initial conditions that have arbitrary arrangements of the extra states will eventually be driven to configuration in T that make no mention of such states. On the other hand, F might *make use* of those states, so that even on initial conditions that are in T and make no mention of the extra states, F might temporarily cause one or more such extra states to appear before removing them again in the end. This is precisely what happens in the constructions of Naive Backtracking algorithm F_Θ in §3.2; \triangleleft and \triangleright and Δ_i are just this sort of extra state. The algorithm \tilde{F}_Θ uses even more such states.

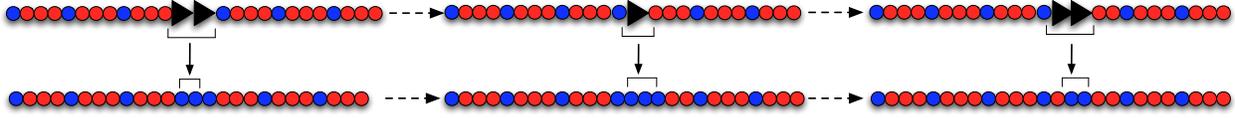
Hence we are led to ask if we can always make a form of dynamic radius/state tradeoff to replace F with a (possibly larger radius) rule F' that is a solution to T over \mathbb{C}_{S_1} without any use of extra states.

6.2.1 Isolated Turing Head Algorithms

There is a simple case in which the extra states can be removed from the Naive Backtracking rule without the need to increase the radius at all: when no backtracking actually occurs, as described in §3.1. In that case, the gradient eq. 3.1 defines a rule using no extra in the first place. However, a construction of that form cannot easily be generalized to solve more general check schemes. This is because check schemes that have multiple nontrivial decision points will require heads to travel in both directions across the system to implement backtracking. This difficulty is the reason why, in the original construction of the Naive Backtracking rule in chapter 3, I introduced the extra state to begin with. To remove them, we will have to more explicitly find an encoding of the extra states and their activity as dynamically coherent substructures of larger radius created from just the original states.

Example 41 Take for example the pattern $T = \{(1000)^n\}$, which has a radius 2 local check scheme Θ . In the Naive backtracking algorithm F_Θ , $\triangleright \circ \triangleright$ acts as a “turing head”, a marker for the moving border of Θ -correctness. As the \triangleright -head progresses, there is an alternation of a state with one \triangleright and a state with two \triangleright ’s.

These two states are bound together, so that a coherent substructure $\triangleright \rightarrow \triangleright \circ \triangleright \rightarrow \triangleright$ propagates through the system. In fact, we can encode the \triangleright head and its dynamic propagation pattern with the sequence $111 \rightarrow 11 \rightarrow 111$ as shown schematically here:



Because neither of the subconfigurations 111 or 11 is present in the final correct pattern, we can extend the definition of the local rule to force the 111 and 11 states to act as “movable coherent border markers”, just like the \triangleright -states originally did.

To see how to make this encoding in general, let Θ be a local check scheme of radius r over the states $1, \dots, m$. We can assume that for each of the $i = 1, \dots, m$, the r -balls $i \circ i \circ \dots \circ i = i^{2r+1}$ are *not* Θ -accepted. That is because if $\Theta[i^{2r+1}] = 1$ for any i , then Θ has a sub-check scheme containing a cycle of size 1, in which case the construction in §3.1 would apply. (In fact, Θ would also be locally patchable, so that the construction \widehat{F}_Θ from §4.2, which also already uses no extra state, could be used as well.) Let O be any left-choice function for Θ , such that if $O(b, j) = 0$ then $j = |R(b)|$ and if $O(b, j) = 1$ then $j = 1$; that is, 0 is always the “last choice” of O , and 1 a “first choice”, when they are choices at all.

Recall how F_Θ^O is defined in §3.2 by **Rules 1-10**. Now, let’s define a new rule F' that produces modified versions of several of the trajectories generated by those rules. Specifically:

1. Replace the trajectory

$$y \circ \triangleright \circ s \rightarrow y \circ \triangleright \circ \triangleright \rightarrow y \circ O(y, 1) \circ \triangleright$$

as generated by **Rules 1-4** by defining F' to generate the trajectory

$$y \circ 10^{2r+2}1 \circ s \rightarrow y \circ 10^{2r+2}11 \rightarrow y \circ 110^{2r+1}11 \rightarrow y \circ O(y, 1) \circ 10^{2r+1}11 \rightarrow y \circ O(y, 1) \circ 10^{2r+2}1.$$

In words, the subconfiguration pattern $10^{2r+2}1 \rightarrow 10^{2r+1}11 \rightarrow 10^{2r+2}1$ acting as an encoding of the right-moving Turing head. **Rules 1-4** will only be triggered when y is Θ -consistent, so since $10^{2r+1}1$ is not Θ -consistent by assumption, the locations of these encodings are non-overlapping and well-defined.

2. Replace the trajectory

$$y \circ \triangleleft \rightarrow y(1 : |y| - 1) \circ \triangleleft \circ \triangleleft$$

generated by **Rule 6** by defining F' to generate the trajectory

$$y \circ 10^{2r+5} \circ 1 \rightarrow y(1 : |y| - 1) \circ 110^{2r+5}1 \rightarrow y(1 : |y| - 1) \circ 110^{2r+4}11 \rightarrow y(1 : |y| - 1) \circ 10^{2r+5}11.$$

In words, the subconfiguration $10^{2r+5}1$ is encoding the left-moving Turing head. Again, since by assumption $10^{2r+5}1$ is not consistent, but y must be for **Rule 6** to be triggered, these encodings are non-overlapping and well-defined.

3. Replace the trajectory

$$y \circ t \circ s \circ \triangleleft \rightarrow y \circ t \circ \Delta_t \circ \triangleleft \rightarrow y \circ y_O^+ \circ \Delta_t \circ \triangleleft \rightarrow y \circ y_O^+ \circ \triangleright \circ \triangleleft$$

generated by **Rules 7-9** by defining F' to produce the trajectory

$$y \circ t \circ 10^{2r+5}1 \rightarrow y \circ t \circ 10^{2r+4} \circ 0 \circ 11 \rightarrow y \circ t \circ 10^{2r+3} \circ t \circ 11 \rightarrow y \circ y_O^+ \circ 10^{2r+3} \circ t \circ 11 \rightarrow y \circ y_O^+ \circ 10^{2r+3}111.$$

In words, the state Δ_t is encoded by $10^{2r+3} \circ t \circ 11$. Because O is assumed to only have 0 as a “last choice” the value of t is never equal to 1 in the above trajectory since **Rule 7** is, by definition, only triggered by values of t that are not “last choices.” Similarly, since O only has 1 as a “first choice,” the value of y_O^+ is never equal to 1. Hence overall, this encoding is well-defined.

4. Define F' to with the direct analogs of **Rules 1,4,5, and 10** that generate the right-moving $10^{2r+3}1$ -head, halt and disappear it when a completion exists, and turn it to the left-moving $10^{2r+5}1$ -head when a completion does not exist.

The trajectories above require F' to have radius $4r + 8$ to accommodate the size of the encoded versions of the turing heads. Hence, we have:

Proposition 34 *All locally checkable patterns T can be solved with a local rule whose radius is at most $4 \cdot \text{LCR}(T) + 8$, using no extra states.*

This improves Theorem 2 in chapter 3 by removing the requirement for extra state, albeit at the expense of adding $2r + 6$ to the radius of the solving rule.

With a small amount more radius, we can remove the condition on the choice function, and so provide encodings for F_{Θ}^O for all O . A similar procedure will lead to an encoding of the Less Naive Backtracking algorithm for single terminus graphs, because the extra states \triangleright , \triangleleft , and \triangle_i are the same as in the Naive algorithm. In all of these cases, the number of extra states is $m + 2$. To encode more states – as for instance required by the Less Naive backtracking algorithm in multi-terminus graphs – we need a slightly more sophisticated procedure.

Suppose that Θ is a local check scheme of radius r over the states $S = \{1, \dots, m\}$, such that 0^{2r+1} is not Θ -accepted. Suppose that F is a local rule of radius $R > 2r + 1$ over S and the extra states $\triangleright_1, \dots, \triangleright_k$. Suppose the only dynamics generated by F are those in the trajectories of the form:

$$y \circ \triangleright_i \circ l \rightarrow y \circ \triangleright_i \circ \triangleright_j \rightarrow y \circ f(j, y) \circ \triangleright_j \quad (6.4)$$

where l is any state, y is a Θ -consistent subconfiguration, j is some function of i and y , and $f(i, y)$ is such that $y \circ f(i, y)$ is Θ -consistent whenever y is. In words, this is the dynamics of a right-moving turing head with k internal states. Now, suppose we wish to encode these dynamics using only the m states in S . We can do this with a rule F' of radius

$$R' = R + 4r + 4 + \lceil \log_m(k + 2r + 1) \rceil + 10$$

in the following way. For each $j \in \{1, \dots, k\}$ we can assign j to a unique length- $\lceil \log_m(k + 2r + 1) \rceil$ m -ary sequence, s_j , containing no instance of 0^{2r+1} as a subsequence. For each Θ -consistent y , and all states l , consider the following trajectory:

$$\begin{aligned} & y \circ 110^{2r+2}1 \circ s_i s_i \circ 10^{2r+2}11 \circ l \rightarrow y \circ 110^{2r+2}1 \circ s_i s_i \circ 10^{2r+2}111 \rightarrow \\ & y \circ 110^{2r+2}1 \circ s_i s_i \circ 110^{2r+1}111 \rightarrow y \circ 110^{2r+2}1 \circ s_i s_i (1) s_j (1) s_i (3 : |s|) \circ 110^{2r+1}111 \rightarrow \\ & y \circ 110^{2r+2}1 \circ s_i s_i (1) s_j (1 : 2) s_i (4 : |s|) \circ 110^{2r+1}111 \rightarrow \dots \rightarrow y \circ 110^{2r+2}1 \circ s_i s_i (1) s_j \circ 10^{2r+1}111 \rightarrow \\ & y \circ 110^{2r+2}1 \circ s_i s_i (1) s_j \circ 10^{2r+2}11 \rightarrow y \circ 1110^{2r+1}1 \circ s_i s_i (1) s_j \circ 10^{2r+2}11 \rightarrow \\ & y \circ 1110^{2r+1}1 \circ s_i s_j (|s|) s_j \circ 10^{2r+2}11 \rightarrow y \circ 1110^{2r+1}1 \circ s_i (1 : |s| - 1) s_j (|s| - 1 : |s|) s_j \circ 10^{2r+2}11 \rightarrow \dots \\ & \rightarrow y \circ 1110^{2r+1}1 \circ s_i (1) s_j s_j \circ 10^{2r+2}11 \rightarrow y \circ 1110^{2r+1}11 \circ s_j s_j \circ 10^{2r+2}11 \rightarrow \\ & y \circ f(j, y) \circ 110^{2r+1}11 \circ s_j s_j \circ 10^{2r+2}11 \rightarrow y \circ f(j, y) \circ 110^{2r+2}1 \circ s_j s_j \circ 10^{2r+2}11. \end{aligned} \quad (6.5)$$

For the first six steps, the system is in “Stage 1” – translating the right-most of the two copies of s_i one step to the right, and in the process transitioning it to s_j . In the next six steps, the system is in “Stage 2” – translating the left of the two copies of the s_i , making it too into a s_j . In the last two steps, the system is in “Stage 3” – writing the $f(j, y)$ state at its left end. The reason that two copies of the s_i 's are included is so that during the process of the head's being translated one step to the right, the system knows where the left-most boundary of each copy of s_i is, by comparing to the other copy. This is the analog of the two-step Turing head propagation discussed in §3.2. Which stage the system is in is signaled by the number of 1s at the borders of the 0^{2r+1} strings. And, since 0^{2r+1} is illegal both in y and the s_j s, no two trajectories of this form can be overlapping. By inspection, one can see that at each stage, every ball of radius R' has enough information to uniquely determine the indicated transitions. Hence a rule of radius R' can replace trajectory 6.4 with 6.5.

A similar procedure can be described for a left-moving turing head, and for turing head birth, death, and reversal operations. Hence, any rule that can be described by such operations – including both the Naive

and Less Naive Backtracking rule in the non-locally-patchable case – can be translated into an equivalent rule using no extra state at the expense of adding $4r + 4 + \lceil \log_m(k + 2r + 1) \rceil + 10$ to the rule's radius (where k is the total number internal states of the head). These “Isolated Turing head algorithms” have enough structure to allow for a systematic dynamic radius state tradeoff.

The radius-state tradeoff defined above works only for a specific class of algorithms. It is desirable to have a generic procedure for making such tradeoffs for any local rule, as we did for the static tradeoff. This would involve finding a generally-applicable process for encoding the dynamics of one local rule in the operation of another, with different states and radius. Even *defining* the notion of dynamic encoding properly, much less finding such encodings algorithmically, turns out to be difficult. See appendix §E for a discussion of the possible approaches.

Chapter 7

Pattern Space Analysis

By associating every locally checkable pattern with radius r and m states with an induced subgraph of the larger graph $\mathbb{D}(r, m)$, proposition 10 in §2.3 establishes that $\mathbb{D}(r, m)$ is the “ambient space” of locally checkable (and therefore locally solvable) patterns. Subsequent chapters made much use of the graph structures associated with patterns to aid analysis and constructions. Most of the graph properties mentioned in these discussions were very general. Apart from the repeated use of the bound on graph degree by the number m of states, and the (loose) bound of m^{2r+1} on the maximal size of irreducible cycles, we made use of no properties that would distinguish local check scheme graphs from any other finite directed graphs.

However, because the ambient space $\mathbb{D}(r, m)$ itself is highly structured (as we shall see), the set of graphs that can arise from local check schemes with a give choice of the parameters r and m is constrained. We might therefore expect that important but non-obvious features of locally checkable patterns, and the local rules that can form them, could be learned by studying $\mathbb{D}(r, m)$ in more detail.

In this chapter, I develop an abstract mathematical theory to characterize $\mathbb{D}(r, m)$. In §7.1, I reduce questions about the structure of $\mathbb{D}(r, m)$ to questions about the DeBruijn graphs, a well-known construction in computer science. In §7.2, I explicate some helpful but well-known simple properties of the DeBruijn graphs. In §7.3, I construct a concrete realization of the DeBruijn graphs that is useful for developing some of the DeBruijn graphs’ more sophisticated properties. In §7.7 I prove a useful fact regarding geodesics in the DeBruijn graphs. In §7.5, I define two abstract operations on the cycles of $\mathbb{D}(r, m)$ – a cycle embedding operation and a cycle sum operation – that preserve the geodesics, and that allow us to place a graded algebraic structure on the DeBruijn graphs that respects geometry. In 7.6, I use this abstract structure to elucidate aspects of the DeBruijn graphs’ subgraph topology and geometry.

Finally, in §7.7, I show how the mathematical results of the preceding sections can be interpreted as having practical consequences for the design and analysis of local rules.

7.1 $\mathbb{D}(r, m)$ and the DeBruijn Graph

For any positive integer m , let $[m] = \{0, \dots, m - 1\}$. Thus, $[m]^i = [m] \times [m] \times \dots [m]$, taken i times, is the set of m -ary sequences of length i . Define the DeBruijn graph $DB(n, m)$ to be the directed edge-labeled graph

$$(V_{n,m}, E_{n,m}),$$

whose nodes $V_{n,m}$ are $[m]^n$, the m -ary sequences of length n , and whose edges are

$$E_{n,m} = \{(v_i, v_j, k) \in V(n, m) \times V(n, m) \times [m] \mid v_j = (v_i(2), v_i(3), \dots, v_i(n), k)\}.$$

In words, the edges connect the vertex corresponding to m -ary sequence v_i to that for v_j , with an edge labeled by k , whenever the sequence v_j is obtained from v_i by removing the first element $v_i(1)$ and appending k to the end.

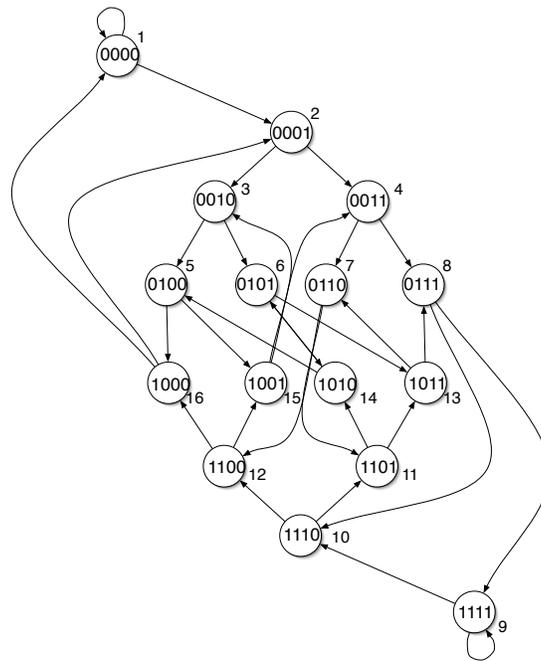


Figure 7.3: The DeBruijn Graph $DB(4, 2)$.

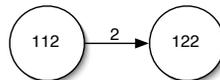


Figure 7.1: Two nodes and a valid edge between them in the DeBruijn graph $DB(3, 2)$.

$DB(n, m)$ has m^n nodes, since there are that many m -sequences of length n . It has m self-loops, one for each sequence i^n :



Figure 7.2: A self-loop in $DB(3, 2)$.

$DB(n, m)$ is a strongly connected graph, meaning that any two nodes can be connected by a directed path. Each node in has in-degree and out-degree both equal to m , counting self-loops:

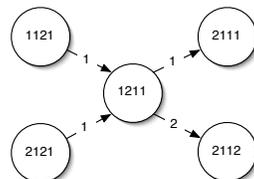


Figure 7.4: The typical in-and-out structure of $DB(4, 2)$ (valid except for self loops).

A key feature about the DeBruijn graph is that paths in it correspond to sequences. Suppose we're given a path of length N :

$$P = (p_1, p_2, \dots, p_N)$$

where each p_i is a node in $DB(n, m)$, and (p_i, p_{i+1}) is a valid edge for all i . By definition of the nodes of the DeBruijn graph, each node p_i corresponds to an m -ary sequence of length n . Just like for $ID(r, m)$, we can stitch these sequences together to make an m -ary sequence S_P . To do this, take the first n elements of S_P to be the length- n sequence corresponding to p_1 . Then, since p_2 is linked to from p_1 , it must be the case that the first $n - 1$ elements of p_2 coincide with the last $n - 1$ elements of p_1 . Hence, just add on the last element

of p_2 to make a sequence of length $n + 1$. Since p_3 is linked to from p_2 , the first $n - 1$ elements of p_3 coincide with the last $n - 1$ elements of p_2 . Just add on the last element of p_3 , extending to length $n + 2$. This can be repeated, until the last element of p_N has been used, making a sequence of length $N + n - 1$:

$$S_P = p_1 \circ p_2(n) \circ p_3(n) \circ \dots \circ p_N(n)$$

where $p_i(n)$ indicates the n -th (and therefore last) element of the m -ary sequence associated with p_i .

Example 42 In figure 7.3, the path given by node sequence (3,6,13,8,10,12,15) corresponds to the binary sequence 0010111001.

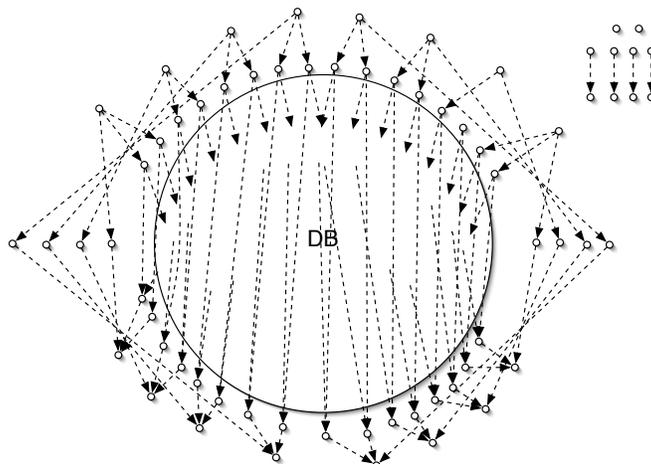
For m -ary sequences of length K , this construction can evidently be reversed as long as $K > n - 1$. Hence:

Proposition 35 m -ary sequences of length K are in 1-1 correspondence with paths of length $K - n + 1$ in $DB(n, m)$.

The reason I've introduced the DeBruijn graph is that $DB(2r + 1, m)$ captures most of the relevant structure in $\mathbb{D}(r, m)$. To see how, it is useful to separate the question in terms of the isomorphism classes of balls described in appendix §A.1. The details of the computation are described in appendix §G.1, but the basic answer is that $\mathbb{D}(r, m)$:

- Has one strongly connected component isomorphic to $DB(2r + 1, m)$ corresponding to the m^{2r+1} central balls.
- Has $1 + \sum_{i=1}^r m^i = (m^{r+1} - 1)/(m - 1)$ connected components, one for the main component and the others for each of the small balls.
- All nodes in the strongly connected component have indegree and out-degree m .
- All nonterminal nodes corresponding to left-end balls have indegree 1 and out-degree $m + 1$.
- All nonterminal nodes corresponding to right-end balls have indegree $m + 1$ and outdegree 1.
- All nodes corresponding to small central or very small balls are belong to a single weakly connected component, having indegree and outdegree 1.
- Initial nodes (with indegree 0) correspond to left-most balls with the \star at position 1, and terminal nodes to right-most balls b with the \star at position $|b|$.

Pictorially, $\mathbb{D}(r, m)$ looks like:



The effect of this is to break the classification and counting problems into two parts: (A) the subgraph $G(\Theta) \setminus DB(2r(\Theta) + 1, m)$ is the “small size” piece corresponding to configurations with size comparable to the information radius of the local check. This part has simple structure since $\mathcal{D}(r, m) \setminus DB(2r + 1, m)$ is (essentially) the trivial graph, and the configurations generated by it are just the elements themselves i.e. $\Theta(\mathbb{C}_{i \leq 2r(\Theta), m}) = G(\Theta) \setminus DB(2r(\Theta) + 1, m)$. (B) The other part part, $G(\Theta) \cap DB(2r(\Theta) + 1, m)$ generates larger configurations corresponding to paths, and inherits its more complex structure from DB . In other words, understanding $\mathbb{D}(r, m)$ boils down to classifying and counting the subgraphs of the DeBruijn graphs.

7.2 Simple Properties of DeBruijn Graphs

Before characterizing the DeBruijn graph's more sophisticated abstract structure, it is useful first to organize some of its simplest well-known properties (good sources on the DeBruijn graph are [7, 11]). The three simplest properties of $DB(n, m)$, which I've implicitly used through this thesis, are that:

- $DB(n, m)$ has n^m nodes – corresponding to the m^n m -ary sequences of length n .
- DB has m self-loops – corresponding to the sequences i^n , for each i .
- The in-degree and out-degree is m at each node – Given an m -ary sequence x of length n , then for each integer $i \in [m]$, the sequences $y_i = x(2 : n) \circ i$ are the m sequences of length n to which x is connected via out-edges; similarly, $z_i = i \circ x(1 : n - 1)$ represent the m nodes connected to x via incident edges.

The most well-known nontrivial property of the DeBruijn graph is that it has a maximal length-cycle:

Proposition 36 *The DeBruijn graph admits a Hamiltonian circuit.*

Proof: The nodes of $DB(n, m)$ can be put into correspondence with the edges of $DB(n - 1, m)$ via the mapping:

$$v_i \mapsto (v_i(1 : n - 1), v_i(2 : n), v_i(n)).$$

Under this mapping, consecutive nodes, connected via an edge in $DB(n, m)$, go to consecutive edges, connected by a node in $DB(n - 1, m)$. Hamiltonian cycles in $DB(n, m)$ therefore correspond to Eulerian cycles in $DB(n - 1, m)$. A well-known theorem of Euler says that directed graphs whose indegree at each node equals the out-degree at that node must possess an eulerian cycle. All nodes in $DB(n - 1, m)$ have in-degree and out-degree equal to $n - 1$, so $DB(n - 1, m)$ has an eulerian cycle, and thus $DB(n, m)$ a Hamiltonian circuit. ■

The state sequences S_H corresponding to hamiltonian cycles H are the DeBruijn sequences which were useful in chapter 6. The short paths in $DB(n, m)$ also have some simple nice properties:

Proposition 37 *Let x, y be any two nodes in $DB(n, m)$.*

- *There is a unique acyclic path between x and y of length n or less.*
- *DB can be partitioned into a union of disjoint cycles of size n or less.*
- *For each $k \leq n$, there is at least one cycle of length k .*
- *In fact, for $k \leq n$, the number of distinct cycles of length k in $DB(n, m)$ is*

$$\frac{(-1)^{\mu(k)}}{k} \sum_{d|k} (-1)^{\mu(d)} m^d$$

where the sum is taken over divisors d of k and $\mu(j)$ is 0 when j has an even number of factors and 1 if odd.

Proof: First, let's establish a little notation: As we saw in §7.1, any path P in $DB(n, m)$ also corresponds naturally to an m -ary sequence, simply by reading off the labels of the edges along the path, with the length- n sequence associated with the first node of the path prepended. Denote this sequence $seq(P)$. For a single node, let s_x be shorthand for $seq(x)$, the length- n m -ary sequence labeling the node.

(i) Let $p(x, y)$ be the path $p = (p(1), p(2), \dots, p(n))$ defined by $s(p(i)) = z[i : i + n]$ where $z = s(x) \circ s(y)$. $p(x, y)$ is a path from x to y of length n . On the other hand, suppose there are two paths p_1 and p_2 from x to y both of length n or less. If they have the same length $i \leq n$, then $y = x(i : n) \circ seq(p_1) = x(i : n) \circ seq(p_2)$. But then $p_1 = p_2$. On the other hand, suppose wlog that p_1 is the shorter of the two paths. Then s_y is periodic with period dividing $|p_2| - |p_1|$, and p_2 contains at least one copy of the minimal cycle underlying s_y .

(ii) Notice that, in the notation from (i), $p(x, x)$ is a loop of length n . Denote by $\tilde{p}(x, x)$ the minimal cycle underlying $p(x, x)$. The set $\mathcal{P} = \{\tilde{p}(x, x) | x \in DB(n, m)\}$ is a set of disjoint cycles of length at most n that partition $DB(n, m)$. The length of $\tilde{p}(x, x)$ is a divisor of n .

(iii) For each $i \in \{0, \dots, n-1\}$, consider the length $i+1$ sequence $s^i = 1^i 0$, and for each $k \leq i$ define the length- n sequence $t_k^i(j) = s^i(\overline{j+k})$ where the bar indicates residue modulo $i+1$. The set of nodes x_k^i such that $s_{x_k^i} = t_k^i$ comprise a cycle X^i of length i . (Notice that X^i is the cycle $p(x, x)$ for some $x \in X^i$ if and only if i is a divisor of $n-1$).

(iv) Cycles of length $k < n$ in $DB(n, m)$ correspond to m -ary sequences with period precisely k , modding out for circular shifts. Hence, the number of cycles of length k is equal to $1/k$ times the number of such sequences, which we denote $S(k)$. The number of sequences of period precisely k is the number of all length- k m -ary sequences, less the number of sequences with period d , for each divisor d of k strictly less than k . Hence $S(k) = m^k - \sum_{d|k, d \neq k} S(d)$, which yields the result. ■

In words, this result says: short cycles (of length $\leq n$) are ubiquitous, spaced out from each other evenly, and easily counted. It also has the result that geodesics are unique, so for any two nodes x, y in $DB(n, m)$ let $sp(x, y)$ denote the (unique) geodesic from x to y .

7.3 A General Realization of DB

To develop intuition for more complex properties, it is useful to have a concrete realization of DB . $DB(n, m)$ can be realized in a variety of ways. The way I have found that is most useful is to describe it as a set of m m -ary trees, with connections between the leaves of each tree and the nodes of others. There are other realizations, see e.g. [10] or the Wikipedia site on DeBruijn graphs.

The first step in constructing this realization is to partition the vertices into the sets V_i of all m -ary sequences of length n whose first element is i , i.e. $V_i = \{ix | x \in [m]^{n-1}\}$. There are m such sets V_i , each of which has size $m^n/m = m^{n-1}$. The graph that $DB(n, m)$ induces on V_i , denoted G_i , is obtained from the complete m -ary tree on n levels by removing one vertex at level 2, and the entire subtree below it, and replacing it with a single self-loop on the root vertex. The root vertex in G_i corresponds to the sequence i^n , the level 2 vertices to the sequences $i^{n-1}j$ with $j \neq i$, and so on. Each node $x = iy$ in G_i has m children (with the exception of the root, which has $m-1$); these are the sequences $x(2:n)l$ for $l \in [m]$. (This notation means that we've taken x , cut off the first element to leave $x(2:n)$, and then appended l to the end. A complete m -ary tree with k levels has

$$1 + m + \dots + m^{k-1} = \frac{m^k - 1}{m - 1}$$

nodes, so

$$|G_i| = \frac{m^n - 1}{m - 1} - \frac{m^{n-1} - 1}{m - 1} = \frac{m^n - m^{n-1}}{m - 1} = m^{n-1},$$

which checks out with computation above. Each G_i has $(m-1)m^{n-2}$ leaves, corresponding to sequences ijz where $j \neq i$ and $z \in [m]^{n-2}$. All this is depicted in Fig. 7.5.

Now, assign the vertices in $DB(n, m)$ numbers between 1 and m^n , starting in G_1 , and numbering the nodes of each tree G_i , before moving on to G_{i+1} . Within each G_i , enumerate the nodes in a breadth-first fashion, counting all the nodes a given level before moving on to the next. Number the children of the node iy taking yi first, then $y\bar{i}+1$, then $y\bar{i}+2$, ... getting to $y\bar{i}+m-1$ last, where the bar denotes residue mod m (shifted by 1). This scheme gives nodes in G_i numbers $(i-1)m^{n-1} + j$, for $j \in \{1, \dots, m^{n-1}\}$; the root of G_i gets assigned number $(i-1)m^{n-1} + 1$; and the leaves are $(i-1)m^{n-1} + m^{n-2} + k$, for $k \in \{1, \dots, (m-1)m^{n-2}\}$. Moreover, in this scheme it is always the first node and corresponding sub-tree at level 2 that would be removed from a complete m -ary tree to produce the G_i s. This is depicted in Fig. 7.6.

If we denote the number assigned to the m -ary sequence $a_1 a_2 \dots a_n$ by $N(a_1, \dots, a_n)$, then this numbering satisfies

$$N(i, a_2, \dots, a_n) = (i-1)m^{n-1} + N(1, \sigma^{-i+1}(a_2), \sigma^{-i+1}(a_3), \dots, \sigma^{-i+1}(a_n)), \quad (7.1)$$

where $\sigma(\cdot)$ is the shift permutation on $[m] = \{1, \dots, m\}$ taking i to $\overline{i+1}$, residue modulo m (shifted by 1). More explicitly, the sequence $(a_1, \dots, a_n) \in [m]^n$ is assigned the number

$$1 - \frac{m^n - 1}{m - 1} + \sum_{i=1}^{n-1} b_i m^{n-i}$$

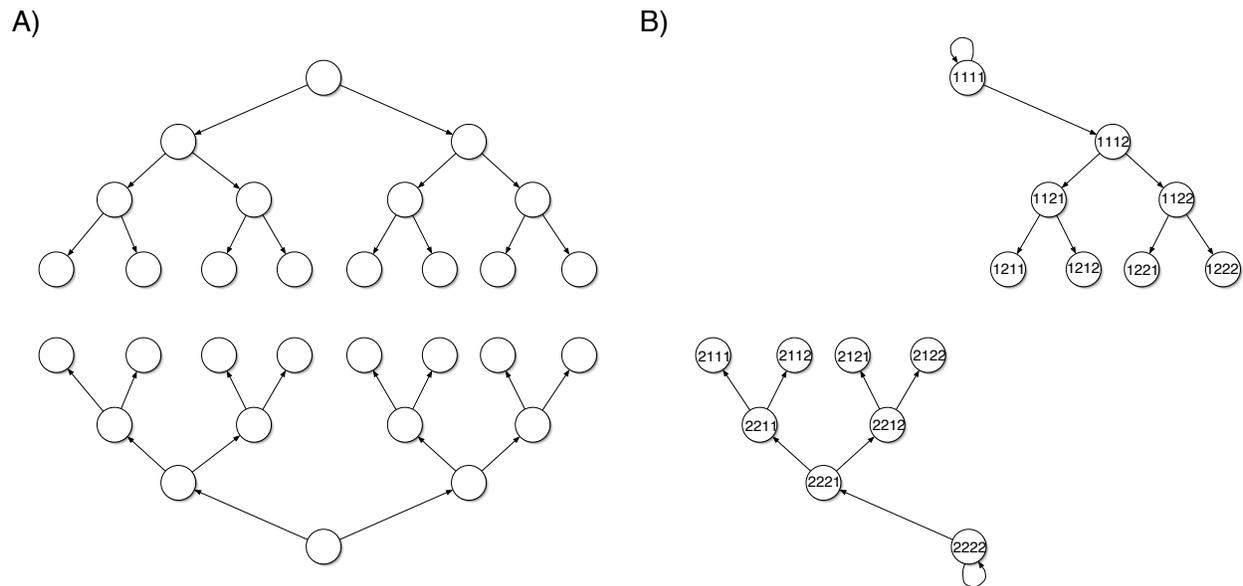


Figure 7.5: Making $DB(4,2)$, Step 1: We start with A) two binary trees with four levels. Then, B) we excise the first level-2 node of each, and its corresponding subtree, and replace it with a self-loop. The resulting trees have 8 nodes each, and with root nodes corresponding to 1111 and 2222. Then ... see figure 7.6.

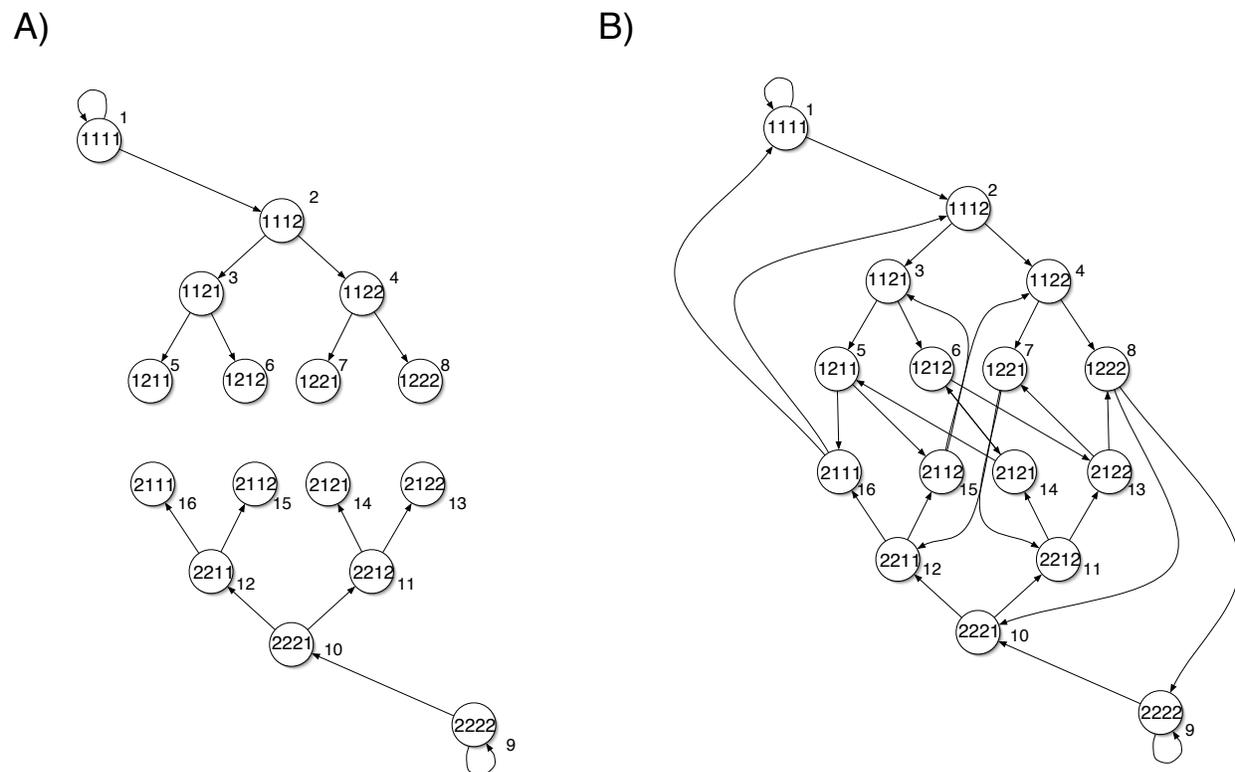


Figure 7.6: Making $DB(4,2)$, Step 2: Next, A) number the nodes in a sequence fashion and B) connect nodes as according to (7.2).

where $b_1 = a_1$ and for $i \geq 2$, $b_i = \overline{a_i - a_1} = \sigma^{-a_1}(a_i)$. In terms of this numbering scheme, the edges of $DB(n, m)$ are

$$\left(1 - \frac{m^n - 1}{m - 1} + \sum_{i=1}^{n-1} a_i m^{n-i}, 1 - \frac{m^n - 1}{m - 1} + \sum_{i=2}^{n-1} b_i m^{n-i+1} + l \right), \quad (7.2)$$

where $b_2 = \overline{a_1 - a_2}$, $b_i = a_i - a_2$ for $i \geq 3$, and l ranges in $[m] = \{1, \dots, m\}$. For non-leaf nodes – in the “main tree body” T_i – these edges are simply those of the (mostly) m -ary tree within each G_i . For leaf nodes, the l -th consecutive group L_i^l of m^{n-2} leaves in G_i is connected to consecutive nodes in G_{i+1} .¹

This realization relates to the graph $\mathbb{D}(r, m)$ described in the previous section very easily: any vertex in $\mathbb{D}(r, m)$ corresponding to a left-end ball of size $2r$ connects to vertices $jm + i$ for all $i \in [m]$ and some fixed $j \in [m^{n-1}]$; and any right-end ball of size $2r$ is connected to from $jm^{n-1} + i$ for some fixed $i \in [m^{n-1}]$ and all $j \in [m]$.

7.4 A Useful Fact

Given a directed graph G and $x \in G$, let $P(x) = \{y \in V(G) | (y, x) \in E(G)\}$ denote the set of *parents* of x and $C(x) = \{y \in V(G) | (x, y) \in E(G)\}$ denote the set of *children* of x . Let $P^i(x) = \bigcup_{y \in P^{i-1}(x)} P(y)$ and define $C^i(x)$ analogously. The following fact about the behavior of $DB(n, m)$ with respect to child/parent relations in $DB(n, m)$ is very useful:

Proposition 38 For all $x, y, z, w \in DB$,

- if $C^l(x) \cap C^l(y) \neq \emptyset$ then $C^l(x) = C^l(y)$, and
- if $\text{dist}(x, z) = \text{dist}(x, w) = \text{dist}(y, z) = \text{dist}(y, w)$, then $|\text{sp}(x, z) \cap \text{sp}(x, w)| = |\text{sp}(y, z) \cap \text{sp}(y, w)|$.

Proof: This result is almost obvious from the tree realization. Let’s focus on a little piece of the tree, as shown here: Now, (i) Suppose there are paths from x to z and y to z of length l . That is to say there are m -ary

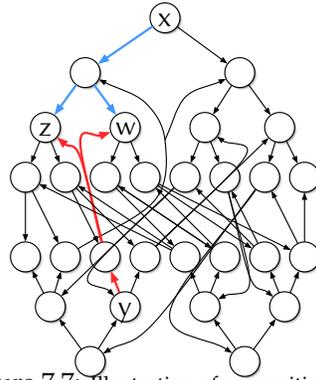


Figure 7.7: Illustration of proposition 38.

sequences l_1 and l_2 of length l such that $s_x \circ l_1(l+1 : n+l) = s_y \circ l_2(l+1 : n+l) = s_z$. Hence $x(l+1 : n) = y(l+1 : n)$. Now, suppose $w \in C^l(y)$. Then there is l_3 of length l such that $s_y \circ l_3(l+1 : n+1) = s_w$; but then $x(l+1 : n) \circ l_3 = s_w$ and $w \in C^l(x)$. (ii) Suppose z, w have $\text{dist}(x, z) = \text{dist}(x, w) = L$. Then $z, w \in C^L(x)$ and $z, w \notin C^j(x)$ for $j < L$. Let $d = |\text{sp}(x, z) \cap \text{sp}(x, w)|$. Then by the uniqueness of geodesics $\text{sp}(x, z)(j) = \text{sp}(x, w)(j)$ for all $j \leq d$ and $\text{sp}(x, z)(d+1) \neq \text{sp}(x, w)(d+1)$. Let $a = \text{sp}(x, z)(d+1)$. Then $\text{dist}(a, z) = \text{dist}(a, w) = l - d$. On the other hand, if $\text{dist}(y, z) = \text{dist}(y, w) = L$, then let $d' = |\text{sp}(y, z) \cap \text{sp}(y, w)|$ and $\text{sp}(y, z)(j) = \text{sp}(y, w)(j)$ for all $j \leq d'$ and

¹Equations 7.1 and 7.2 may have significance for building a general geometrical theory of graphs. Specifically, I imagine writing general graphs as disjoint unions of trees (or “almost”-trees), then connecting up those trees by edges between their leaves. In this way, we would think of the trees as “trivial locally flat patches” which would be identified along their boundaries (i.e. the leaves) via boundary maps like eqs. 7.1 and 7.2. The structure of these boundary maps, which describe the global violations of the flat structure of the trees – introduces non-trivial geometric curvature into the graph.

$sp(x, z)(d' + 1) \neq sp(x, w)(d' + 1)$. Let $b = sp(y, z)(d' + 1)$. Then $dist(b, z) = dist(b, w) = l - d'$. Applying part (i) with $x = a, y = b$ forces $l - d = l - d'$ and thus $d = d'$. ■

For a cycle C and x a node of C , let $c(x, C)$ denote the node that follows immediately after x in C (x 's unique "child"), and $p(x, C)$ denote the node immediately previous to x in C (x 's "parent"). Let $c^i(x, C) = c(c^{i-1}(x, C), C)$ and $p^i(x, C) = p(p^{i-1}(x, C), C)$. For two nodes x, y in any simple cycle C (i.e. one in which each node appears once), let $C(x : y)$ denote the unique path from x to y in C .

Corollary 4 *Let C_1 and C_2 be two cycles in $DB(n, m)$, and let $a \in C_1$ and $b \in C_2$. If there is a path α from a to b , then there is a path β of length $|\alpha|$ from $p^{|\alpha|}(b, C_2)$ to $c^{|\alpha|}(a, C_1)$.*

Proof: Apply the first part of proposition 38 with $x = p^{|\alpha|}(y, C_2), w = c^{|\alpha|}(x, C_1), y = a$ and $z = b$. ■

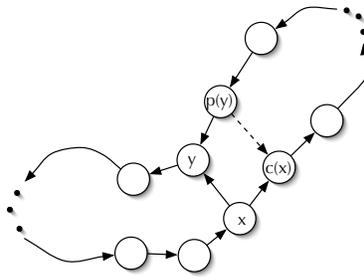
Corollary 5 *Let C_1 and C_2 be two cycles in $DB(n, m)$. Then $dist(C_1, C_2) = dist(C_2, C_1)$.*

Proof: By definition, $dist(C_1, C_2) = \min\{dist(a, b) | a \in C_1, b \in C_2\}$. Suppose this minimum is achieved by nodes a^*, b^* . Let $\alpha = sp(a^*, b^*)$, and apply corollary 4. ■

In general, given two disjoint cycles C_1 and C_2 , the pair of points $a^* \in C_1, b^* \in C_2$ are *points of closest approach* if $|sp(a^*, b^*)| = dist(C_1, C_2)$.

Corollary 6 *Any cycle in DB can be written as a union of disjoint irreducible cycles.*

Proof: Let C be a cycle in DB . If C is reducible, then there are by definition $x, y \in C$ such that (x, y) is a valid edge in DB . But then by proposition 38 i), $(p(y, C), c(x, C))$ is a valid edge. Thus $C_1 = C(y : x) \circ y$ is a cycle, as is $C_2 = C(c(x, C), p(y, C)) \circ c(x, C)$. In fact, $C_1 \cap C_2 = \emptyset$ and $C_1 \cup C_2 = C$. This is obvious from the picture:



If both C_1 and C_2 are now irreducible, we're done; else, the same procedure may be repeated. ■

By similar reasoning, since any strongly connected component in a directed graph is either a single node or a union of cycles,

Corollary 7 *Any strongly connected subgraph of DB is either a single node or a union of connected irreducible cycles.*

7.5 Two Operations

The properties seen so far are a bit *ad hoc*. I will now go a bit more abstract and define two operations that make the loops of the DeBruijn graphs $DB(n, m)$ into a graded semigroup, for fixed m and variable n .

Cycle Embeddings

The proof that DeBruijn graphs have Hamiltonian cycles given above makes use of a correspondence between edges of $DB(n - 1, m)$ and nodes of $DB(n, m)$. An edge of $DB(n - 1, m)$ is a pair $(v_i, v_j) \in V(n - 1, m) \times V(n - 1, m)$. Each node v_i corresponds to an m -ary sequence of length $n - 1$, and the edge being valid implies $v_j(1 : n - 2) = v_i(2 : n - 1)$. (We'll abuse notation and refer to nodes and m -ary sequences

interchangeably.) Hence, the overlap sequence $v_i \circ v_j(n-1)$ corresponds to a length- n m -ary sequence, and thus a node in $DB(n, m)$. Thus, we define a map $\gamma : E(n-1, m) \rightarrow V(n, m)$ by

$$\gamma((v_i, v_j)) = v_i \circ v_j(n-1).$$

This map is bijective, and takes sequential edges to sequential nodes.

The map γ induces a map on cycles: If C is a cycle $(c_1, c_2, \dots, c_N, c_1)$ of length N in $DB(n-1, m)$, then $\gamma(C) = (\gamma(c_1, c_2), \gamma(c_2, c_3), \dots, \gamma(c_n, c_1), \gamma(c_1, c_2))$ defines a length- n cycle in $DB(n, m)$. Similarly, given a length- n cycle C in $DB(n, m)$, $\gamma^{-1}(C)$ is a length- n loop in $DB(n-1, m)$ in which each edges are not duplicated (although nodes implicated by the egdes might appear twice).

γ could equally well have been defined by setting for all $x \in C$, $\gamma_C(x) = \gamma((x, c(x, C))) = x \circ c(x, C)(|x|)$. (One can define thereby an embedding of $DB(n-1, m)$ in $DB(n, m)$ by taking C to be a Hamiltonian cycle.) We can iterate γ , so that applied to a cycle C in $DB(n, m)$, $\gamma^k(C)$ is a cycle in $DB(n+k, m)$. By inspection it is clear that

$$\gamma_C^k(x) = x \circ c(x, C)(|x|) \circ c^2(x, C)(|x|) \dots \circ c^k(x, C)(|x|) = Seq(C(x : c^k(x, C)))$$

and

$$\gamma_C^{-l}(x) = x(1 : |x| - l).$$

Given a cycle C in DB , let $s(C)$ be the m -ary sequence of length $|C|$ generated by reading off the labels of the edges in C – this is defined up to a circular shift; for a basepoint $x \in C$, let $s(C, x)$ denote the sequence starting at $(x, c(x, C))$ and ending at $(p(x, C), x)$. The sequence $s(C)$ defines a unique cycle in DB . Another way to characterize γ is that

$$s(\gamma(C)) = s(C)$$

for all cycles C .

The key feature of γ is that it “flattens” the local geometry of $DB(n, m)$.

Definition 34 [Flatness] A graph $G \subset DB$ is k -flat at $x \in G$ if $B_k(x, G) = B_k(x, DB) \cap G$, where $B_k(x, G) = \{y \in G | d_G(x, y) \leq k\}$ and $B_k(x, DB) = \{y \in DB | d_{DB}(x, y) \leq k\}$.

Example 43 It follows immediately from the definition that: A cycle in DB is irreducible iff it is 1-flat; and a union of disconnected irreducible cycles is equivalent to a union of 1-flat cycles whose mutual distances are at least 2.

It is not hard to see that:

Proposition 39 If cycle C (considered as a subgraph) is k -flat at x , then $\gamma(C)$ is $k+1$ -flat at $\gamma_C(x)$. Furthermore, all k -flat cycles in $DB(n, m)$ arise as $\gamma(C)$ of a $k-1$ -flat cycle in $DB(n-1, m)$, for $k \geq 1$.

Example 44 Consider the Hamiltonian cycle H in $DB(4, 2)$ given by nodes 1, 2, 4, 8, 9, 10, 11, 13, 7, 12, 15, 3, 6, 14, 5, 16. In $DB(5, 2)$, $\gamma(H)$ is an irreducible cycle of length 16. See figure 7.8.

This proposition evidences a nice relationship. We already know that Eulerian Cycles in $DB(n-1, m)$ are equivalent to Hamiltonian Cycles in $DB(n, m)$. Proposition 39 makes this equivalence part of a sequence:

$$\begin{aligned} \text{Eulerian Cycles in } DB(n-1, m) &\leftrightarrow \text{Hamiltonian Cycles in } DB(n, m) \\ &\leftrightarrow \text{length-}m^n \text{ Irreducible Cycles in } DB(n+1, m) \\ &\leftrightarrow \text{length-}m^n \text{ 2-flat cycles in } DB(n+2, m) \dots \end{aligned}$$

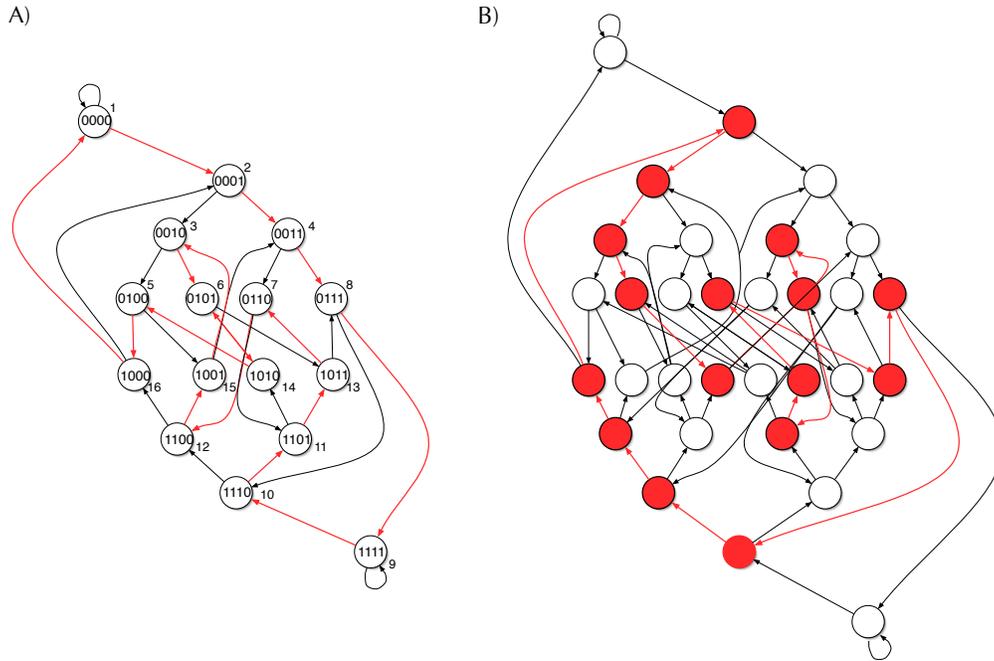


Figure 7.8: A) A hamiltonian path H in $DB(4,2)$, marked in red. B) Irreducible cycle $\gamma(H)$ in $DB(5,2)$.

Cycle Addition

Given two loops C_1, C_2 and $x \in C_1$ and $y \in C_2$, define the *geodesic sum* operation \odot_y^x so that $C_1 \odot_y^x C_2$ is the unique loop C of length $|C_1| + |C_2|$ whose sequence is

$$s(C_1, c(x, C_1)) \circ s(C_2, c(y, C_2))$$

defined up to circular shifts. I use “loops” instead of cycles because the objects need not touch each node only once. Since for all cycles $s(\gamma(C)) = s(C)$, γ commutes with the geodesic sum, i.e.

$$\gamma(C_1 \odot_y^x C_2) = \gamma(C_1) \odot_{\gamma(y)}^{\gamma(x)} \gamma(C_2)$$

where $\gamma(x)$ means $\gamma_{C_1}(x)$ and $\gamma(y)$ means $\gamma_{C_2}(y)$.

The key point (and the reason I call it geodesic sum) is that:

Proposition 40 Suppose C_1 and C_2 are k -flat cycles such that $d = \text{dist}(C_1, C_2) \geq k + 1$. Let $x \in C_1$, $y \in C_2$ be points of closest approach, and let $y' = p^d(y, C_2)$. Then

$$C_1 \odot_{y'}^x C_2$$

is a k -flat cycle.

Example 45 Let C_1 be the cycle associated with the repeat pattern T_{1110} and C_2 the cycle associated with T_{1000} . These patterns are 3-flat 4-cycles in $DB(5,2)$. They are distance 3 away from each other. Their sum (at one of their points of closest approach) produces a 2-flat 8-cycle, the repeat pattern $T_{110100001}$. (See figure 7.9)

The proof of prop. 40 is given in appendix §G.2.

Corollary 8 Let C_1, C_2, \dots, C_N be a set of k -flat cycles in $DB(n, m)$ whose mutual distances are greater than k . Then there is a k -flat cycle in $DB(n, m)$ of length $\sum_i |C_i|$.

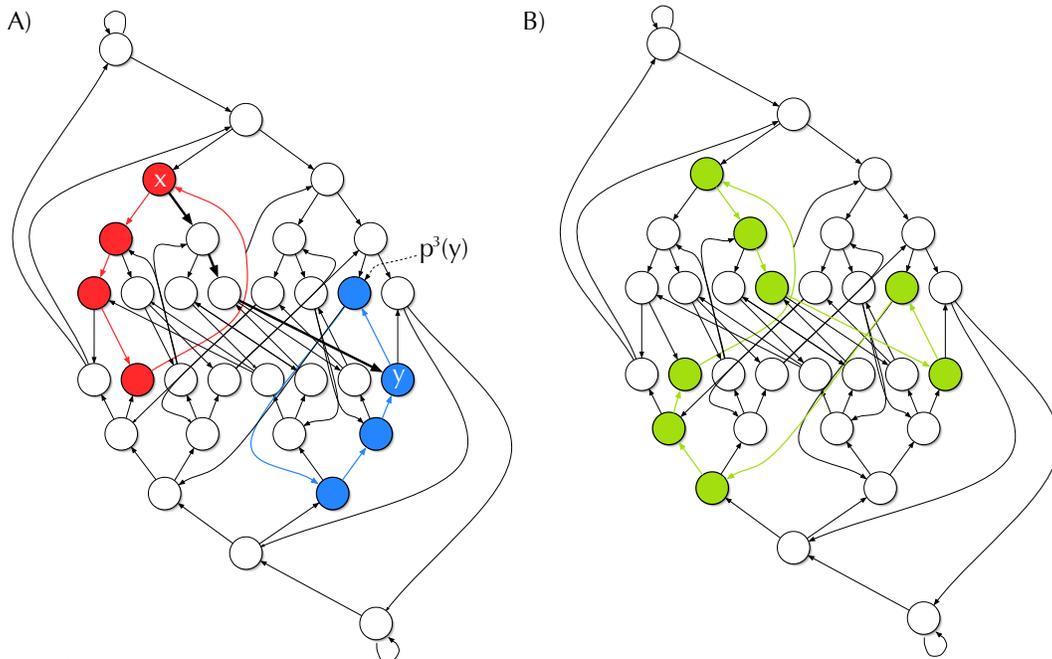


Figure 7.9: A) Two 3-flat paths of length 4, at distance 3 from each other. (x, y) points of closest approach. B) Their sum.

Proof: Let C_{i_1} and C_{i_2} be two distinct cycles such that $dist(C_{i_1}, C_{i_2}) \leq dist(C_i, C_j)$ for all $i \neq j$. But then apply the sum operation, i.e. form $C^* = C_{i_1} \otimes_{y'}^x C_{i_2}$ where (x, y) is a point of closest approach of C_{i_1} and C_{i_2} . Since $N_1(C_{i_1} \otimes_{y'}^x C_{i_2}) \cap (C_{i_1} \cup C_{i_2}) \subset \{x, y\}$, and $N_1(C_{i_1} \otimes_{y'}^x C_{i_2}) \cap C_j = \emptyset$ for $j \neq i_1, i_2$, we can replace C_{i_1}, C_{i_2} with $C_{i_1} \otimes_{y'}^x C_{i_2}$, obtaining a new set of k -flat cycles whose mutual distances are at least $k + 1$ – but now with one fewer components one of which is of size $|C_{i_1}| + |C_{i_2}|$. Applying this procedure repeatedly until one component is left finishes the argument. ■

7.6 Subgraph Geometry

The structure described in *DB* by combining the flatness-increasing embedding γ , and the flatness-preserving \otimes operation, leads to a partial characterization of the subgraph geometry of the DeBruijn graphs. The main result is that:

Theorem 4 • A graph G can only be embedded k -flatly in $DB(n, m)$ for $k \leq n$.

- The maximal size of a k -flat subgraph in $DB(n, m)$ is m^{n-k} .
- There are precisely

$$\frac{(m!)^{m^{n-k-1}}}{m^{n-k}}$$

maximal k -flat subgraphs in $DB(n, m)$.

- There are k -flat subgraphs of all sizes between 1 and m^{n-k} .

To prove this, I prove a number of subsidiary results.

Proposition 41 Suppose that $G \subset DB(n, m)$ is a union of k -flat cycles whose mutual distances are at least $k + 1$. Then $|G| \leq m^{n-k}$.

Proof: I will give two proofs:

#1: Suppose otherwise, and let G be an example. For each $x \in G$, consider the k -neighborhood $N_k(x, DB)$ of x in $DB(n, m)$. Since $|G| > m^{n-k}$, by the pigeonhole principle there will be an x' such that $N_k(x', DB) \cap G$ contains at least $k + 1$ distinct elements. Hence, for some $j \leq k$, $C^j(x', DB)$ contains at least two elements. Now, let E be the k -flat cycle in G that x' is in. One of the things in $C^j(x', DB) \cap G$ must be $c^j(x, E)$, the j -child of x in the cycle it lies on. Let's consider what's left, i.e. $y \in C^j(x, DB) \cap G - c^j(x, E)$. If on the one hand $y \in E$, then E is not k -flat; if on the other hand $y \notin E$, but is instead in some other cycle E' in G , then the mutual distance between E and E' is at most $j < k + 1$, also a contradiction.

#2: If $G \subset DB(n, m)$ is a union of k -flat cycles whose mutual distances are at least $k + 1$, then $\gamma^{-k}(G)$ is a set of disjoint cycles in $DB(n - k, m)$. But $DB(n - k, m)$ contains only m^{n-k} elements. ■

Proposition 42 For each $k < n$, $DB(n, m)$ contains a k -flat cycle of size m^{n-k} , i.e. proposition 41 is sharp.

Proof: I will give two proofs again.

#1: Suppose $n > k$. Proposition 37 guarantees the existence of a hamiltonian cycle H in $DB(n - k, m)$. It has size m^{n-k} . Then $\gamma^k(H)$ is, by proposition 39, a k -flat cycle of size m^{n-k} in $DB(n, m)$.

#2: Let H_0 be any subset of $DB(n, m)$ with the following property: for all $x \in DB$, and $i \leq k$, $|P^i(x) \cap H_0| = |C^i(x) \cap H_0| = 1$. That is, for $i \leq k$, H contains exactly one i -child and one i -parent of each node in $DB(n, m)$. To see that such a set exists, simply take H_0 to include all nodes whose numbers in the realization described in §7.3 have residue 1 modulo m^k , that is, $m^k j + 1$ for $0 \leq j < m^{n-k}$. Note that any H with this property has $|H_0| = (1/m^k)|DB| = m^{n-k}$, and is the union of k -flat cycles that are distance $k + 1$ apart cycles. Now apply proposition 8. ■

The above is actually a special case of a more general result:

Proposition 43 $DB(n, m)$ contains k -flat cycles of all sizes between 1 and m^{n-k} .²

The proof of this result is given in appendix §G.3.

A *disjoint cover* of $DB(n, m)$ is a set of cycles D such that i) for all $c_i, c_j \in D$, $c_i \cap c_j = \emptyset$ whenever $i \neq j$ and ii) $\bigcup D = \bigcup_{c \in D} c = DB(n, m)$. A *k -dispersed cover* is a set D of k -flat cycles with distance $k + 1$ between neighboring cycles such that $\bigcup_{c \in D} B_r(c) = DB(n, m)$, so that $|\bigcup_{c \in D} c| = m^{n-k}$.

Proposition 44 There are precisely $(m!)^{m^{n-k-1}}$ k -dispersed covers of $DB(n, m)$. Of these, a fraction $(1 - 1/m)^m$ contain no self-loops.

Taking $k = 0$ gives $(m!)^{m^{n-1}}$ disjoint cycle covers of $DB(n, m)$.³

Proof: In light of the second part of proposition 39, we need only prove the result for $k = 0$ and then apply γ .

To make a disjoint cover of $DB(n, m)$ we simply need to assign each node in $DB(n, m)$ a unique child such that no two nodes are assigned the same child. We saw in §7.3 that each tree G_i is divided into m parts – the main body of the tree (non-leaf nodes), T_i – of which there are m^{n-2} in each tree, and whose children are in the tree; and the leaves L_i , which come in $m - 1$ consecutive groups of leaf nodes L_i^k , each containing m^{n-2} nodes, in which the k -th group of leaves in G_i distributes its edges equally among the m^{n-2} nodes of G_{i+k} (where $i + k$ is taken modulo m).

To make a covering D let's start first with the main tree bodies T_i . For each $x \in T_i$ choose a child c_x arbitrarily among the m elements of $C(X)$, and put the edge (x, c_x) in D . There are m^{n-2} elements in each T_i and m such trees, so there are a total of m^{n-1} choices made. Since each choice has m possibilities, the total number of ensemble choices is $m^{m^{n-1}}$.

Now let's move on to the first consecutive groups of leaves, L_i^1 , for each i . Again, we'll choose children for L_i^1 (which are nodes in G_{i+1}) arbitrarily, except now avoiding the elements chosen in the first round, i.e. for each $x \in L_i^1$, we choose arbitrarily amount $c(x) - D$. 1 in each group of m siblings is the target of a link from the first round, so that there are $m - 1$ possible choices for each node x . Since there are m^{n-2} nodes in each L_i^1 , and m sets L_i^1 , there are again m^{n-1} choice-points, for a total of $(m - 1)^{m^{n-1}}$ ensemble choices. Moving on

²There is a partial proof of a less general result in [11]

³After proving this result, I found that it had already been. See, for example, the Wikipedia site on "DeBruijn sequences. No prove is provided there.

to the second groups of leaves, L_i^2 , now avoiding in the arbitrary choices the things chosen in the first two rounds. Again, there will be m^{n-1} elements in $\cup L_i^2$ and $m - 3$ choices for each, for a total of $(m - 3)^{m^{n-1} - m^{n-2}}$ ensemble choices. Add these to D .

Continue to repeat this process, for each consecutive groups of leaves, choosing thereby children for all nodes. At each tree, the number of choices goes down by one, so that in the end, there are:

$$\prod_{i=0}^{m-1} (m - i)^{m^{n-1}} = (m!)^{m^{n-1}}$$

total choices, as claimed.

To ensure no self-loops in D , we simply need to make sure that the m self-loop nodes, whose children are all chosen in the first round of the process above, do not choose themselves. This means that in the first round, the m self-loop nodes have $m - 1$ possible choices, while the remaining $m^{n-1} - m$ have m choices; all the other choices proceed as above. This leads to

$$(m - 1)^m m^{m^{n-1} - m} \prod_{i=1}^{m-1} (m - i)^{m^{n-1}} = \left(\frac{m - 1}{m}\right)^m (m!)^{m^{n-1}}$$

total choices, which is $(1 - 1/m)^m$ fraction of the first number, as claimed. ■

Hamiltonian cycles of $DB(n, m)$ are equivalent to disjoint covers D of $DB(n, m)$ that are singletons, i.e. $|D| = 1$, and maximal k -flat cycles to singleton k -dispersed covers. A disjoint cover D is *irreducible* if every element of D is an irreducible cycle in $DB(n, m)$.

Proposition 45 *A fraction $1/m^{n-k}$ of all k -dispersed covers are singleton, for a total of*

$$\frac{(m!)^{m^{n-k-1}}}{m^{n-k}}$$

maximal-length k -flat cycles; and the number of irreducible disjoint covers is equal to the number of hamiltonian cycles.

Taking $k = 0$ yields $(m!)^{m^{n-1}}/m^n$ hamiltonian cycles, while taking $k = 1$ gives $(m!)^{m^{n-2}}/m^{n-2}$ irreducible cycles. The proof of prop. 45 is given in §G.4.

7.7 The Interpretation of $\mathbb{D}(r, m)$ Structure

Any subgraph $G \subset \mathbb{D}(r, m)$ corresponds to the pattern T_G consisting of the set of configurations corresponding to maximal paths in G . The concept of flatness in definition 34 can be viewed as establishing a hierarchy of increasingly strong structural equivalences between such patterns:

Definition 35 [Hierarchy Structural Equivalence] *Two patterns T_G and T_H , where $G \subset \mathbb{D}(r, m)$ and $H \subset \mathbb{D}(r', m')$, are k -equivalent if*

$$N_k(G) \cap G$$

is isomorphic with

$$N_k(H) \cap H$$

where $N_k(\cdot)$ denotes the k -neighborhoods of G and H in $\mathbb{D}(r, m)$ and $\mathbb{D}(r', m')$ respectively.

As k increases, k -structural equivalence becomes an increasingly strong criterion.

We will see in the few sections that three broad ranges of k -equivalence have useful concrete interpretation for understanding locally checkable patterns and the local rules that solve them. The $k = 0$ level corresponds to purely topological properties of patterns, and informs the issue of static encoding and radius-state tradeoff. The $k = 1$ level corresponds to the induced subgraph topologies from $\mathbb{D}(r, m)$, and informs the issue of local checkability. $k > 1$ corresponds to more detailed geometric equivalence, and informs the issue of a patterns' robustness under perturbation.

7.7.1 $k = 0$: Topological Structure and Static Encodings

Taking $k = 0$ in def. 35 above is equivalent to:

Definition 36 [Topological Equivalence] Two patterns T_G and T_H are topologically equivalent, denoted $T_G \sim T_H$, if G and H are isomorphic graphs.

Many relevant features of a pattern are “topological invariants”. If $G \sim H$ as graphs then T_G and T_H can be treated as the “same pattern” in many regards:

- T_G and T_H contain the same admissible sizes, i.e. $Sizes(T_G) = Sizes(T_H)$.
- T_G and T_H have the same combinatorics, i.e.

$$\mathbf{A}_{T_G} = \mathbf{A}_{T_H},$$

where \mathbf{A}_T is the ordinary generating function associated with T , defined by

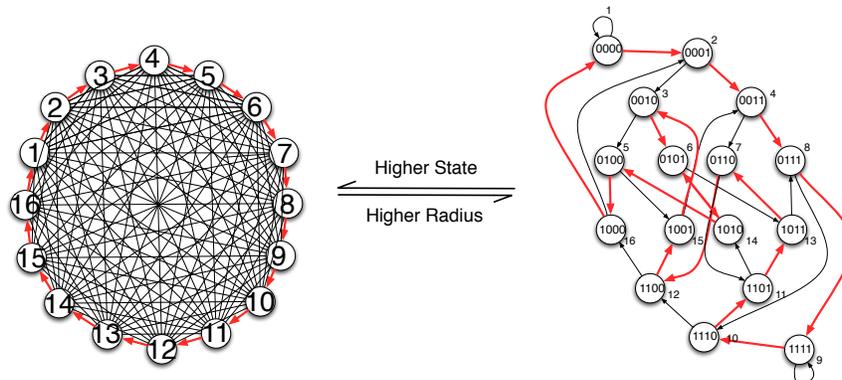
$$\mathbf{A}_T(z) = \sum_{n=1}^{\infty} |T \cap \mathbf{C}_n| z^n$$

for complex numbers z .

- T_G and T_H have the same fundamental frequencies, i.e. $\omega(T_G) = \omega(T_H)$, where $\omega(T)$ is as defined in §5.5.⁴
- T_G and T_H are mutual encodings, i.e. there are decoding functions ϕ and ψ such that (T_G, ϕ) is an encoding of T_H and (T_H, ψ) is an encoding of T_G .
- The left- and right- choice functions O, P of G are in 1-1 correspondence with left- and right-choice functions O', P' of H . Hence the Naive and Less Naive backtracking algorithms for T_G for each choice function are strict dynamical encodings of their corresponding versions for T_H . Similarly, T_G is locally patchable if and only if T_H is locally patchable, and if so then the locally patching algorithm for T_G is a strict dynamical encoding of that for T_H .

Of these equivalences, the most “practically useful” is the mutual encoding property, necessary for efficient implementation of the static radius-state tradeoff.

Example 46 For each n , the graph $DB(1, 2^n)$ contains a cycle C_1 of length 2^n corresponding to the path $(1, 2, \dots, 2^n)$. On the other hand, $DB(n, 2)$ contains a (hamiltonian) cycle C_2 , also of length 2^n . C_1 corresponds to the discrete gradient of length 2^n , whereas C_2 is the DeBruijn encoding of that gradient using 2 states. In the $n = 4$ case, for instance, this looks like:



The fact that C_1 are topologically equivalent corresponds to the radius-state tradeoff between high-state/low radius encodings in $DB(1, 2^n)$ and low-state/high-radius encodings in $DB(n, 2)$.

⁴Notice however that two patterns that have the same generating function \mathbf{A} may not have the same fundamental frequencies. In this sense, many of the properties in this list are “independent” combinatorially.

We thus have the outlines of a general procedure for static radius-state encoding:

To find a strict encoding in m states of a given pattern T_G , find a subgraph of $DB(n, m)$ for some n that is a covering of the graph G . The minimum n for which this is possible determines the smallest radius r for which T can be encoded in m states.

The static radius-state tradeoff is thus an “existence problem”: which graph topologies arise as subgraphs of $DB(n, m)$? On the one hand, the fact that the high-radius \rightarrow high-state tradeoff is easy follows from the fact that this existence problem is trivialized by increasing m , since $DB(1, m)$ is the complete graph on m elements and every graph can be easily be embedded in a sufficiently large complete graph. Theorem 4, part 4, answers the reverse question for the simplest sorts of graphs, those which consist of disjoint unions of cycles. In effect, the result says that every such graph G that arises in $DB(n, m)$, and has no more than 1 cycle for each size $\leq n$, has a covering that arises as a subgraph of $DB(2n\lceil\log_2(m)\rceil, 2)$, and the proof of the result gives a recipe for finding the covering. One direction in which I would like to generalize theorem 4 would be to show that there is some constant C such that any degree- k subgraph G arising in $DB(n, m)$ with sufficiently few small cycles, can be covered by a graph arising in $DB(Cn \log_k(m), k)$; a constructive proof of this result would be the substance of a generalized most-efficient radius-state tradeoff algorithm.

Given a positive answer to the existence problem of whether a pattern with a given topology can be built with radius r and m states in the first place, the next engineering question is to ask how *hard* it is to find such a pattern. This entails measuring how common such patterns are compared to a larger (and presumably easier to construct) set of patterns that share some features with the original pattern. The exact form of such a measure depends on what is considered “easier to construct” and which features are “shared”. Given a graph G , it will typically be easy to construct a graph with the same out-degree distribution. If we denote by N_G the number of graphs isomorphic to G and by N the number of graphs with the same out-degree distribution as G , then the log-likelihood ratio $H = -\log(N_G/N)$ is a reasonable notion of “hardness”. Theorem 4 allows us to compute this measure for the case of G being a hamiltonian cycle in $DB(n, m)$; since $N_G = (m!)^{m-1}/m^n$ and $N = m^{m^n}$, we obtain (after applying Stirling’s approximation) that $H \sim n + m \log_m(e)$. If it is considered “easy” to construct graphs with the same in- and out-degree distribution, then the measure $H' = -\log_m(N_G/N')$ is also reasonable, where N' is the number of graphs with the same degree distribution as G . In this case, combining propositions 44 and 45 says that hamiltonian cycles in $DB(n, m)$ have $H' = n$. In the future, generalizing Theorem 4 to compute the complete cycle distribution of $DB(n, m)$ will allow a much more thorough analysis of this problem.

7.7.2 $k = 1$: Induced Topology and Local Checkability

Just because a graph G arises as a subgraph of $ID(r, m)$ does not mean that there is a radius- r local check scheme Θ for T_G . This is because to be locally checkable with radius r , a pattern has to correspond to an *induced* subgraph of $ID(r, m)$. Since not all subgraphs of $ID(r, m)$ are induced subgraphs, there will be patterns associated with subgraphs of $ID(r, m)$ which cannot be locally checked at radius r . A very simple example of this is that the hamiltonian cycles in $DB(n, m)$, corresponding to DeBruijn encodings, are NOT induced subgraphs of $DB(n, m)$ and are not locally checkable with radius $n/2$.

The obvious question, therefore, is: when are patterns associated with non-induced subgraphs of $ID(r, m)$ locally checkable? And when they are locally checkable, what is their local check radius, in relation to r ? This is a stricter sort of existence question than purely topological, asking: when does $ID(r, m)$ contain a subgraph G of a given topology such that G is a retract of its own radius-1 neighborhood inside $ID(r, m)$? Proposition 39 answers this question when G is a union of disjoint cycles. For all such G , $\gamma(G)$ is an induced subgraph of $ID(r + 1/2, m)$, consisting of a union of irreducible cycles whose mutual distances within $ID(r + 1/2, m)$ are at least 2. This has the immediate practical application of allowing us to find a local check scheme (and thus local rule solution) to the DeBruijn encoding:

Corollary 9 *If q is a DeBruijn sequence of length m^n , then graph $\gamma(q)$ corresponding to the $n + 1$ -windows in q constitute a radius- $(n + 1)/2$ local check scheme for the repeat pattern T_q . Hence, the local rule $F_{\gamma(q)}$ will generate a DeBruijn encoding from any initial condition.*

Pictorially, this is the situation indicated in figure 7.8.

Theorem 4 implies that the maximum possible length of an irreducible cycle in $DB(n, m)$ is at most m^{n-1} , and that this bound is tight. Imposing condition of “irreducibility” on a cycle, which is determined by its 1-neighborhood, constrains its size by factor of $1/m$, compared with the largest non-irreducible cycles in $DB(n, m)$. As a result:

Corollary 10 *For any infinite locally checkable patterns T , the quantity*

$$1/m^{2 \cdot \text{LCR}(T)}$$

is a tight lower bound on:

- *the density of $\text{Sizes}(T)$ in the natural numbers.*
- *the coefficients in $\text{Growth}_T(n)$.*
- *the elements in the set of fundamental frequencies, $\omega(T)$.*

Analogous numerical bounds derived without considering the induced topological structure are too small by a factor of $1/m$.

Given a check scheme Θ in $\mathbb{D}(r, m)$, it might be the case that a radius smaller than r is sufficient to generate the same pattern as that generated by Θ . The discussion above about induced cycles implies a very simple criterion for determining this:

Proposition 46 *Suppose Θ is a local check scheme with $G(\Theta) \subset \mathbb{D}(r, m)$. Then $r(\Theta)$ is the smallest radius r for which there is a check scheme Θ' of radius r with $\Theta(\mathbb{C}) = \Theta'(\mathbb{C})$ if and only if there is an irreducible cycles $C \subset G(\Theta)$ such that $\gamma^{-1}(C)$ is not irreducible.*

This result generates a simple algorithm for minimizing a check scheme’s radius. Specifically, for any directed graph G let $\text{IrrCyc}(G)$ denote a list of the irreducible cycles in G .

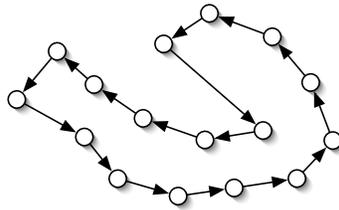
Algorithm 3: Radius Minimization Algorithm

Data: Local Check Scheme Θ
Result: Minimal Check Scheme $\tilde{\Theta}$
 $\text{minimal} \leftarrow \text{FALSE};$
 $\tilde{\Theta} \leftarrow \Theta;$
while $\neg \text{minimal}$ **do**
 if $\exists C \in \text{IrrCyc}(G(\tilde{\Theta}))$ *such that $\gamma^{-1}(C)$ is reducible* **then**
 $\text{minimal} \leftarrow \text{TRUE};$
 else
 $\tilde{\Theta} \leftarrow \gamma^{-1}(\tilde{\Theta});$
 end
end

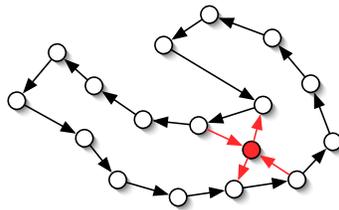
7.7.3 $k > 1$: Local Geometry and Robustness

Throughout this thesis we’ve focused on robust *rules* – dynamical rules which produce some desired pattern from many initial conditions. The notion of robustness for the pattern itself is somewhat different: intuitively, a locally checkable pattern is robust if small changes to the set of admissible configurations do not radically change the patterns overall properties. In this section, we briefly discuss this idea of robust patterns, and show how the notion can be understood in terms of the patterns’ associated DeBruijn graph.

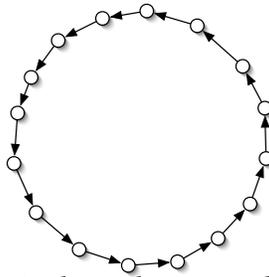
It turns out that a pattern’s robustness under perturbation is neither a topological nor induced-topological invariant. Instead, it is a function of the patterns more detailed “local geometry.” To see why, let’s take for example a repeat pattern T_q . The graph $G(T_q)$ contains a single cycle C corresponding to the single repeating unit q . The cycle C is embedded in $DB(|q|, m)$ in some way, looking something like this:



Suppose that distances in the drawing above are meant to be literal, that the cycle curved and twisted so that in some points nodes that are far away in the cycle come close together in the ambient space. Near these points, a small number of “point mutations” in the structure of the pattern could dramatically change the topology of the graph, and thus the properties of the resulting pattern:



The mutation caused the segment to break into two subsegments, both with significantly shorter lengths. On the other hand, suppose the cycle were instead been embedded in DB like this:



in which no point comes close to another in the cycle, except through the cycle itself. In this case, more mutations will be necessary to drastically change shape of $G(\Theta)$, and thus the resulting pattern scaling properties.

This example exposes that:

- The classes of topologies of patterns that arise from single-point mutations of a pattern can depend on features of the pattern other than the pattern’s topology. Two LCSs with the same topology – e.g. two repeat patterns consisting of a single cycle – can mutate to different things, somehow depending on the shape of the pattern’s graph in the ambient space. Said another way, the evolution behavior of patterns under mutation is NOT a topological invariant.
- Some points in a pattern are more vulnerable than others. These places, where small numbers of mutations can have large effects, are the places where the pattern’s local “curvature” is high.

These observations suggest that we should think about another level of detail in pattern space beyond topology, considering geometric notions of curvature. It is just this sort of curvature information that is captured by taking $k > 1$ in def. 35. A pattern Θ whose graph $G(\Theta)$ is k -flat at every point will require at least k mutations to significantly change topology. A pattern that has constant curvature is one with no points being especially vulnerable to change.

From the viewpoint of “local-to-global engineering”, we would like to know how adding a given level of robustness effects the ability design patterns with given topological structures (and therefore, a given set of qualitative features). For the simplest form of patterns (again, those that are unions of disjoint cycles), proposition 39 gives a recipe for increasing robustness for a fixed topology:

Simply apply the γ operator k times to obtain robustness protection against k additional mutations.

Moreover, Theorem 4 shows that

For each layer of robustness against one additional mutation, (a) the maximal obtainable pattern decreases by a factor of $1/m$, and (b) the “hardness” H_1 increases by 1 unit.

In the future, it would be desirable to expand theorem 4 to apply to more general geometries. Another direction of generalization would be to compute *average* robustness, not just maximum robustness, by “integrating” curvature along paths in the graphs. In the long run, it may be possible to prove an analog of a “Gauss-Bonnet” theorem for the DeBruijn graph, connected the topology of a subgraph with (bounds on) the average curvature. This would be practically useful for understanding the limits on pattern robustness.

Related Work

The DeBruijn graph, especially in its connections to coding theory, has been a subject of fairly extensive study in computer science [10, 11, 30]. Its connection to cellular automata – through the route of regular languages – has also been noticed in [37] (though little of its structure is recognized or used there). For some reason, however, the study of its structure as the key classifier of both regular languages and stable fixed points of robust local rules seems to have been largely neglected.

The DeBruijn theory as developed in this chapter seems (as far as I can tell) to be essentially orthogonal to the standard theory of regular languages. On the one hand, the Myhill-Nerode monoid decomposition theory (see e.g. [8]) is essentially blind to the “internal structure” of the labels on the NDFA graph corresponding to a regular language. But these labels – i.e. the symbol strings sitting on top of the nodes in the DeBruijn graphs – are precisely what make the distinctions between the levels in the DeBruijn hierarchy make sense. By abstracting away from the physical representation of agents and their local fixed-radius state configurations, to a level where many different labeling schemes could have produced equivalent graph structures, the monoid theory loses much of the structure that matters for our purposes. Practical results of this are, for example, that the radius-minimization algorithm I describe in §7.7.2 is quite different from the *minimal automata* algorithm described by Eilenberg, and that the counting and characterization questions throughout the chapter simply do not arise in the monoid theory. The algebraic generating function representation of languages loses much of the same structure (and more).

The fact that regular languages generated by a specific “physical instantiation” with fixed-radius agents have very nontrivial “geometry” not captured by the standard theories seems to me to be an area where much structure goes essentially unrecognized. Perhaps the main reason this has happened is that the questions of information requirements (the $k = 0, 1$ cases) and robustness (the $k > 1$ case) are motivated by the construction problem with spatially embedded computing agents but not so clearly by the recognition or decidability problems that are normally studied regarding languages. The rich detailed structure hinted at in this chapter, which matters integrally for local-rule design and analysis problems, may point to a new direction of research in language theory.

Chapter 8

Two Applications

In this chapter I sketch two applications of the techniques developed in the previous chapters: building a prototype Global-to-Local Compiler, and analyzing a developmental gene regulatory network from the fruit fly *Drosophila melanogaster*.

8.1 A Prototype Global-to-Local Compiler

A Global-to-Local Compiler is a procedure which takes as input a target pattern and set of resource limits, and whose output is a local rule generating the pattern using no more than the specified resources. The utility of such a compiler is that it “takes out the work” of having to figure out exactly how to program local rules, and allows a user’s global design intent to be directly transduced into local agent programs. In this section I show how to assemble the results from previous chapters to build a prototypical Global-to-Local compiler for our 1-D model.

8.1.1 What Kind of a Thing is a Global-to-Local Compiler?

Loosely speaking, the general mathematical format of a global-to-local compiler is simply an *algorithmically computable function* GC :

$$GC : \text{Global Task Set} \longrightarrow \text{Local Solution Set}$$

such that for any task T in the global task set, $GC(T)$ is a local solution to T . In the present case, the Global Tasks are spatial pattern formation problems, so the domain of our compiler will be \mathcal{T} , the set of all 1-D patterns T over finitely many states. Our Local Solutions are local fixed-radius rules, so the range will be \mathbb{D} , the set of all such rules (as defined in §1.2).

Of course, in many situations, agents have limitations on their radius and memory. A Global-to-Local compiler should also therefore take as input a description of resource limitations, in the form of a maximum usable radius R and amount of state M . Ideally, the compiler would then output a local rule that is a solution to the given global task, and which uses no more than the indicated resources. However, there is a tradeoff between R and M , so it may be impossible to meet specified limits on R and M simultaneously, at least for the given pattern. Hence, there are three forms of Global-to-Local Compiler:

- (Type 1) A function GC_1 whose inputs are a pattern T and a radius limit R , and whose output is a local rule solution F to T such that $r(F) \leq R$, using whatever amount of state M might be necessary.
- (Type 2) A function GC_2 whose inputs are a pattern T and a state limit M , and whose output is a local rule solution F with no more than M states, using whatever radius might be necessary.
- (Type 3) A function GC_3 whose inputs are pattern T , the radius limit R , and the state limit M , and whose outputs are:

- either a solution F to pattern T , such that $r(F) \leq R$ and which uses no more than M states,
- or, an error message “ \emptyset ” when the resource limits conflict and no solution exists meeting both.

We could implement any of these three types by deploying the radius/state tradeoff algorithms from chapter 6 in different ways.

Finally, the compiler should in general be *performance optimal*, meaning that its output should be a rule that solves each global task as fast and as with as little state/radius overall as possible. Hence, if there are different classes of global tasks whose optimal solutions have different architectures, the compiler should treat these tasks separately. In the present case, this means that the basic pattern classes, as discussed in §4.3, should each be solved by their respective optimal local rules.

8.1.2 Pattern Description for Input Specification

The results of the previous chapters get us most of the way to the compiler. One missing piece, however, is that to make the procedure really automatic and computable, the input pattern T must be *described finitely*. Just thinking of T as an unstructured set of final configurations, as we have been doing since making Definition 8 in §1.3, is not enough. If we want to work with big or infinite pattern sets (which we have been), we need to do something more sophisticated.

We thus are bumping up against a more general Problem about spatial multi-agent systems that could have been listed in the introduction to the thesis, along with the existence, construction, optimization, &c problems:

A Description Problem: Find a description language that efficiently and intuitively captures the inherent structure of global tasks.

An extremely simple approach is an idea I call *local feature invariance*: the user specifies a set of *sample configurations* that are instances of the pattern, together with a *feature radius* at which she’d like the features of the samples to be preserved. We then compute the simplest local check scheme consistent with those features.

Formally, let $\mathcal{X} = \{X_1, X_2, \dots, X_K\}$ be a finite set of configurations (the samples) and fix an $r > 0$ (the feature radius). Let

$$\mathcal{B}(\mathcal{X}, r) = \{B_r(i, X) \mid X \in \mathcal{X}, i \in \{1, \dots, |X|\}\},$$

that is, the set of all distinct r -neighborhoods present in the samples. Define a local check scheme $\Theta(\mathcal{X}, r)$ to accept only r -neighborhoods consistent with the samples, i.e.

$$\Theta(\mathcal{X}, r)(B) = \begin{cases} 1, & \text{If } B \in \mathcal{B}(\mathcal{X}, r) \\ 0, & \text{otherwise} \end{cases}.$$

Denote the pattern generated by $\Theta(\mathcal{X}, r)$ as $T(\mathcal{X}, r)$. By definition, $T(\mathcal{X}, r)$ is the set of all configurations Y such that $\Theta(\mathcal{X}, r)$ accepts every r -neighborhood in Y . Since all the r -neighborhoods accepted by $\Theta(\mathcal{X}, r)$ are exactly those appearing as r -neighborhoods in the sample set \mathcal{X} , $T(\mathcal{X}, r)$ can simply be thought of as the largest pattern consistent with the features of the original samples, at scale determined by r . In words, the samples are a set of “pictures” that indicate features to be generalized into the complete (and often infinite) pattern.

For example, let

$$\mathcal{X}_1 = \{(100000010000001000000)\}.$$

This sample is evidently “trying” to capture the 1000000 repeat pattern. Taking large enough radius generates the “right” answer, i.e.

$$T(\mathcal{X}_1, 4) = T_{1000000}.$$

By choosing a smaller radius, other structures emerge. For instance,

$$T(\mathcal{X}_1, 2) = 10000 \cdot (T_{10000} \times T_{00000}) \cdot 00000.$$

Only a few samples are required to generate complex patterns. For the three-sample set

$$\mathcal{X}_2 = \{100100, 10001000, 1001000\},$$

we have

$$T(\mathcal{X}_2, 3) = T_{100} \cdot T_{1000}.$$

Thus the logic and concatenation operators $\{\wedge, \vee, \circ, \times\}$ described in at the end of chapter 2 are easily expressed by the invariant features of their samples.

There are many other approaches to addressing the description problem besides local feature invariance, and the question of description has interest beyond that of building a compiler. See appendix §F for a discussion of several, include the use of formal logic.

8.1.3 The Compiler

We're now ready describe a global-to-local compiler. Here, I will first build a compiler of Type 1; the other types can be patterned on this construction.

Step 0: Input. Obtain input in the form of a sample set \mathcal{X} , a feature radius r , and a maximal allowable system radius R .

Step 1: Generate Pattern. Compute the check scheme $\Theta(\mathcal{X}, r)$ defined local feature invariance technique defined in §8.1.2.

Step 2: Minimize Radius. Use algorithm 3 in §7.7.2 to compute $\widetilde{\Theta}(\mathcal{X}, r)$, the minimum-radius check scheme equivalent to $\Theta(\mathcal{X}, r)$.

Step 3: Resource Tradeoff. Use the radius \rightarrow state tradeoff algorithm defined in §6.1.2 to compute:

$$\Theta(\mathcal{X}, r, R) = \begin{cases} (\widetilde{\Theta}(\mathcal{X}, r))^{R/2}, & \text{if } \Theta(\mathcal{X}, r) \text{ is single-choice} \\ (\widetilde{\Theta}(\mathcal{X}, r))^{(R-2)/2}, & \text{otherwise} \end{cases}.$$

Step 4: Construct Optimal Rule. Using the pattern classification described in §4.3, the single-choice algorithm f_{Θ} construction from Chapter 3, and the algorithms \widetilde{F}_{Θ} and \widehat{F}_{Θ} from Chapter 4, define:

$$F(\mathcal{X}, r, R) = \begin{cases} f_{\Theta(\mathcal{X}, r, R)}, & \text{if } \Theta(\mathcal{X}, r) \text{ is single-choice} \\ \widetilde{F}_{\Theta(\mathcal{X}, r, R)}, & \text{if } \Theta(\mathcal{X}, r) \text{ is multi-choice and not locally patchable} \\ \widehat{F}_{\Theta(\mathcal{X}, r, R)}, & \text{if } \Theta(\mathcal{X}, r) \text{ is locally patchable and nontrivial} \end{cases}.$$

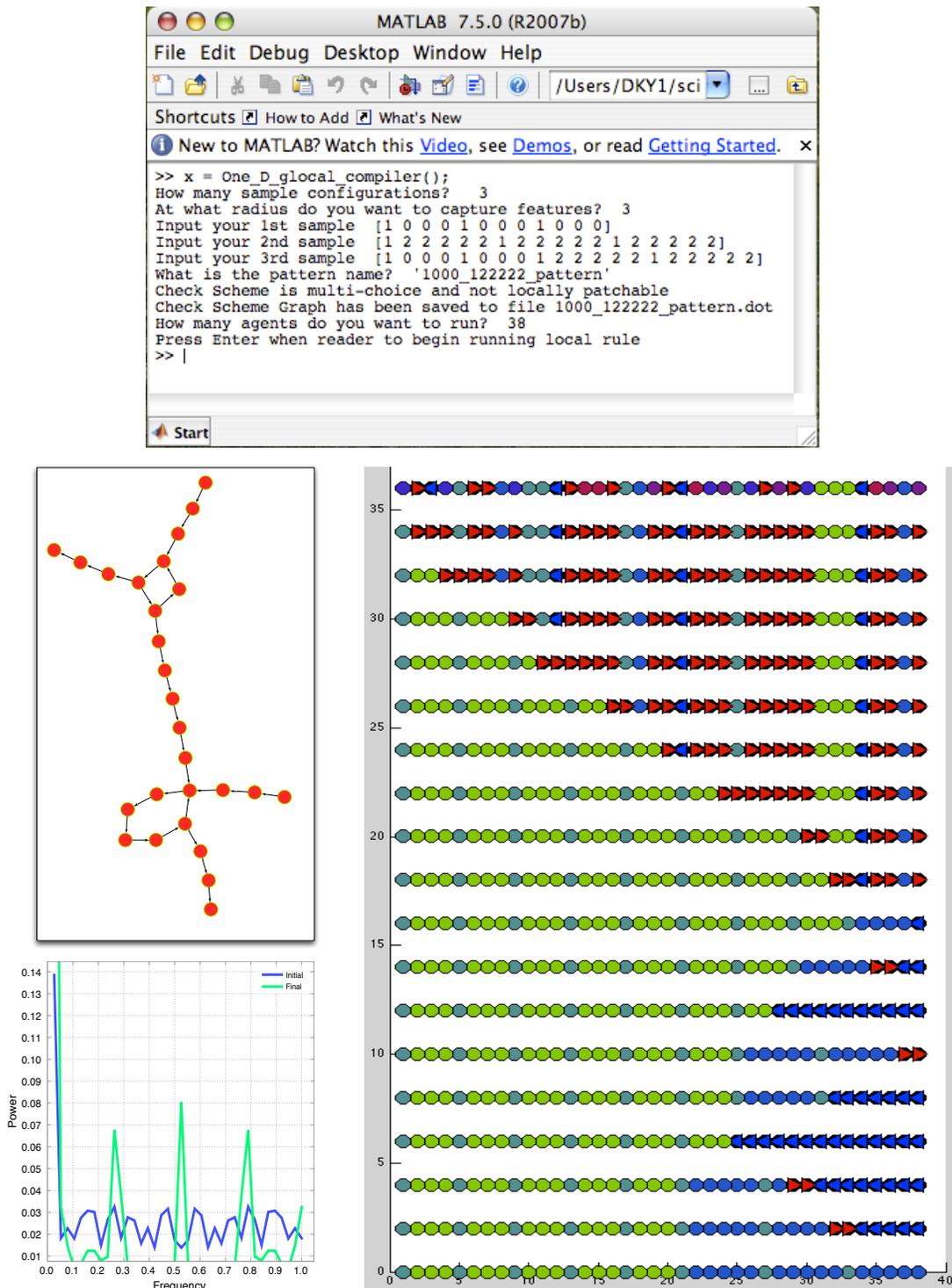
The compiler is given by the function $GC : (\mathcal{X}, r, R) \mapsto F(\mathcal{X}, r, R)$. $GC(\mathcal{X}, r, R)$ is, by construction a local rule that solves the pattern $T(\mathcal{X}, r)$, with radius at most R .

I have implemented a version of this compiler in Matlab. Fig. 8.1 shows a screen shot of the Matlab code in action. The input to this example were three samples

$$\mathcal{X} = \{100010001000, 122222122222122222, 10001000122222122222\}.$$

The chosen feature radius is 3, and the pattern generated is $T_{1000} \cdot T_{122222}$. The output includes an *GraphViz* .dot file containing the minimal local check scheme graph, and a tracing of the power spectrum order measure as the system self-organizes, as well as the local rule itself. It is important to note that this compiler is *prototypical*: a real global-to-local compiler, to be used by a real spatial computer programmer, would have to handle larger classes of problems and work on a large class of underlying spaces. In chapter 9 I indicate how in future work this might be accomplished.

Figure 8.1: **Top:** Compiler console with commands being run. Three samples input are: 100010001000, 122222122222122222, 10001000122222122222. Feature radius is 3. The pattern generated is $T_{1000} \cdot T_{122222}$. **Middle Left:** Minimal-radius local check scheme graph as generated by compiler. **Middle Right:** Compiler selects random initial and runs forward generated rule until final configuration is reached. Selected timesteps from trajectory produced on random initial condition are shown. Timesteps proceed from top to bottom. **Bottom Left:** Power spectrum of initial condition (blue line) and final condition (green line), the showing the emergence of global order in the latter in comparison to the former.



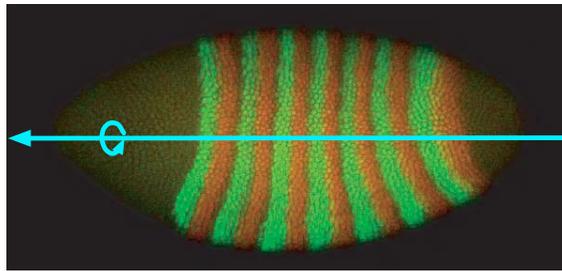
8.2 Analyzing a *Drosophila* Gene Network

In this section, I will show how the techniques of spatial multi-agent systems may be useful in analyzing the structure of a genetic regulatory network that figure in the embryonic development of the common fruit fly, *Drosophila melanogaster*. I present the necessary background material from *Drosophila* biology in §8.2.1, describe a simple multi-agent model of the *Drosophila* embryo in 8.2.2, and get to the analysis in §8.2.3.

8.2.1 *Drosophila melanogaster*

Drosophila melanogaster is a two-winged insect that looks like this:  – but about half that size. It is one of the most commonly used model organisms in biology, especially in developmental genetics.

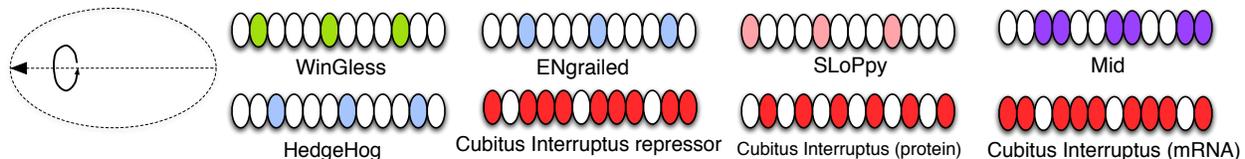
Shortly after a *Drosophila* egg is fertilized, fluorescence imaging techniques reveal the existence of a distinct striped pattern that is roughly radially symmetric around the embryo's major axis:



The stripes are important because they are developmental precursors of major body components in the adult fly.

To see how these stripes are made, let's "zoom in" and consider the embryo at the cellular level. Any individual cell in the embryo is like a partially-mixed fluid compartment containing biomolecules, mostly proteins and mRNAs. Experiments show that the macroscopic stripes correspond to contiguous bands of cells that contain differing concentrations of specific proteins. These proteins are the product of expression of specific genes in the fly's DNA.

Among these genes are the so-called "Segment Polarity" genes, which are known to be crucial in the stripe-formation process. Cells in a band corresponding to one stripe have certain Segment Polarity genes turned "on" (highly expressed) and others turned "off", while in other bands the reverse is true. The spatial pattern of on/off states for each of eight core Segment Polarity gene/proteins has a well-defined repetitive sequence, as shown here:



For instance, the gene called *engrailed* (EN, shown top left), repeats in a 4-cell "Off-Off-On-Off" pattern. The superposition of all these repetitive expression patterns constitute the stripes.

But this begs the question. Each of the cells in the embryo has the same underlying genetic program, and the embryo began life as a roughly homogenous single cell. Nonetheless, cells quickly develop into sharply different gene expression regions. How does *this* happen? The answer lies in the fact that the genes making up the stripes *regulate* each other's expression.

When an increase in the abundance of one gene's protein product sets off biochemical reactions that cause another gene's expression to also increase, the former is said to "up-regulate" (or *induce*) the latter. Conversely, when an increase in one gene's protein product leads to a decrease in another gene's expression,

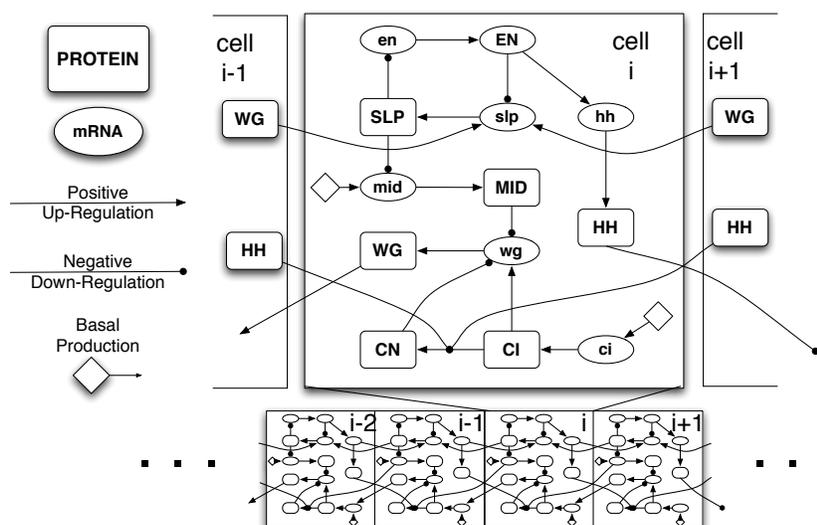


Figure 8.2: Top: *Drosophila* segment regulatory network, as taken from [24]. Genes involved include *engrailed* (*en*), *wingless* (*wg*), *sloppy* (*slp*), *cubitus interruptus* (*ci*), *cubitus interruptus* N-terminal fragment (CN), *hedgheg* (*hh*), and *mid*. Bottom: Inter-cellular spatial coupling of networks, shown along the anterior-posterior axis of the embryo.

the relationship is one of “down-regulation” (or *inhibition*). The genes underlying a given development process are typically linked to each other through sequences of regulatory interactions. The totality of these molecules and their interactions are collectively known as a *gene regulatory network*.

Gene regulatory networks are often represented using graphs. Each node in the network corresponds to a protein or mRNA species that arises from the expression of the genes underlying the network. The edges correspond to regulatory interactions: up-regulation is represented by edges with arrowheads, and down-regulation by edges with circular heads. A series of edges indicates a chain of regulatory dependences, with cycles corresponding to feedback loops. The network structure is an abstraction of the system’s chemical dynamics, describing the manifold ways in which one gene influences all the others to which it is connected. In practice, gene regulatory network diagrams are inferred by distilling up- and down-regulatory influences from the biological literature about a given system.

The graph of the specific Segment Polarity Network (SPN) describing the interactions of the *Drosophila* segment polarity genes is shown in Fig. 8.2. For example, the arrow from the node labeled EN to the node labeled hh indicates that the protein *engrailed* induces the transcription of mRNA of the *hedgheg* gene. (Further details of the specific interactions in the SPN network, corresponding to the various edges in the graph, are described in appendix §H.) In the case of the *Drosophila* stripes, some of the proteins involved in the Segment Polarity regulatory network interact across cell membranes with proteins in neighboring cells. The concentration dynamics of the network in one cell is linked by such intercellular signals to that of its neighbors. The overall structure is thus a *spatially coupled* regulatory network, as indicated at the bottom of fig. 8.2.

Recent work in computational biology [34, 12, 24] have examined models of this coupled network (and related variants), based on differential equations of the form:

$$\frac{dS_j^i}{dt} = f_j(S_1^i, S_2^i, \dots, S_N^i, S_1^{i-1}, S_2^{i-1}, \dots, S_N^{i-1}, S_1^{i+1}, S_2^{i+1}, \dots, S_N^{i+1}), \quad (8.1)$$

in which S_j^i represents the concentration of substance j in cell i . The functions f_j express the regulatory dependences of the system as determined by the network structure. Influences from within a given cell correspond to the S^i variables while interactions with neighbors are captured by the presence of the S^{i-1} and S^{i+1} variables.

The general forms of the functions f_j are based on chemical principles (e.g. the law of mass action, Michaelis-Menten enzyme kinetics), and the specific parameters are chosen to fit the known data. The knowledge that gene product k up-regulates expression of gene j is captured by taking $\partial f_j / \partial S_k > 0$; the sign would be flipped for down-regulation; and no influence represented by $\partial f_j / \partial S_k = 0$. Typically, models in the literature [12, 24] utilize so-called *Hill functions* to represent the induction/inhibition dynamics. For example, the induction of *hedgehog* by *Engrailed* would be represented by the equation:

$$\frac{d hh(t)}{dt} = \frac{1}{\tau_{En}} \left(\frac{En(t)^n}{En(t)^n + k^n} - hh(t) \right), \quad (8.2)$$

where $hh(t)$ and $En(t)$ represent the concentrations of *hedgehog* and *Engrailed*, respectively, at time t , and τ_{En} , n and k are parameters. The first term of the RHS of eq. 8.2 generates an “S”-like shaped response curve, by which increasing concentration of *engrailed* leads to faster *hedgehog* production rate – but the effect levels off eventually. The second term represents a standard first-order degradation process by which *hedgehog* is removed from the system, so that in the absence of any stimulated production by *Engrailed*, *hedgehog* concentration will eventually tend to 0. Further details of a full mathematical model are described in appendix §H.

Coupled ODE systems like these are too complex to be solved analytically, but simulations and analysis of simplified versions indicate a key fact: the *inherent* spatial dynamics of the coupled network can create ordered spatial patterns. Intuitively, a gene being on in one cell influences other genes through the regulatory dynamics to turn off; this in turn induces genes in neighboring cells to turn on, which effects other genes in those cells, &c. For a large range of parameters and initial conditions (though *not* all), the cell on-off patterns will converge to spatially interlocking consistent stable states, corresponding to the observed expression. Even though any individual cell only has direct access to information about neighboring cells, inter-cell interactions force stable equilibria at correct the global pattern.

This realization suggests several profound biological questions:

- Are there generic network structure (“genotypic”) features that translate into pattern (“phenotypic”) features?
- What changes in the network would need to be made to achieve *other* patterns? Conversely, what other patterns are accessible with small modifications?
- For what are the networks optimized (if anything)? For robustness? or efficiency? or perhaps flexible evolvability?¹ Is there a theoretical tradeoff between these objectives, and if so, are actual systems close to optimal?

Answering these questions would provide a link between a fundamental phenotypic observable (the spatial expression pattern), the basic genotypic structure (the regulatory network), and the developmental constraints on their co-evolution. In the next section, I show how treating the embryo as a spatial multi-agent system and applying the techniques from previous chapters may afford the beginnings of a useful approach.

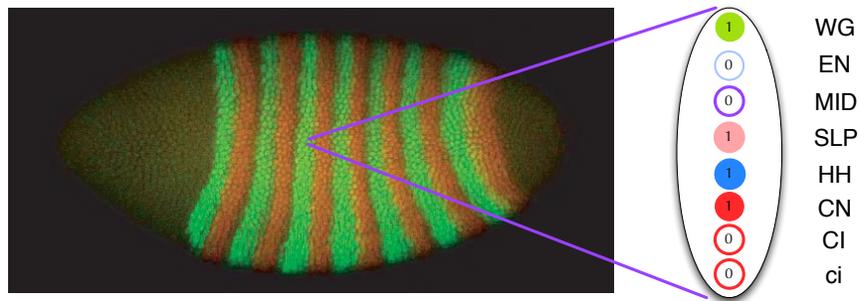
8.2.2 Multi-Agent Model

In contrast to the ODE model described above, here we model the *Drosophila* embryo as a spatial multi-agent system.² The model consists of **cell-agents**, connected in a **organism graph**, operating according to **local regulatory program rules**. Specifically:

¹Robustness means: small changes to the network do not change the final pattern much. Efficiency means: the minimal number of nodes or inter-node connections are used to achieve a given pattern. Evolvability means: the network structure is built so that typical network changes are biased towards creating a variety of useful new patterns.

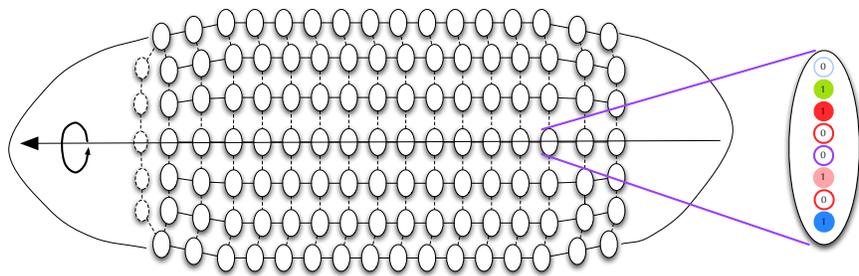
²The *Drosophila* stripes first begin to appear at a stage before that which we model here, when cell boundaries are not yet well defined. Once cell boundaries are invaginated, intercellular interactions significantly sharpen and stabilize the stripes. Stripe localization in the pre-cellular stage is probably mediated by discrete and regularly spaced nuclei [17], but this question is outside the scope of the current investigation.

1. **Cells correspond to agents.** Using the setup from §6.1.2, the agents each carry several binary-valued slots. A slot corresponds to a specific gene, and the slot's binary state corresponds to that gene's being on or off. The "tuple" of slot-states in a single agent represents the ensemble of on/off states in the cell's gene expression. Pictorially:



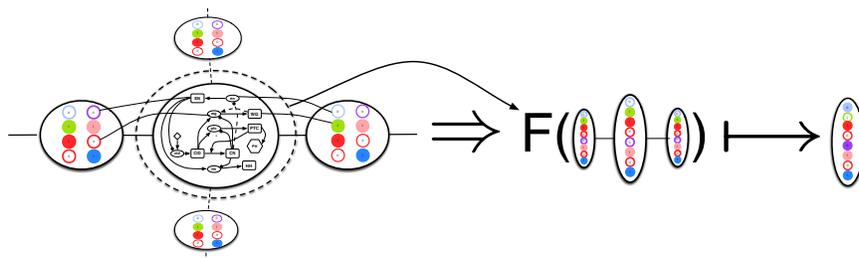
The total number of states is 2^k , where k is the number of gene slots. This model of states is similar to a variety of boolean gene network models [14].³

2. The *organism* consists of agents connected in a **cylindrical two-dimensional graph**:



This structure corresponds to the fact that at this developmental stage, the cells of the *Drosophila* embryo lie on the *two-dimensional surface* of the three-dimensional ellipsoidal embryo [17].

3. **The gene regulatory network corresponds to a nearest-neighbor local rule** updating the agents' states:



The network graph, in other words, somehow generates a finite state automata that computes *outputs* (the state of the cell) from local *inputs* (states of neighbors). This notion of the regulatory network as an input-output transducer follows [6]. Because the stripe pattern is rotationally symmetric around the anterior-posterior axis of the embryo, we will assume that the update rule is rotationally decoupled, allowing us to treating the organism as a cylinder of parallel one-dimensionally spatially coupled gene networks.⁴ Notice that we've bypassed the level of differential equations models and gone straight from the network influence graph to the state update rule.

³Whether it is appropriate to treat cell states, which in reality are molecular concentrations, with a discrete on/off binary variable, is an open question [14].

⁴Two- and three-dimensional effects are relevant when modeling the stripe formation processes biophysically – to make diffusion constants, &c, work out dimensionally. At the level we model it here, these effects can probably be ignored.

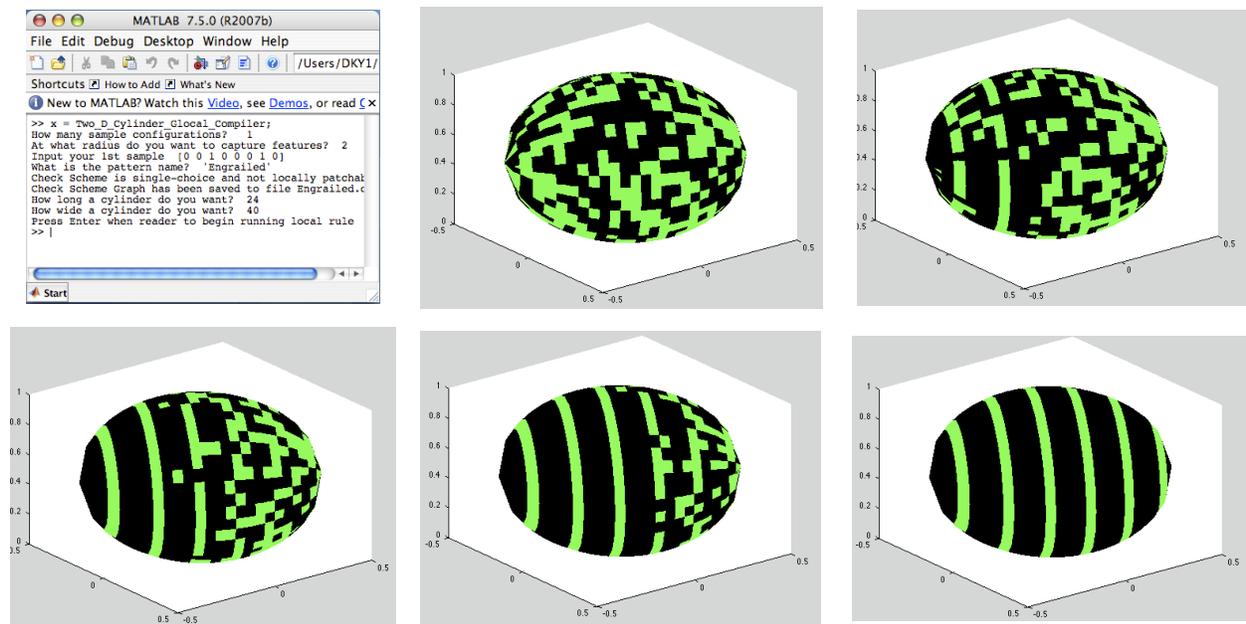


Figure 8.3: Global-to-local compiler implemented on the two-dimensional cylindrical organism graph. For the rotation of the T_{0010} pattern, the compiler constructs a single-choice radius-3 solution. Shown here are selected timesteps from the trajectory of this rule operating on a randomly chosen initial condition.

8.2.3 Analysis

Let's focus first on the *engrailed* gene. It's pattern is to repeat Off-Off-On-Off, in a 4-cell period. In terms of the multi-agent model, this is the repeat pattern T_{0010} .

Suppose we isolated the *engrailed* portion of the network, and allowed it to operate only by its own endogenous dynamics. A very simple local checkability argument shows that such dynamics could stably produce the observed pattern. Specifically:

- The *engrailed* pattern T_{0010} has check scheme radius 2, and would require a radius of at least $5/2$ to solve robustly.⁵
- However, nearest-neighbor local rules have $r = 1$, falling short of the required minimum.

To see what this means physically, think back to the differential equations model described in equations 8.1 and 8.2. A local check in the multi-agent model corresponds to a locally interlocking *stable steady state* of the differential equation. The simple local check argument above is an *abstraction* of the complex underlying dynamics, implying that *no equation of the form 8.1 and 8.2 can have a non-degenerate stable solution $\frac{dS_i}{dt} = 0, \forall i, j$ of the observed form, if the engrailed terms are decoupled from the others.* (See appendix §H for more details.) In network graph terms, several non-trivial edges must feed out and in of *engrailed* in a feedback loop. So even if we knew nothing about the actual network, the local check concept helps us predict a very simple network feature.

Let's carry this analysis one step further. What does the theory of previous chapters tell us about what a two-state rule that *did* create T_{0010} would look like? Like all repeat patterns, T_{0010} is a single-choice pattern. Hence, the simple gradient construction f_{Θ} will yield a radius-3 solution with two states. Using the global-to-local compiler described in the previous section, it is simple to implement this solution in the two-dimensional cylindrical geometry, are shown in fig. 8.3.

⁵Meaning that a cell would have to receive direct information from at least one of its distance-3 neighbors.

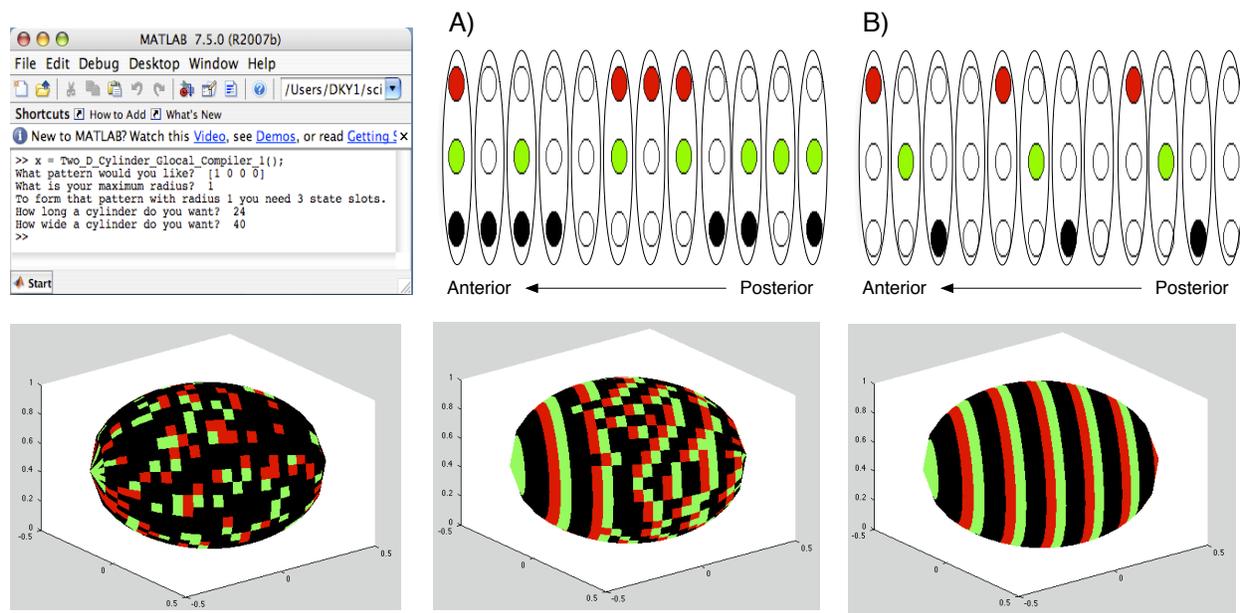
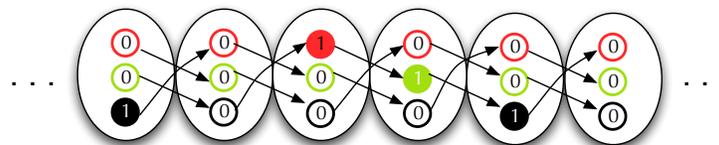


Figure 8.4: Global-to-local compiler implementing a solution to the T_{0010} pattern with an $r = 1$ nearest-neighbor rule, using the radius-state tradeoff algorithm. The generated rule uses three state slots. Panel A) shows a portion of an anterior-posterior cross section of a randomly chosen initial condition. B) shows the final state in the same cross section after the rule converges. The lower panels show a whole-organism view at three timesteps along the trajectory.

However, f_{Θ} requires any given cell agent to potentially receive direct information from a cell three steps away. How would we be able to implement this rule in a nearest neighbor system? The obvious answer from the theory of chapter 6 is to make a radius/state tradeoff. That is, we should add state slots, corresponding to other genes. The newly added states will need to have their update depend on the original state and vice versa, establishing a feedback relationship which resolves the correct pattern. To achieve a nearest-neighbor ($r = 1$) rule, we need to find a radius-1/2 check scheme Θ for T_{0010} – since the f_{Θ} construction doubles the radius of check scheme.

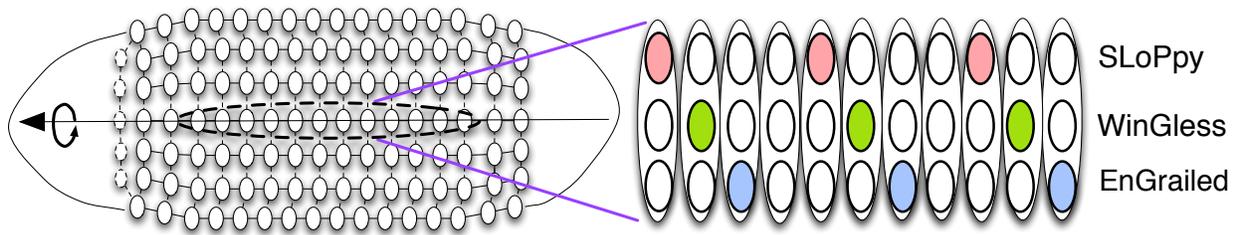
We are thus faced with the question of how many states we need to add, and what the feedback relationships should look like. Luckily the “cut-and-shift” radius \rightarrow state tradeoff algorithm in §6.1.2 gives us a standard way to do this. Applying this algorithm to the problem of checking T_{0010} with radius 1/2 check, we obtain a check scheme using three slots:



Intuitively, what the three state slots do is feed forward the result of one slot on to the next slot – in the next cell – until each of the three slots is used, and feedback closes the loop spatially. Figure 8.4 illustrates the global-to-local compiler implementing this check scheme and the nearest-neighbor rule created from it.

The result of the compiler and cut-and-shift algorithm can be viewed as a “null-hypothesis” prediction for what a nearest-neighbor network that can stably form the T_{0010} will look like. The obvious question is to determine which aspects of this prediction are consistent with the actual structure of the *Drosophila* segment polarity network.

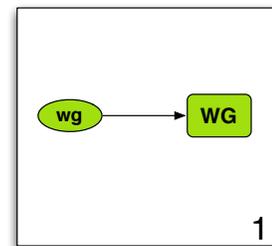
Focus on the three proteins *engrailed* (*En*), *wingless* (*Wg*), and *SLoPpy* (*slp*). If we highlight the expression profiles of these three, we see:



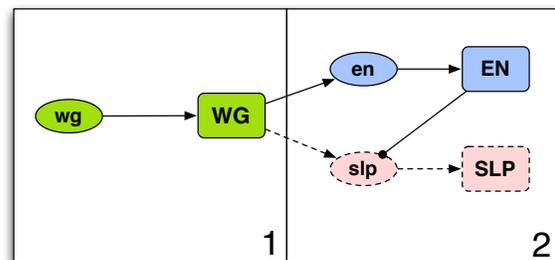
The expression structure is consistent with what would be expected if these proteins were playing the role of the various shifts in the local check scheme produced by the “cut-and-shift” algorithm. But for the expression profile to be meaningful, we have to delve deeper and see whether the actual regulatory relationships confirm the picture. Moreover, if these three *do* form a cut-and-shift local check, this raises the question of what role the *other* genes play.

Specifically, we need to isolate a path within the SPN network that puts these three proteins in a stable spatial feedback sequence. In what follows, I describe a simple chain of steps that identify such a path. Each step summarizes a differential-equations stable steady-state computation, using the specific edge structure of the network in Figure 8.2. For more mathematical details on the meanings of these steps, please see appendix §H.

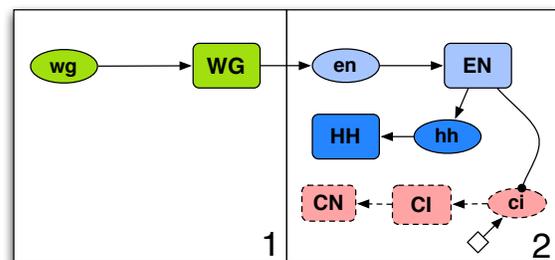
Step 1: Suppose that we began with a cell expressing *wingless*, but none (or only a small amount) of the others:



Step 2: Suppose also that the next cell to the right, Cell 2, is expressing *engrailed* at a high enough level to so that the bistable activation from *wg* Cell 1 system remains expressing *en* and suppressing *sloppy*:

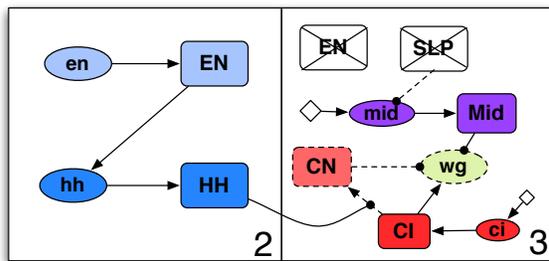
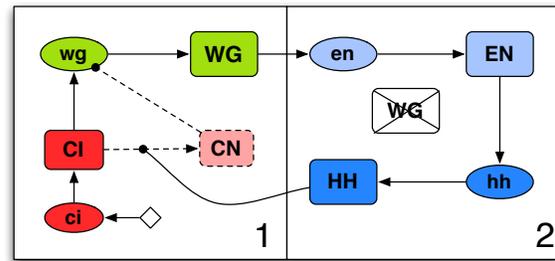


Step 3: The *en-slp* negative feedback relationship in Cell 2 will stabilize *en* expression. This in turn activates *hedgehog* and overcomes basal expression to suppress *cubitus interruptus*, its protein, and repressor fragment (CN):



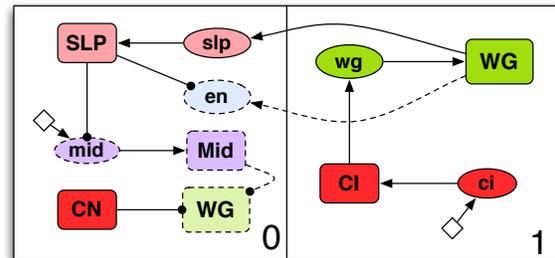
Step 4: In Cell 2, the lack of *CI* means that no *wg* will be produced so any residual *wg* will degrade^a. However, the presence of *hh* also inhibits the cleavage of *CI* into *CN* in Cell 1, which in turn stabilizes the expression of *wg* in Cell 1:

^aAnd be suppressed by *Mid*, although here this link is unnecessary and could be used for other things.

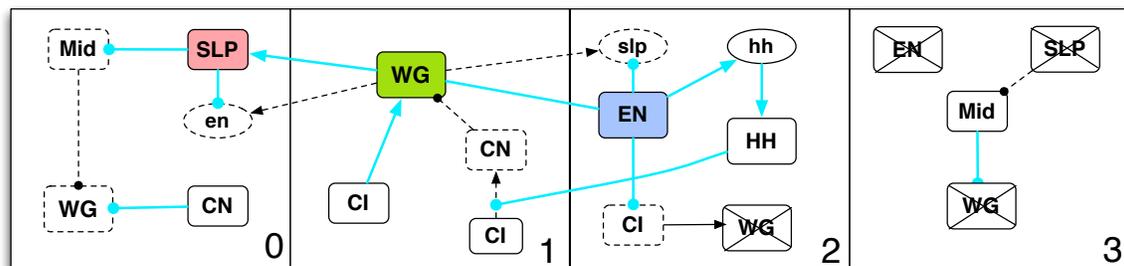


Step 5: Meanwhile, over in the *next* right cell, Cell 3, the lack of *wg* from cell 2 will cause any residual small amount of *en* and *slp* to degrade. Hence, *ci* will be transcribed, so *CI* will be expressed, and the *ci* → *wg* link be active. Since *hh* expresses in Cell 2, *CI* will not be cleaved. However, *Mid* will be being expressed (since *slp* is not), so *wg* is suppressed.

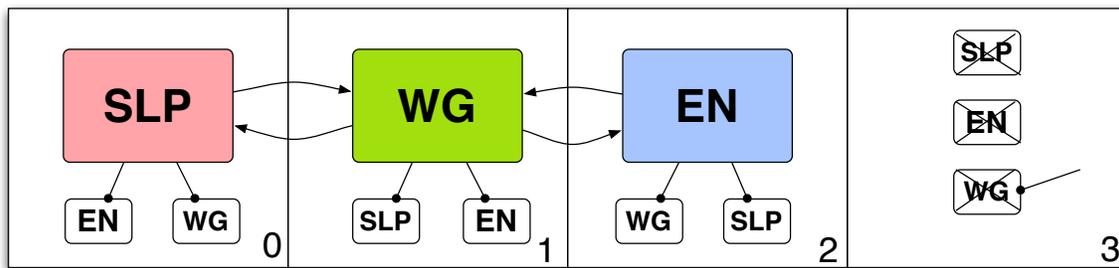
Step 6: Shifting attention back to Cell 1, if the cell to the left had a lower initial expression of *en*, then *Wg* from Cell 1 will favor the *slp* expression equilibrium and residual *en* will die out. Moreover, *mid* will be suppressed, but *wg* suppression will be stabilized by *CN* expression:



Refocussing attention on the three original proteins, and highlighting the active links that directly contribute to pattern functionality, we have:



Finally, consolidate the “intermediate” interactions between *slp*, *wg*, and *en*. This reduces to:



This is the interaction structure underlying the cut-and-shift local check scheme. So, because it picks not just the correct pattern but also the local structural generator of the pattern, the cut-and-shift mechanism might therefore be thought of as part of a “rationalization” of the structure of the segment polarity network. We have made some small progress toward answering the first of the three “profound” biology questions mentioned that the end of §8.2.1, the essential result being that:

The coordination of multiple input-output state relationships that unfold over space and time allows cellular agents with single-cell views to generate a long-range coordinate pattern

The analysis also begins to address questions of designability and optimality.

Because of its structural generality, the cut-and-shift mechanism indicates how the network might be modified to create other patterns. To modify the pattern phenotype to have extend to additional cell, it would suffice to cut the link between last slot and the first, connect in a new “last” slot, and link that up to the first slot. Implementing this change with regulatory elements is straightforward conceptually, requiring the addition of two new network elements (one to take the role of the new slot, and the other to mediate the cutting of the link between the old last slot and the first slot). However, building this in practice may be difficult because it would involve identifying and integrating several new proteins with very specific interactions.

The analysis also shows that the segment polarity network is minimal in one regard but not in another. All the elements of the network are used in the cut-and-shift pathway, and removing any one would prevent correct operation. Moreover, the smallest possible network that could implement the cut-and-shift pathway would require, for each of the three target state slots, one to promote the next state distally in the neighboring cell and two to repress the other states proximally in the current cell, as shown in the “consolidated” figure. The actual segment polarity cut-and-shift pathway – the bright blue arrows in the “non-consolidated” figure – does not use significantly more than this minimal number. It does, however, use more *nodes* than is minimally necessary – all the eight of the nodes of the network are utilized in the putative cut-and-shift pathway, while (as in the consolidated figure) in theory the cut-and-shift pattern could be achieved with just three proteins. But this node-minimal version would require each protein, by virtue of its own biophysics, to have one effect across a cell membrane and another within the cell, without any mediation by interactions with other proteins. This is very hard to imagine biochemically. In the actual network, the several extra nodes essentially act as “helpers” which *reverse* protein effects when moving across the membrane. In summary, it might be said that the SPN appears close to edge-minimal but not node-minimal.

In regard to robustness the situation is somewhat subtle. Each of the steps the regulatory path is fairly parameter-robust, depending in some parameter regime only on the sign of the relationship (i.e. its being up- or down-regulation). Moreover, the fact that the cut-and-shift mechanism is a *check scheme* means that, locally, the system is robust to changes in initial conditions – i.e. the pattern is a stable attractor of the system that is unique within some reasonably large basin of attraction. On the other hand, it is not guaranteed that the system will converge to this basin from all starting points. Moreover, it is probably the case that there are large classes of perturbations to which the *Drosophila* embryo is not robust, since a diffuse pre-pattern in *wingless*, set up by other gene regulatory networks acting earlier in development, seems to be empirically required for the segment polarity network to function.⁶ So while there is a local robustness, there may not

⁶It is definitely known that genetic mutations that lead to drastic mistakes in the *wingless* pre-pattern are not corrected during the

be complete global robustness.

This distinction can be understood by comparing the difference between the static concept of “local checkability” – i.e. the ideas of chapter 2 – and the dynamic local rule that is used to generate that a check scheme – the ideas of chapters 3 and 4. Local checkability defines local structural correctness, but more structure is required to control the specific dynamic paths from disorder to order. While the preceding analysis suggests that SPN is implementing (in large part) a cut-and-shift check scheme Θ , it does not address whether any specific local rule is in force. Dynamic analysis at that level is an important next step of this work. Most starkly, the real biological pathway is probably *less* robust than theoretically possible – compared, for example, to the *completely robust* rule constructed from the local check scheme by the global-to-local compiler as shown in Fig. 8.4.

If all these ideas are borne out in the long run, we may summarize by saying that the *Drosophila* segment polarity network is partially optimized for each of the main optimality criteria – robustness, flexible evolvability, minimality – but apparently not completely pushed toward any one extreme optimum. However, there are many caveats regarding the analysis presented here. The influence graph model could be wrong, for example, or perhaps some other pathway is primarily responsible for the expression pattern; or perhaps some other motif aside from the cut-and-shift radius-state tradeoff is a more complete explanation of overall network dynamics [35, 12]. Laboratory experimentation – especially comparative studies in closely related *Drosophilid* species to explore the evolutionary predictions – is the necessary next step if models of this kind are to be truly validated.

Finally, it is useful to note that in this analysis, we are essentially using a high level discrete-state “local check scheme” abstraction to perform an intelligent directed search for steady state of the lower-level differential equation model (compare with the exhaustive search technique used in [24]). It would be reasonably straightforward to automate such analysis and search large numbers of networks from other related organisms for similar structures.

stage in which the SPN usually works [17]. However, the structure of the SPN itself is probably also modified by these genetic changes, so it's hard to know exactly what the bounds of stability are. The needed experiment, which has not been done due to its enormous technical difficulty, is to perturb the pre-pattern in real time in wild-type flies at various levels of disturbance to probe the exact limits of initial condition robustness.

Chapter 9

Conclusions and Future Work

9.1 Discussion

In this thesis, I developed the beginnings of a theory of design and analysis for local-to-global algorithms in spatial multi-agent systems. First, I identified those patterns that are robustly self-organizable in one dimension via the concept of local checkability, and characterized them in graph-theoretic terms (Chapters 2 and 7). I then solved the inverse problem, using the idea of a self-organized distributed Turing machine to generate robust local rule solutions (Chapter 3). Next, I used smart protocols to bring to bear the detailed structure of locally checkable properties to optimize the performance of those rules (Chapter 4). I then developed a measure of global order in multi-agent systems to analyze the inherent limits on how fast local rules can create structures (Chapter 5). Next, I analyzed resource usage properties of local check schemes, and demonstrated a radius/state resource tradeoff (Chapter 6). Finally, I combined these techniques to address applications of local-to-global theory from an engineering and scientific perspective (Chapter 8).

9.2 Future Work

Three main aspects of future work are: generalization, application, and deeper analysis.

9.3 Generalization

One of the key questions is whether the concepts and techniques developed in the previous chapters apply beyond the one-dimensional model. Initial exploration indicates that, to a large extent, they do.

The Modeling Framework: The spatial model defined in §1.1, and used throughout the rest of this thesis, is based on the “One Dimensional Directed Finite Lattices” – the graphs L_n defined at the beginning of §1.1. Having introduced the basic spatial structure in terms of a graph, all further definitions (local balls, local rule dynamics, timing models, patterns, robust solutions &c) were defined in graphical terms. A useful consequence of this modeling choice is that generalizing the model from one dimension to other more sophisticated spaces is essentially trivial: simply replace the graphs L_n with graphs describing different spaces, and all the definitions of Chapter 1 automatically go through.

For example, consider replacing L_n with the *undirected* line graphs U_n , defined by

$$U_n = (V_n, E_n) = (\{1, \dots, n\}, \{\{1, 2\}, \{2, 3\}, \dots, \{n-1, n\}\}).$$

Local rules on configurations over U_n are functions from local balls in the configuration graphs over U_n . Unlike balls over L_n , these balls *do not* have any local orientation information, so the local rule functions defined on them will be blind to directionality. Similarly, if we instead use the *ring graphs* R_n defined by:

$$R_n = (V_n, E_n) = (\{1, \dots, n\}, \{(1, 2), (2, 3), \dots, (n-1, n), (n, 1)\}),$$

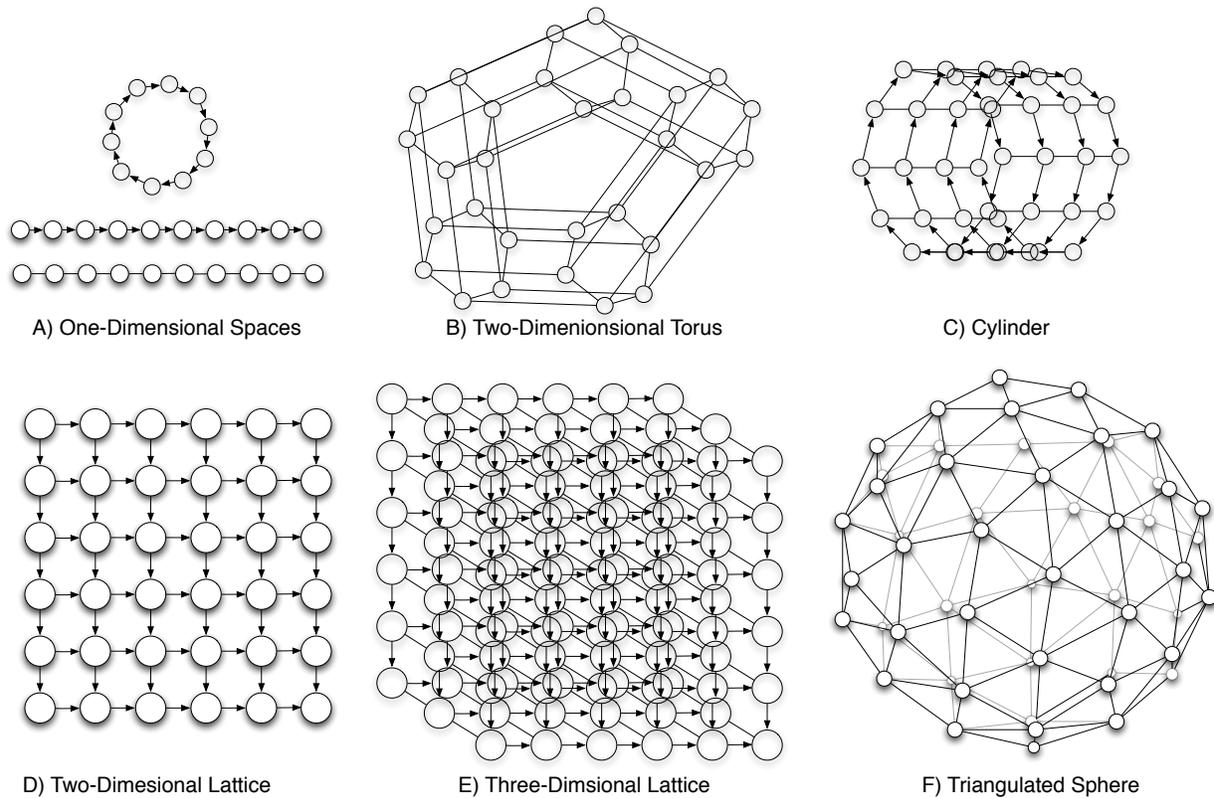


Figure 9.1: A variety of graphically modeled spaces that can be “plugged in” to the agent framework discussed in Chapter 1, including torii, cylinders, two- and three-dimensional lattices, and the triangulated sphere

the resulting configurations will have no well-defined “end” agents. Higher-dimensional geometries are also easily defined (fig. 9.1). Local rules are defined in such spaces, because local ball neighborhoods are defined in the graph structure. All the graphs shown in fig. 9.1 are reasonably highly-structured and “regular” (though not in the graph-theoretic sense). To the extent that spatial structure is lost we move closer to traditional distributed computing (see e.g. [23]) and further from the distinct territory of spatial multi-agent systems, though using approximately regular spaces may be great utility. The problem of graph self-assembly, in which the self-organization problem is actually to construct the underlying space itself, is of great interest and an area of active research [15].

Local Checkability: The concept of local checkability makes sense for spaces other than the one-dimensional directed line, simply by using the same definition of local check scheme (def. 18) with different underlying graphs. More importantly, the proof of proposition 5 requires very few geometric properties of the directed line graphs. Hence, the result that local checkability is necessary for robust solvability is probably very general, so future research on local check schemes in more complex spaces is worthwhile.

As in the one-dimensional case, the analog of repeat patterns on regular spaces are locally checkable, while the analog of proportionate patterns are not. Panels A and B of Fig. 9.2 show a locally checkable two-dimensional repeat pattern and a non-locally checkable proportionate pattern, respectively. Thus, some of the intuition we developed about one-dimensional locally checkable patterns holds for other spaces. However, there are some extremely important differences. While all one-dimensional local check schemes Θ always have some underlying periodicities, local check schemes in more complex geometries need not have any inherent periodicity whatever. For example, the well-known fractal “Sierpinski Gasket” pattern as shown in panel C) of Figure 9.2, is locally checkable with radius 1 but has no inherent periodicity.

Another difference is more unintuitive, and potentially of great use. In one dimension, it is easy to see

that all local encodings of one-dimensional locally checkable patterns are again locally checkable (this underlies the radius/state tradeoff in Chapter 6). No pattern in one dimension that simply could not be locally checked with any radius for a fixed amount of state suddenly becomes checkable with more state. However, in more complex geometries, the situation is *much* more flexible. For example, consider the “cross pattern” in which agents on the diagonals are in one state (blue) and the others in another state (white), as shown in panel D). This pattern is locally checkable with radius 1 and two states. Applying the encoding function which sets the state of any agent whose radius-1 neighborhood contains four or more blue states to red, the cross pattern is encoded into a pattern with a single red region the very center, as shown in panel E). This “center-marked” pattern is *not* locally checkable in two states with any finite radius but *is* the local encoding of a checkable pattern. The center-marked pattern, and similar constructions, can iteratively serve as the “hidden scaffold” for a huge variety of meaningful and complex structures. This includes proportionate and circular patterns, among many others (see panels F-I). Exploring the pattern design possibilities of local check schemes in higher dimensions is rich area of future work.

Local Rule Constructions: Just as local checkability can be generalized to spaces beyond the one-dimensional lattice, it seems likely that the local rules to construct those check schemes can also be significantly generalized. Suppose, for example, that a two-dimensional local check scheme is “single-choice.” Just as in one dimension, this means that in any direction, a two-dimensional local block can be extended by at most one choice of states that are consistent with the local check scheme. (The Sierpinski Gasket corresponds to a two-dimensional single choice check scheme, for example.) In this case, the *two-dimensional gradient* is well defined, and can be implemented almost identically to the one-dimensional case described in §3.1, with radius twice that of the check scheme. Figure 9.3 shows a variety of complex locally checkable patterns being constructed by generalizations of the rules from chapters 3 and 4, in the ellipsoidal geometry described in the Introduction. Exploring and developing such rules further is an important direction for future work.

The techniques of chapters 5,6, and 7, also have potential to be generalized. Fourier analysis can easily be generalized to many different spaces to capture periodic structure, while generalize autocorrelation analysis will be useful to quantify non-periodic ordered structures. DeBruijn graphs can also be associated to any underlying space graph, by taking the set of nodes to be the local configurations of a fixed radius in the underlying space, and the edges determined by spatial proximity. While in the one-dimensional case the canonical DeBruijn structures are the cycles corresponding to periodic patterns, the subgraphs corresponding to higher dimensional check schemes are more complex and suggest that interesting and useful geometric questions may be asked (see figure 9.4).

9.3.1 Application

Future lines of research also involve several possible applications. There are three main areas of application which I envisage as being especially relevant:

1. The results of Chapter 8 in analyzing *Drosophila* suggest a variety of biological experiments to pursue. Spatial multi-agent modeling in general, and the specific concepts of this thesis, may provide useful guide for an experimental program in the study of the evolution-development connection.
2. The algorithms and concepts developed here may be useful for programming distributed self-assembly tasks. The ability to input a “building plan” into a global-to-local compiler and obtain as output rules that local agent-based blocks or robots can use to construct the building, especially given resource-tradeoff capabilities, may enable new approaches to distributed construction.
3. The ideas developed to analyze and engineer spatial multi-agent systems may provide a useful “agent-based” perspective on the theories of computation, formal languages, and cellular automata. Exploring the implications of local check schemes, the self-organized Turing machine, and global order measures, and especially their higher-dimensional analogs, may open new areas with these more traditional disciplines.

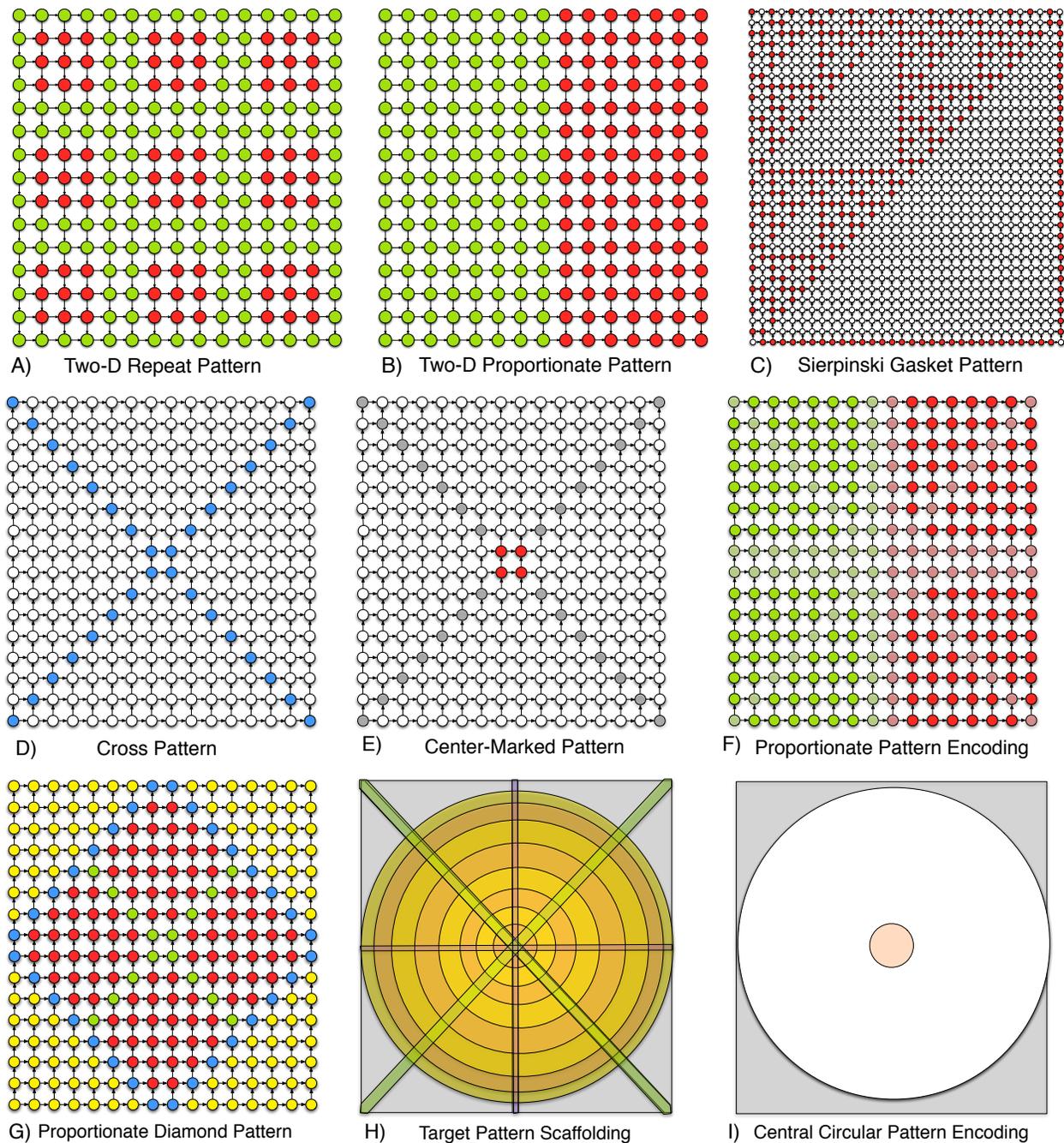


Figure 9.2: A) Two-D lattice repeat pattern. This pattern is locally checkable with radius 3. B) Non-locally checkable 2-D lattice proportionate pattern. C) The Sierpinski gasket pattern is locally checkable with radius 1 but is aperiodic. D) The cross pattern. E) The Center-Marked pattern, using the cross pattern as an encoding scaffold (grayed-out). F) The same proportionate pattern from B), now produced as an encoding of a locally checkable pattern (grayed out). G) Another 2-D lattice pattern resulting from a local encoding of a checkable pattern. H) In a two-D continuous (or Amorphous) environment, a local check scheme that is useful to scaffold the Proportionate Circular Pattern shown in I). By combining circular and proportionate patterns, arbitrarily complicated vector spline images can be synthesized from local check schemes in two- or higher-dimensional spaces.

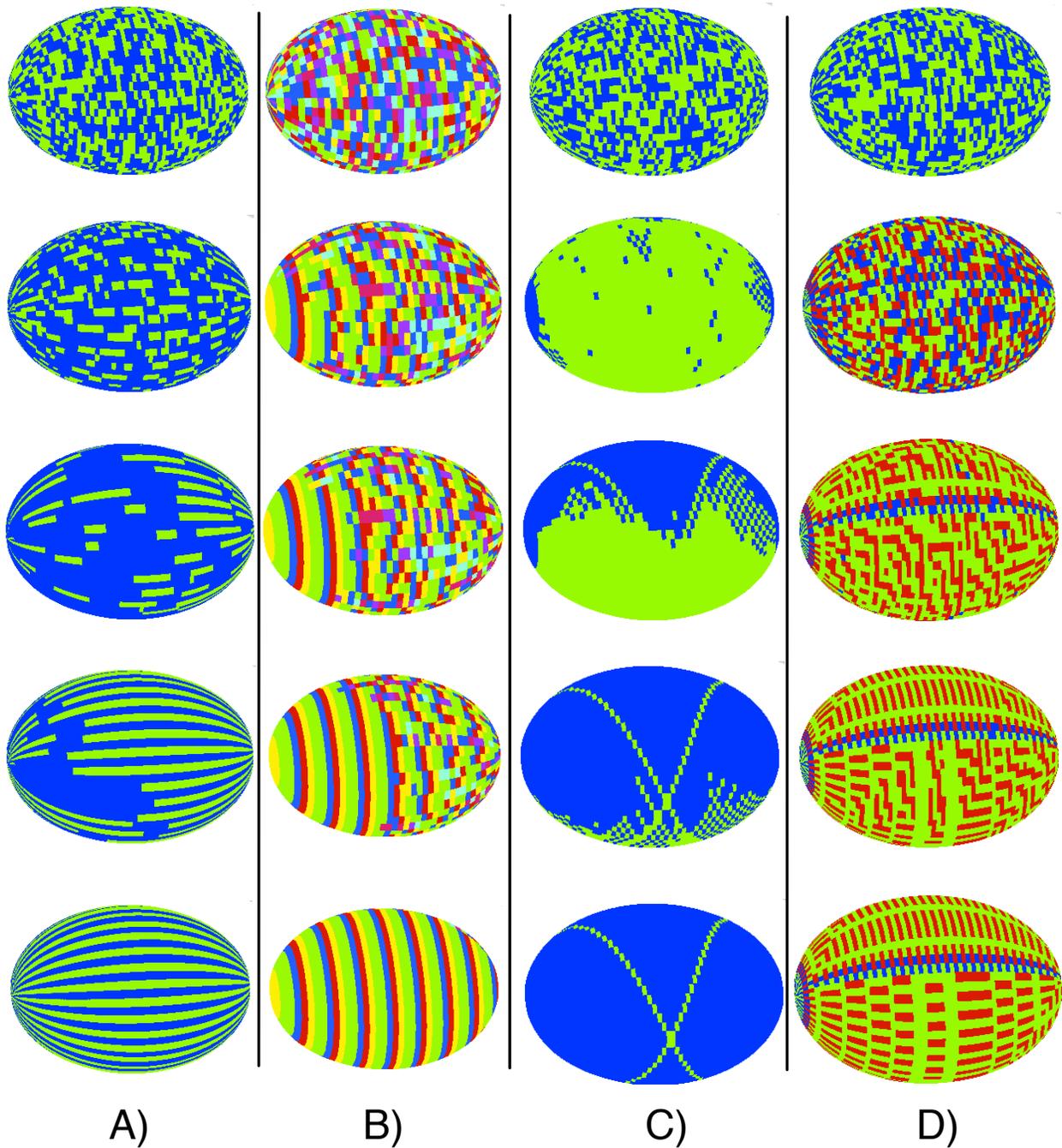


Figure 9.3: Trajectories of local rules constructing check schemes defined on two-dimensional ellipsoidal geometries. A) The “Watermelon” pattern. B) Vertical stripes pattern. C) The cross pattern (same pattern as in Fig. 9.2D, in this geometry). D) The “Football” pattern, which involves first self-organizing a “circumference” and then patterning the two hemispheres with orthogonal “grip” patterns.

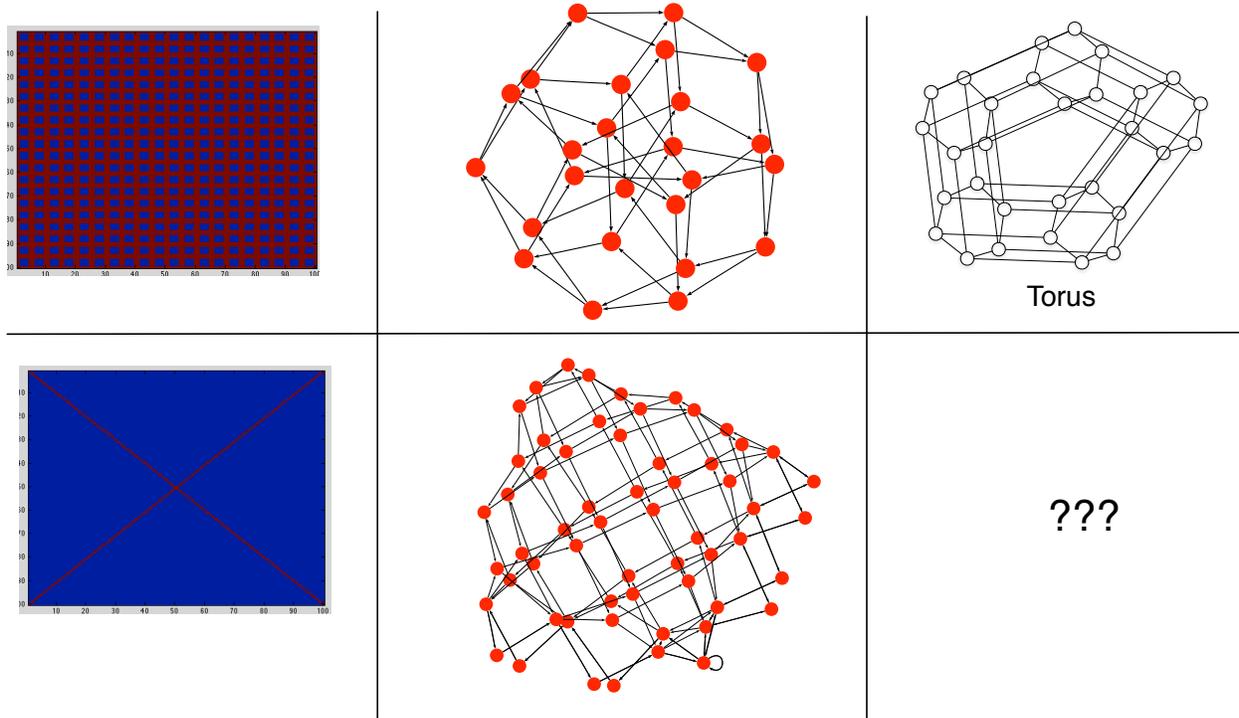


Figure 9.4: Two-dimensional local check schemes and their corresponding DeBruijn subgraphs. Top: The DeBruijn graph for the pattern shown in Fig. 9.2A is a 5-by-5 torus. Bottom: The DeBruijn graph for the Cross pattern check. I don't yet know a standard model for its topology.

9.3.2 Deeper Analysis

Finally, there are several questions about the one-dimensional model which bear further investigation. These include:

- More thoroughly understanding the self-organized coherent structures inherent in any local rule, their effect on computation, and connection to the theory of Cellular Automata “particles” mentioned at the end of chapter 3.
- Developing techniques for constructing more sophisticated, non-locally checkable patterns from specialized initial conditions. For example, given a 1-dimensional proportionate pattern, what is largest set of initial conditions on which a local rule can form it? And what “local-rule” engineering techniques could be used?
- Addressing questions of dynamic patterns, beyond static “picture-formation” tasks explored here; more generally, developing a theory of “whole trajectory” properties, not just properties that apply to the “final state” alone.
- Further understanding the topological and geometric structure of DeBruijn space, and
- Developing a theory of the robustness of local rules to systematic perturbations in their structure. For example: given a pattern T , which robust local solution F to T produces the least error when the look-up table defining F is slightly modified? (Or are all robust solutions, at a given resource level, roughly equivalent in this regard?)

Bibliography

- [1] H. Abelson et al. Amorphous computing. *Comm. ACM*, 43(5), 2001.
- [2] H. Abelson, G. Sussman, and J. Sussman. *Structure and Interpretation of Computer Programs*. The MIT Press, 1996.
- [3] W. Butera. *Programming a Paintable Computer*. PhD thesis, MIT, 2002.
- [4] S. Camazine, J. L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organizing Biological Systems*. Princeton Univ. Press, 2001.
- [5] J. Conway. The game of life. *Scientific American*, March 1970.
- [6] Eric Davidson. *The Regulatory Genome*. Academic Press, 2006.
- [7] N. DeBruijn. A combinatorial problem. *Indagationes Math.*, 8, 1946.
- [8] S. Eilenberg. *Automata, Languages, and Machines*. Academic Press, 1973.
- [9] Phillipe Flageolet. *Algebraic Combinatorics*. Self-Published Online, 2007.
- [10] H. Fredricksen. A new look at the de bruijn graph. *Discrete Applied Mathematics*, 37, 1992.
- [11] S. W. Golomb. *Shift Register Sequences*. Aegean Park Press, 1981.
- [12] N T Ingolia. Topology and robustness in the *Drosophila* segment polarity network. *PLoS Biology*, 2(6):805–815, June 2004.
- [13] J M Kahn, R H Katz, and K S J Pister. Mobile networking for smart dust. In *Proc. ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99)*, 1999.
- [14] S. Kauffman. *The Origins of Order*. Oxford Univ. Press, 1993.
- [15] E. Klavins. Directed self-assembly using graph grammars. In *Foundations of Nanoscience*, 2004.
- [16] M. Kloetzer and C. Belta. Hierarchical abstractions for robotic swarms. In *Proc. IEEE ICRA 06*, 2006.
- [17] Peter Lawrence. *The Making of a Fly*. Wiley, 1992.
- [18] H. Lewis and C. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall, 1997.
- [19] W. Li. Power spectrum of regular languages and cellular automata. *Complex Systems*, 1989.
- [20] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.
- [21] N. Linial, Y. Mansour, and M. Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the Association for Computing Machinery*, 40(3), 1993.
- [22] M. Lothaire. *Algebraic Combinators on Words*. Cambridge University Press, 2005.
- [23] N. Lynch. *Distributed Algorithms*. Morgan Kaufman, New York, 1996.

- [24] W. Ma et al. Robustness and modular design of the *Drosophila* segment polarity network. *Molecular Systems Biology*, 2006.
- [25] J. D. McLurkin. Stupid robot tricks: A behavior-based distributed algorithm library for programming swarms of robots. Master's thesis, MIT, 2004.
- [26] R. McNaughton and S. Papert. *Counter-Free Automata*. MIT Press, 1971.
- [27] M. Mitchell, J. Crutchfield, and R. Das. Evolving cellular automata to perform computations: A review of recent work. In *Proc. of the First International Conference on Evolutionary Computation and Its Applications*. Russian Academy of Sciences, 1996.
- [28] G. Myerson. How small can a sum of roots of unity be. *The American Mathematical Monthly*, 93(6), 1986.
- [29] R. Nagpal. *Programmable Self-Assembly: Constructing Global Shape Using Biologically-Inspired Local Interactions and Origami Mathematics*. PhD thesis, MIT, 2001.
- [30] A. Ralston. De bruijn sequences – a model example of the interaction of discrete mathematics and computer science. *Mathematics Magazine*, 55(3), 1982.
- [31] C. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Proc. SIGGRAPH*, 1987.
- [32] D. Rus et al. Self-reconfiguration robots. *Communications of the ACM*, 45, March 2002.
- [33] J. v. Neumann. *The Theory of Self-reproducing Automata*. U. Illinois Press, 1966.
- [34] G von Dassow, E Meir, E Munre, and G M Odell. The segment polarity network is a robust developmental module. *Nature*, 406:188–192, July 2000.
- [35] G von Dassow and G M Odell. Design and constraints of the *Drosophila* segment polarity module: Robust spatial patterning emerges from intertwined cell state switches. *J. of Experimental Zoology*, 294:179–215, 2002.
- [36] G. Werner Allen et al. Monitoring volcanic eruptions. In *Proc. EWSN 05*, 2005.
- [37] S. Wolfram. *Rev. Mod. Phys.*, 55, 1983.
- [38] L. Wolpert. Positional information. *J. Theor. Bio.*, 25(1), 1969.
- [39] D. Yamins. *Structural Organization Theory*. Harvard University Undergraduate Thesis, 2002.
- [40] D. Yamins. In *Proceedings of the 2005 Conference on Autonomous Agents and Multi-Agent Systems*, 2005.

Appendix A

Local Balls and Parts

A.1 Counting Ball Types

The following question is very simple to answer, and turns out to be very useful: for any fixed n and r , what are the isomorphism classes of balls of radius r in L_n ? And, what can an agent determine about its position and the size of the configuration merely from observing the structure of its local neighborhood?

There are several cases. If $n \leq r$ – what we’ll term “very small” balls – then for each $i \in \{1, \dots, 2r + 1\}$, the r -ball around position i is a graph with n nodes, and a \star at one of the nodes. For example, if $n = 3$ and $r = 3$, the r -balls in L_n are:

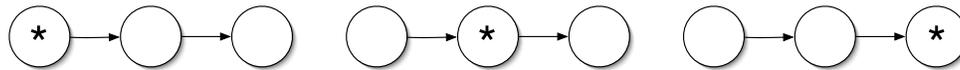


Figure A.1: The balls of radius 3 in L_3 . Stars denote the “middle” agent in each case.

Notice that each ball is unique, i.e. no two positions yield the same ball, and moreover that each agent can derive from the structure of the graph that $n = 3$. (Notice also that without the \star 's, it would be impossible to distinguish these balls, which is why I introduced the \star in the first place.)

If $r + 1 \leq n \leq 2r$ – the “small” balls – then for all $i \leq \lfloor n/2 \rfloor$, there are two balls of length $r + i$, one at position i and the other at position $n - i + 1$. For example, if $n = 6$ and $r = 3$:

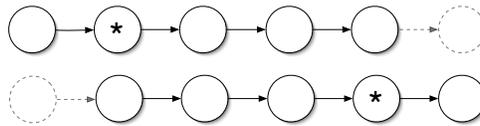


Figure A.2: The radius 3 balls at position 2 and 5 in L_6 .

If n is even this counts all the balls. If n is odd, then there is one ball of length n , with the \star at position $(n + 1)/2$. Again, balls at all positions have unique structure, so each agent can tell what its position is. Without prior knowledge of n , only positions $n - r \leq i \leq r + 1$ can determine n from the structure of ball; these are “small center” agents; those $i < n - r$ know they’re closer to the left and are thereby “left-end” agents, while those with $i > r$ know they’re closer to the right and are thereby “right-end” agents.

Finally, if $n \geq 2r + 1$ – the “large” balls – then: First, for $i \leq r$ there are two balls each of size $r + i$, similar in structure to the left and right-end configurations in the previous case (and we’ll call them the same thing). Then, there are $n - 2r$ “central” balls of size $2r + 1$ whose positions are $r + 1 \leq i \leq n - r$, all of which look the same. For example, if $r = 3$ (and $n \geq 7$), these are:

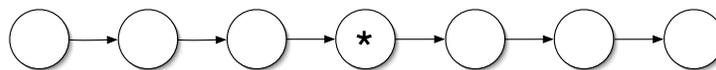


Figure A.3: A radius 3 central ball.

The left- and right-end agents can determine their positions, but the central balls cannot. Unlike in the previous cases where all the balls were unique, now many of the balls will be the same.

The local balls $\mathcal{B}_{r,S}$ are simply versions of each of the above graphs, labeled by elements of S . Hence there are basically four overall classes of balls:

- Central balls, which have size $2r + 1$ – and only arise in configurations of size $\geq 2r + 1$. Denote the central balls $\mathcal{B}_{\text{central}}(r)$. There are m^{2r+1} central balls, where $m = |S|$.
- Left balls, of size between $r + 1$ and $2r$ or less – which arise only in configurations of size $r + 1$ and greater. Denote them $\mathcal{B}_{\text{left}}(r)$. There are $m^{r+1} \sum_{i=0}^{r-1} m^i = m^{r+1}(m^r - 1)/(m - 1)$ of these.
- Right balls, also of size between $r + 1$ and $2r$ or less – and again only arising in configurations of size $r + 1$ and greater. Denote them $\mathcal{B}_{\text{right}}(r)$. There are also $m^{r+1}(m^r - 1)/(m - 1)$ right balls.
- Small central balls, of size between $r + 1$ and $2r - 1$. Denote them $\mathcal{B}_{\text{small}}(r)$. There are $\sum_{i=1}^{r-1} im^{2r-i} = m^{r+1}(m^r - mr + r - 1)/(m - 1)^2$ of these.
- Very small balls, of size between 1 and r . Denote these $\mathcal{B}_{\text{v-small}}(r)$. There are $\sum_{i=1}^r im^i = m(rm^{r+1} - m^r - m^r + 1)/(m - 1)^2$ of these.

Hence, over all there are

$$m^{2r+1} + 2m^{r+1} \frac{m^r - 1}{m - 1} + m^{r+1} \frac{m^r - mr + r - 1}{(m - 1)^2} + m \frac{rm^{r+1} - m^r - m^r + 1}{(m - 1)^2} = m \left(\frac{m^{r+1} - 1}{m - 1} \right)^2$$

local r -balls.

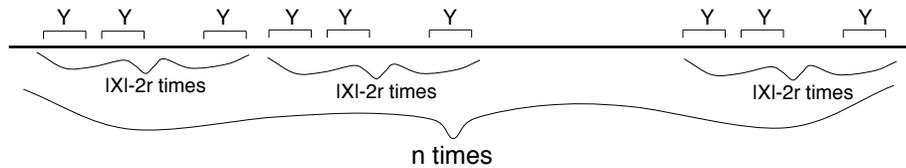
Usually, we'll combine classes 2 and 3 into the “left and right-end balls” and classes 4 and 5 into the “small balls”. We also distinguish two classes of configurations X – the “small configurations” with $|X| \leq 2r$, and the “large” configurations with $|X| \geq 2r + 1$.

A.2 Reduction to Parts

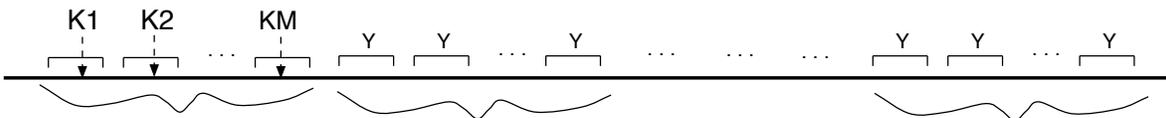
Proposition 47 *Let $T = \Theta(\mathbb{C})$ be a locally checkable pattern, and suppose Y is T -repeatable (or expandable). Then any configuration X all of whose parts are contained in Y is also T -repeatable (or expandable).*

Proof: Suppose Y is Θ -repeatable, and $B_r(X) \subset B_r(Y)$. Then, we have to show that given any integer n , there is some Θ -admissible Z containing (at least) n repeats of X .

The fact that Y is repeatable guarantees that we can find a Θ -admissible configuration with as many repeats of Y as we want. We choose a configuration Z_n that has $n \cdot M$ repeats of Y , where $M = |X| - 2r$. To see why we do this, consider this figure:



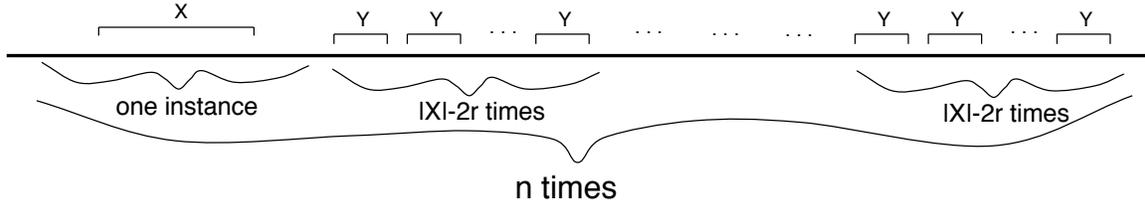
Each of the copies of Y contains at least one instance of each part in X , since by assumption $B_r(X) \subset B_r(Y)$. Write out a list of the r -balls in X , calling the left-most r -ball $b_{1,r}$, the r -ball one step to the right $b_{2,r}$ and so on, defining $b_i = X(i : i + 2r)$, until $i = M$ (the right-most part of X). Now, in Z_n , locate the first instance of the left-most part $b_{1,r}$ in the left-most copy of Y , say at position k_1 . Then locate the first instance of the next part $b_{2,r}$ that occurs in second copy of Y , say at position k_2 . Then locate an instance of $b_{3,r}$ in the third copy of Y . Keep doing this until each one of the M parts of X has been located, at positions k_i :



Excise the points in Z_n between k_i and k_{j+1} , to get

$$Z'_n = Z_n(1 : k_1) \circ Z_n(k_2) \circ Z_n(k_3) \dots Z_n(k_M : |Z_n|) = Z_n(1 : k_1 - 1) \circ \left(\bigcirc_{i=1}^M Z(k_i) \right) \circ Z_n(k_M + 1 : |Z_n|)$$

where $\bigcirc_{i=1}^N a_i$ indicates the concatenation $a_1 \circ a_2 \circ a_3 \circ \dots \circ a_N$. Of course, putting the parts of X next to each other simply makes X , so we're left with



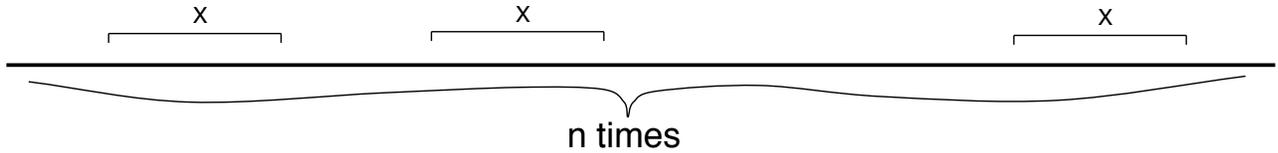
or, formally,

$$Z'_n = Z_n(1 : k_1 - r - 1) \circ X \circ Z_n(k_M + r + 1 : |Z_n|).$$

Now, repeat above process $n - 1$ more times, finding positions k_j for $j = 1, \dots, n(|X| - 2r)$ at which the r -ball around k_j is an instance of b_j , where j is taken modulo $|X| - 2r$ (shifted by 1). That is,

$$Z_n(k_j - r : k_j + r) = b_{1+\text{mod}(j-1, |X|-2r)}.$$

Then excise the positions between the k_j s, for all j that are not multiples of $|X|$, but retaining the parts between $k_{j=|X|-2r}$ and $k_{j=|X|-2r+1}$. To do this each time without disturbing the copy of X just made to the left, we have to assume that the right-most element of the $j(|X| - 2r) + 1$ -st copy of Y is at least distance r away from left-most element of the $j(|X| - 2r)$ -st copy, i.e. $k_{j|X|-2r+1} - k_{j|X|-2r} > 3r$. This way, we're left with:



or, formally,

$$Z'_n = \left(\bigcirc_{j=0}^{n-1} Z_n(k_{jM} + 1 : k_{(j+1)M} - 1) X \right) \circ Z_n(k_{nM} + 1 : |Z_n|)$$

which is what we wanted in the first place. ■

In chapter 1 I also described the T -required set of X , denoted $R_T(X)$. This was defined to be the set of subconfigurations Y such that any T -admissible configuration Z containing X would have also to contain Y :

$$R_T(X) = \{Y \mid \forall Z \in T, X \in Z \Rightarrow Y \in Z\}.$$

Recall that we defined $B_r(X)$ to be the set of parts in X (without regard to order or number of each). Then $B_r(R_T(X))$ is the set of parts of configurations required by X , and, reversing the order, $R_T(B_r(X))$ is the set of configurations required by any configuration that contains the parts of X . We have:

Proposition 48 *A locally checkable pattern $T = \Theta(\mathbb{C})$ satisfies*

$$B_r(R_T(B_r(X))) = B_r(R_T(X)),$$

where $r = r(\Theta)$. That is: "the set of all parts set used in configurations required by the parts of X , without regard to how they're put together, is the same as the set of parts used in configurations required by X itself."

Proof: What we have to show formally is that

$$B_r(R_\Theta(B_r(X))) = B_r(R_\Theta(X)),$$

where

$$R_{\Theta}(B_r(X)) = \{Y \mid \forall Z \in \Theta(\mathbb{C}), B_r(X) \subset B_r(Z) \Rightarrow Y \in Z\}.$$

So suppose that $Y \in \Theta(B_r(X))$. Then any Θ -admissible Z such that $B_r(X) \subset B_r(Z)$ must have $Y \in Z$. Hence a fortiori $Y \in \Theta(X)$ and thus $\Theta(B_r(X)) \subset \Theta(X)$ and thus $B_r(\Theta(B_r(X))) \subset B_r(\Theta(X))$. On the other hand, if $y \notin B_r(\Theta(B_r(X)))$, that means there is a Z with $B_r(X) \subset B_r(Z)$ such that $y \notin B_r(Z)$. Obviously $y \notin B_r(X)$ also. Now, in particular, X_1 and $X_{|X|-2r}$ are in $B_r(Z)$. Define Z_1 to be the segment of Z from its left end upto the first (or any) instance of X_1 ; then define Z_2 be the of Z from the first (or any) of instance of $X_{|X|-2r}$ to its right end. Then $Z_1 X (r : |X| - r) Z_2$ is Θ -admissible and contains X but not y since $y \notin B_r(X)$. Hence y cannot be in $B_r(\Theta(X))$. ■

Appendix B

Naive Backtracking Proofs

B.1 Appendix: Proof of Prop. 14

Proof: In the course of this proof, we use the shorthand definitions that:

- $\alpha_1[B], \alpha_2[B], \alpha_3[B]$, and $\alpha_4[B]$ are radius- $2r + 2$ boolean functions denoting when **Rules 1,3,9** and **10** apply, respectively.
- $\beta_1[B]$ and $\beta_2[B]$ are the booleans denoting when **Rules 5** and **6** apply, respectively.
- $\gamma_1[B]$ and $\gamma_2[B]$ are the booleans denoting when **Rules 2** and **4** apply, respectively.
- $\delta[B]$ is the boolean denoting when **Rule 8** applies.
- $\epsilon[B]$ is the boolean denoting when **Rule 7** applies.

Let X be any configuration. Let $B_i = B_{2r+2}(i, X)$. Recall the notation S for the original states and $S' = \{\triangleright, \triangleleft, \Delta_i\}$. Suppose that $B_i(0) \in S$ for all i . Given that $B_i(0) \in S$, we know that $\alpha_{3,4}[B_i], \beta_{1,2}[B_i], \gamma_{1,2}[B_i]$, and $\epsilon[B_i]$, must all not hold, since they require $B_i(0) \notin S$. Moreover, $\beta_3[B_i]$ and $\delta[B_i]$ cannot hold since they would require $B_{i+1}(0) = \triangleleft$ and $B_{i+1}(0) = \Delta_{B_i(0)}$ respectively. Now suppose in addition that $F[B_i] \in S$ for all i as well. Then, $\alpha_1[B_i]$ and $\alpha_2[B_i]$ cannot hold either, since they would require $F[B_i] = \triangleright$. Now, if $\alpha_1[B_i]$ fails to hold then since $B_i(0) \in S$ by assumption, either: a) $\star(B_i) \neq |B_i|$, that is, i is not the end agent, or b) $\star(B_i) = |B_i|$, i.e. i is the end agent. In case a) either $\Theta^-[X[i - 2r - 1 : i - 1]]$ or $\Theta[X[i - 2r : i]]$; if the former, then let j the maximum $l < i$ such that $\Theta[X[i - 2r : j]]$ (such a j must exist since it is vacuously true for $j = 0$), and we'll have $B_{j+1}(0) \in S, \Theta_{-r-1}(B_{j+1})$, and $\Theta_r^-(B_{j+1})$; but then $\alpha_1[B_{j+1}]$ whence $F[B_{j+1}] \notin S$, against assumption. In case b) $\star(B) = |B|$, in which case either $\Theta^-[X[|X| - 2r - 1 : |X| - 1]]$ or $\eta[B_{|X|}]$, and for the same reasons as above, the former cannot be true, so $\eta[B_{|X|}]$ must hold. Hence $\Theta[X[i - 2r : i]]$ for all i and $\eta_{B_{|X|}}$, which implies $X \in \Theta(\mathbb{C})$. Thus, if $B_i(0), F[B_i] \in S$ for all i , X must be a fixed point of F and $X \in \Theta(\mathbb{C})$. (Obviously any $X \in \Theta(\mathbb{C})$ satisfies $B_i(0), F[B_i] \in S$ for all i , so $\Theta(\mathbb{C}) \subset \text{fix}(F)$, where $\text{fix}(F)$ is the set of fixed points of F .)

Now, suppose $X_0 \notin \Theta(\mathbb{C})$. Then for some i , either $B_i(0) \notin S$ or $F[B_i] \notin S$. Wlog we can assume the latter since if $B_i(0), B_{i+1}(0) \in S$, then $F(B_i) \neq B_i(0)$ implies $F(B_i) \notin S$; thus, if $B_i(0) \in S$ for all i , the only "live" agents (which will change state if called) will change out of S ; under live call sequence s eventually such an agent will be called. We now have three cases: (I) $B_{j_0}(0) = \Delta_i$ for some i , (II) $B_{j_0}(0) = \triangleright$, or (III) $B_{j_0}(0) = \triangleleft$.

Case (I): $B_{j_0}(0) = \Delta_E$ for some E : [picture] First of all, for all $j < j_0 - 1$, $F(B_j) = B_j(0)$. Now, there are two subcases: if i) $B_{j_0-1}(0) = E$ and $L^-[B_{j_0} - 1]$, then $\gamma_1(B_{j_0} - 1)$, whence

$$F[B_{j_0-1}] = \min\{j \in [0, m - 1] | \Theta[B_{j_0-1}[-2r : -1] \circ j]\},$$

and $F[B_{j_0}] = B_{j_0}$. Hence for any call sequence, $s = (s_1, \dots, s_k, \dots)$, $F^k(X)[1 : j_0] = X[1 : j_0]$ unless $j_0 - 1 \in s_i$ for some $i \leq k$. Given a call sequence s let s_{k_0} be the first s_i such that $j_0 - 1 \in s_i$ (which must exist since the timing model is assumed to be live). Then

$$X^{k_0} \triangleq F_s^{k_0}(X)[1 : j_0] = X[1 : j_0 - 2] \circ \zeta \circ X[j_0] \circ Y$$

where $\zeta = F[B_{j_0-1}]$, a state necessarily $> E$ and Y is some configuration of length $|X| - j_0$. All agents j in X_{k_0} with $j < j_0$ are fixed, and α_3 applies to $B_{2r+2}(j_0, X^{k_0})$. So let s_{l_0} be the first call in s with $l_0 > k_0$ containing j_0 , and

$$X^{l_0} = X[1 : j_0 - 2] \circ \zeta \circ \triangleright Y'$$

where Y' is some configuration of length $|X| - j_0$. But the first $j_0 - 1$ agents are in S , so X_{l_0} is then in case (II) below. On the other hand, if (ii) $B_{j_0-1}(0) \neq E$ or $L[B_{j_0} - 1], F[B_j] = B_j(0)$ for all $j < j_0$, and already α_3 applies to B_{j_0} , so X^{l_0} is in case (II), where l_0 is (like above) the minimal time l at in which s_l contains j_0 .

Thus all configurations in case (I) eventually end up in case (II) below, with the same j_0 .

Case (II): $B_{j_0}(0) = \triangleright$. First we will state and prove a simple lemma. Let b be an r -ball. The set of extensions of b in Θ , denoted $ext_{\Theta}(b)$, is the set of all subconfigurations-with-right-ends Z such that $\Theta(b'_i)$ holds for all $b'_i = B_r(i, b \circ Z)$ for $i \geq \star(b)$. The set of *incomplete* extensions $ext_{\Theta}^i(b)$ is the set of all subconfigurations Z (without reference to right-ends) such that $b'_i = B_r(i, b \circ Z)$ for $\star(b) \leq i \leq |Z| + |b| - r$.

Lemma 2 *For all configurations X of the form*

$$X = Y \circ \triangleright \circ Z$$

such that $\Theta[B_r(i, Y)]$ for all $i < |Y| - r$, then for all live call sequences s over X there is a $k > 0$ such that for all $l \leq k$, and agents $i \leq |Y|$, F fixes i in $F_s^l(X)$, and either

- $F_s^k(X) = Y \circ W$, with $Y \circ W \in \Theta(\mathbb{C}) \subset \text{fix}(F)$, whenever $ext_{\Theta}(Y[|Y| - 2r : |Y|])$ contains some element (like W) of size $|Z| + 1$, OR
- $F_s^k(X) = Y \circ \triangleleft \circ Z'$, for some configuration Z' of size $|Z|$, whenever $ext_{\Theta}(Y[|Y| - 2r : |Y|])$ does not contain an element of size $|Z| + 1$.

In the latter case, the trajectory $\{X^l = F_s^l(X) | l \leq k\}$ is such that the set $\{X^l[|Y| + 1 : |Y| + j] | j \leq |Z|\}$ contains all elements of $ext_{\Theta}^i(Y[|Y| - 2r : |Y|])$ of size $|Z|$ or less (in lexicographic order).

Notice that taking $|Y| = 0$ in this lemma implies that for all configurations of the form $X = \triangleright \circ Z$, there is k such that either $F^k(X)$ is solved whenever $ext_{\Theta}(\emptyset)$ contains an element of length $|X|$, i.e. for all such initial conditions, F solves X whenever X is a solvable size.

Proof: (Of Lemma) We will now prove the statement by induction on n , the size of $|Z|$. Let's first do the base-case $n = 0$. Then $X = Y \circ \triangleright$, and the only active agent in X is $|X|$ itself, since X satisfies Θ for all agents in Y (and thus up to $|X| - 1$). Let $B = B_{2r+2}(|X|, X)$. Obviously either $\beta_1[B]$ or $\gamma_2[B]$. In the former case, there is state j such that $Y \circ j$ is a Θ -satisfying configuration, and for each live call sequence s , after the first call $k(s)$ containing $|X|$, $F^{k(s)}(X) = Y \circ j'$, the minimum possible such j . Evidently $j \in ext_{\Theta}(Y[|Y| - 2r : |Y|])$, and we're in possibility 1 of the $T(0)$ statement, with $W =$ the singleton subconfiguration containing only j' . If $\gamma_2[B]$, after the first call $k(s)$ containing $|X|$, $F^{k(s)}(X) = Y \circ \triangleleft$. $\gamma_2[B]$ will only hold if for no j is $Y \circ j$ a Θ -satisfying configuration, and evidently $ext_{\Theta}(Y[|Y| - 2r : |Y|])$ does not contain an element of size 1, so we're in possibility 2 of the base-case.

Now suppose that the statement of the lemma has been demonstrated for all sizes $i \leq n - 1$. Suppose $X = Y \circ \triangleright \circ Z$, where $|Z| = n$, and $\Theta[B_r(i, Y)]$ for all $i < |Y| - r$. Let $B_j^k = B_{2r+2}(j, X^k)$ where $X^k = F^k(X)$. Let $J = |Y| + 1$, the location of the \triangleright between Y and Z . All agents $i < J$ are fixed by F in X . If $X(J + 1) \neq \triangleright$, then agent J is fixed as well, while $F[B_{j+1}] = \triangleright$. In this case, after the first time t_0 at which $J + 1$ is called, then $X^{t_0} = F^{t_0}(X) = Y \circ \triangleright \circ \triangleright \circ Z'[2 : |Z|]$. In X^{t_0} , agents $i \leq J + 1$ are all fixed by F except for agent J . Now, there are two cases: i) there is no j such that $X[J - 2r : J - 1] \circ j$ satisfies Θ , in which case $B_j^{t_0}$ satisfies β_1 , and $F[B_j^{t_0}] = \triangleleft$. In this case, we're done, in the second possibility of the statement $T(n)$. If ii), there is j such that $\Theta[X[J - 2r : J - 1] \circ j]$, then $\gamma_1(B_j^{t_0})$ and $\zeta_1 \triangleq F(B_j^{t_0}) = \min\{j | \Theta[X[J - 2r : J - 1] \circ j]\}$. Thus

$$X^{t_0+1} = Y \circ \zeta_1 \circ \triangleright Z''[2 : |Z|].$$

Now, X^{t_0+1} fits into the inductive hypothesis, so at some later time t_1 either $X^{t_1} = Y \circ \zeta_1 \circ W$, with $Y \circ \zeta_1 \circ W \in \Theta(\mathbb{C})$ if $ext_{\Theta}(Y[|Y| - 2r + 1 : |Y|] \circ \zeta_1)$ has an element of size $|Z|$, in which case, the we're done and in the first

case of $T(n)$; or $X^{t_1} = Y \circ \zeta_1 \circ \triangleleft \circ \widetilde{Z}_1$, when $\text{ext}_\Theta(Y[|Y| - 2r + 1 : |Y|] \circ \zeta_1)$ has no element of size $|Z|$. Throughout all agents $i \leq J$ have been fixed by F . Now, suppose A) $L(B_j^{t_1})$, that is,

$$\zeta_1 \neq \max\{j|\Theta[X[J - 2r : J = 1] \circ j]\}.$$

In this case, $\beta_2(B_j^{t_1})$ holds, so that at the next time $t_2 > t_1$ when J is called, $X^{t_2} = Y \circ \triangleleft \circ \widetilde{Z}_2$, and we're done (in possibility 1 of $T(n)$). On the other hand, if B) $L^\neg(B_j^{t_1})$, that is, $\zeta_1 \neq \max\{j|\Theta[X[J - 2r : J = 1] \circ j]\}$, then $F[B_{j+1}^{t_1}] = \Delta_{\zeta_1}$, and after the first time $t_2 > t_1$ calling $J + 1$, $X^{t_2} = Y \circ \zeta_1 \circ \Delta_{\zeta_1} \circ \widetilde{X}_2$. As in the analysis in Case (I) in the main body of the proof, after the first call at time $t_3 > t_2$ to J , $X^{t_3} = Y \circ \zeta_2 \circ \Delta_{\zeta_2} \circ \widetilde{X}_3$, where $\zeta_2 = \min\{j > \zeta_1 | \Theta[X[J - 2r : J = 1] \circ j]\}$. The inductive hypothesis again applies to X^{t_3} , so at time t_4 either $X^{t_4} = Y \circ \zeta_2 \circ W_2$, having been solved; or $\text{ext}_\Theta(Y[|Y| - 2r + 1 : |Y|] \circ \zeta_2)$ has no element of size $|Z|$, so $X^{t_4} = Y \circ \zeta_2 \circ \triangleleft \circ \widetilde{Z}_3$. Again the two cases A) or B) apply; repeating the same reasoning, we eventually reach such time where either the configuration is solved, or case A) applies, and the induction is complete. ■

Back to case (II), when $B_{j_0}(0) = \triangleright$. In this case $X = Y \circ \triangleright \circ Z$, satisfying the conditions of the lemma. Hence for some k , either $F^k(X) = Y \circ W$, a solution in Θ , or $F^k(X) = Y \circ \triangleleft \circ Z'$, whenever there is no solution to Θ of size $|X|$ that extends Y , and we're in Case (III) below.

Case (III): $B_{j_0}(0) = \triangleleft$. There are three subcases: i) $j_0 \neq 1$ and $L^\neg(B_{j_0-1})$, ii) $j_0 \neq 1$ and $L(B_{j_0-1})$, and iii) $j = 1$. In case i) all agents $j \in \{1, \dots, j_0 - 1\}$ are fixed by F , $F[B_{j_0}] = \Delta_{B_{j_0-1}(0)}$. At the first timestep where j_0 is called, call it l_0 like above,

$$X^{l_0} = X[1 : j_0 - 1] \circ \Delta_{X[j_0-1]} \circ Y$$

where Y is some configuration of size $|X| - j_0$. But then we're in case (I)i from above, so that eventually for $l_1 > l_0$,

$$X^{l_1} = X[1 : j_0 - 2] \circ \zeta_1 \triangleright \circ Y'$$

and thus in case (II) above, with the same j_0 . In light of the argument there, for some $l_2 > l_1$ either X^{l_2} is solved, or if $X[1 : j_0 - 2] \circ \zeta_1$ cannot be extended,

$$X^{l_2} = X[1 : k_0 - 2] \circ \zeta_1 \triangleleft \circ Y',$$

and we're in the same case (III). If i) holds again, the same argument yields a $\zeta_2 > \zeta_1$, and the reasoning repeats. Eventually if the configuration is not solved, and we keep returning to (III), i) can hold at most $m - 1$ total times before $L(B_{j_0-1})$ must hold, in which case $L(B_{j_0-1})$, and we're in the next case ii). In case ii), then for all $j \in \{1, j_0\}$ except $j_0 - 1$, F fixes the state of j , while $F[B_{j_0-1}] = \triangleleft$. At the first timestep where $j_0 - 1$ is called, say k_0 ,

$$X^{k_0} = X[1 : j_0 - 2] \circ \triangleleft \circ \triangleleft \circ Y.$$

But then X^{k_0} is in this same case (III), with $j_0^{k_0} = j_0 - 1$. Iterating this reasoning, either X is eventually solved, or eventually for some K , we repeat through case (III) ii) enough times so that $j_0^K = 1$, i.e. we're in case iii) below. In case iii), $\alpha_4[B_1]$ applies, and defining l_0 again to the first timestep where j_0 is called,

$$X^{l_0} = \triangleright Z$$

for some Z . This configuration satisfies the conditions of the lemma with $|Y| = 0$; thus as noted above, there will be k such that $F^k(X)$ is solved if there is a solution of size $|X|$. ■

B.2 Appendix to §3.3

The goal of this appendix is to prove propositions 15, 16, and 17.

Proof: (Of prop. 15). Let $\text{ext}_n^{\Theta,*}(Y)$ denote the set of extensions of Y other than Y itself. We will show by induction on $|X| - |Y|$ that:

- If $\text{ext}^\Theta(Y)$ contains a configuration of size $|X|$, $TTS(F_\Theta^O, X) \leq \max(4 \cdot |\text{ext}_n^{\Theta,*}(Y)|, 1)$.

- If $ext^\Theta(Y)$ does not contain a configuration of size $|X|$, for some time T such that $|ext_n^\Theta(Y)| \leq T \leq \max(4 \cdot |ext_n^{\Theta,*}(Y)|, 1)$, $(F_\Theta^O)^T(X) = Y \circ S \circ Z'$ where $S = \Delta_{Y(|Y|)}$ or \triangleleft .

The base case is that $|Y| = |X| - 1$, i.e. $X = Y \circ \triangleright$. Then, if Y has Θ -admissible extension of size X , that means there is a state s such that $Y \circ s$ is Θ -admissible. F_Θ^O will have the right-end agent adopt this state within one timestep, so $TTS(F_\Theta^O, X) = 1 \leq \max(4 \cdot |ext_n^{\Theta,*}(Y)|, 1)$. If Y has no Θ -admissible extension of size $|X|$, then either the state of the last agent in Y , $Y(|Y|)$, is the largest relative to O or not; in the former case, within one timestep, F_Θ^O will have the right-end agent take the state \triangleleft , and in the latter, $\Delta_{Y(|Y|)}$. So, $T = 1$. Now $|ext_n^{\Theta,*}(Y)| = 0$ and $|ext_n^\Theta(Y)| = 1$, so $|ext_n^\Theta(Y)| = 1 = T = \max(|ext_n^{\Theta,*}(Y)|, 1)$, and we have the result.

Now, suppose $X = Y \circ \triangleright \circ Z$ for $|Z| > 1$. If $ext_n^{\Theta,*}(Y)$ is empty, then $ext^\Theta(Y)$ cannot contain a configuration of size $|X|$; again, $T = 1$, $|ext_n^{\Theta,*}(Y)| = 0$ and $|ext_n^\Theta(Y)| = 1$, so $|ext_n^\Theta(Y)| = 1 = T = \max(|ext_n^{\Theta,*}(Y)|, 1)$. On the other hand, suppose $ext_n^{\Theta,*}(Y)$ is non empty. Let $B_Y = Y[|Y| - 2r - 1 : |Y|]$, and $out(B_Y)$ be the set of all nodes $o \in G(\Theta)$ for which (B_Y, o) is an edge in $G(\Theta)$. Enumerate the nodes of $out(B_Y) = \{o_1, \dots, o_k\}$ in the order that O assigns them, where k is the out-degree of B_Y in $G(\Theta)$. After s timesteps, with $1 \leq s \leq 2$,

$$X_1 \triangleq (F_\Theta^O)^{s_1}(X) = Y \circ o_1 \circ \triangleright \circ Z'.$$

Now we proceed in steps:

1. suppose $ext^\Theta(Y \circ O(B_Y, 1))$ contains a configuration of size $|X|$. The inductive assumption tells us that

$$TTS(F_\Theta^O, X_1) \leq \max(4 \cdot |ext_n^{\Theta,*}(Y \circ o_1)|, 1)$$

so that

$$TTS(F_\Theta^O, X) \leq 2 + \max(4 \cdot |ext_n^{\Theta,*}(Y \circ o_1)|, 1).$$

Now, the key point is that

$$|ext_n^{\Theta,*}(Y)| = k + \sum_{o_i \in out(B_Y)} |ext_{n-1}^{\Theta,*}(Y \circ o_i)|.$$

Since

$$out(B_Y) + \sum_{o_i \in out(B_Y)} |ext_{n-1}^{\Theta,*}(Y \circ o_i)| \geq 1 + |ext_{n-1}^{\Theta,*}(Y \circ o_1)|,$$

so that since $|ext_n^{\Theta,*}(Y)| > 0$,

$$2 + \max(4 \cdot |ext_n^{\Theta,*}(Y \circ o_1)|, 1) \leq 4|ext_n^{\Theta,*}(Y)| = \max(4 \cdot |ext_n^{\Theta,*}(Y)|, 1).$$

2. On the other hand, now suppose $ext^\Theta(Y \circ O(B_Y, 1))$ does not contain a configuration of size $|X|$. First suppose $k = 1$. Then applying the inductive assumption, for some T_1

$$(F_\Theta^O)^{T_1+s}(X) = Y \circ o_1 \triangleleft \circ Z'$$

where

$$|ext_{n-1}^\Theta(Y \circ o_1)| \leq T \leq \max(4|ext_{n-1}^{\Theta,*}(Y \circ o_1)|, 1).$$

Hence

$$X_2 \triangleq (F_\Theta^O)^{T_1+s+1}(X) = \begin{cases} Y \circ \triangleleft \circ Z'', & \text{if } Y \text{ is } O\text{-maximal} \\ Y \circ \Delta_{Y(|Y|)} \circ Z'', & \text{otherwise} \end{cases}.$$

Since

$$T_1 + s + 1 > 2 + |ext_{n-1}^\Theta(Y \circ o_1)| \geq |ext_n^\Theta(Y)|$$

and

$$T_1 + s + 1 \leq 3 + \max(4 \cdot |ext_{n-1}^{\Theta,*}(Y \circ o_1)|, 1) \leq \max(4 \cdot |ext_n^{\Theta,*}(Y)|, 1)$$

we have our result.

3. If $k > 1$, then

$$(F_{\Theta}^O)^{T_1+s}(X) = Y \circ o_1 \triangle_{o_1} \circ Z'$$

instead. Hence after two timesteps

$$(F_{\Theta}^O)^{T_1+s+2}(X) = Y \circ o_2 \triangleleft \circ Z''.$$

Now we return to step 1, and go through reasoning again.

Repeating the above steps at most k times, we find that there are T_l such that

$$|ext_{n-1}^{\Theta}(Y \circ o_l)| \leq T_l \leq \max(4|ext_{n-1}^{\Theta,*}(Y \circ o_l)|, 1)$$

for which

- if $ext^{\Theta}(Y)$ contains a configuration of size n , then if we let j be the minimum l for which $ext^{\Theta}(Y \circ o_l)$ contains a configuration of size n , then

$$TTS(f, X) \leq s + 2(j-1) + \sum_{l=1}^j T_l$$

- while if $ext^{\Theta}(Y)$ contains no configuration of size n , then for $T = 1 + s + 2(k-1) + \sum_{l=1}^k T_l$

$$(F_{\Theta}^O)^T(X) = \begin{cases} Y \circ \triangleleft \circ Z'', & \text{if } Y \text{ is } O\text{-maximal} \\ Y \circ \triangle_{Y(Y)} \circ Z'', & \text{otherwise} \end{cases}.$$

But

$$1 + s + 2(k-1) + \sum_{l=1}^k T_l \geq 2k + \sum_l |ext_{n-1}^{\Theta}(Y \circ o_l)| \geq |ext_n^{\Theta}(Y)|$$

and

$$1 + s + 2(k-1) + \sum_{l=1}^k T_l \leq 1 + 2k + \sum_l \max(4|ext_{n-1}^{\Theta,*}(Y \circ o_l)|, 1) \leq 3k + 1 + \sum_l 4 \cdot |ext_{n-1}^{\Theta,*}(Y \circ o_l)| \leq \max(4|ext_n^{\Theta,*}(Y)|, 1).$$

■

Proof: (of prop 16)

Proposition 16 consists of three separate claims:

1. There are constants A and B such that $Ag_{O,\Theta}(n) + B \cdot n \leq TTS^{avg}(F_{\Theta}^O)(n)$. d
2. There are constants C and D such that $TTS^{avg}(F_{\Theta}^O)(n) \leq Cg_{O,\Theta}(n) + D \cdot n$.
3. There are constants E and F such that $TTS^{worst}(F_{\Theta}^O)(n) \leq E \cdot (\sum_{l=1}^n g_{O,\Theta}(l)) + F \cdot n$. d

I will demonstrate 1 first, then 3, and then 2.

Claim 1: Given a local check scheme Θ of radius r over m states, by prop. 11, the set $Sizes(\Theta(\mathbb{C}))$ is of the form

$$Sizes(\Theta(\mathbb{C})) = \left(\bigcup_{i=1}^{M_{\Theta}} A(a_i^{\Theta}, d_i^{\Theta}) \right) \cup B_{\Theta}$$

where $A(a, d)$ is the arithmetic progression with initial value a and common difference d ; B_{Θ} is a finite set with maximal element $m^{(2r+1)m^{2r}}$, and $a_i^{\Theta}, d_i^{\Theta} \leq m^{2r}$ for each i (so that $M_{\Theta} < m^{4r}$). For any $H \subset G(\Theta)$, $Sizes(\widehat{H})$ satisfies the same result:

$$Sizes(\widehat{H}) = \left(\bigcup_{j=1}^{M_H} A(a_j^H, d_j^H) \right) \cup B_H$$

where $\max(B_H) < m^{(2r+1)m^{2r}}$, $a_j^H, d_j^H \leq m^{2r}$ for all j ; and the added proviso that all for all j , $a_j^H = a_i^\ominus$ for some i , and d_j^H is divisible by d_i^\ominus for some i . Hence, $S(H) \triangleq \text{Sizes}(\Theta(\mathbb{C})) - \text{Sizes}(\widehat{H})$ is also, ignoring a finite set with maximal element $m^{(2r+1)m^{2r}}$, a union of arithmetic progressions whose initial values and common differences are less than m^{2r} .

Now let A be the set of configurations X such that (i) $|X| \in S(H_{O,\Theta}(P^*))$ and (ii) $X = P^* \circ \triangleright \circ Y$ for some subconfiguration-with-left-end Y . Let $A_n = A \cap \mathbb{C}_{\leq n}$. Let $n^- = \max\{m < n \mid m \in S(H_{O,\Theta}(P^*))\}$. Since by definition of P^* , $S(H_{O,\Theta}(P^*))$ is infinite, by virtue of the discussion in the previous paragraph, it contains an arithmetic progression with common difference at most m^{2r} . Hence, $n - n^- \leq m^{2r}$. Then since $|P^*| \leq m^{2r}$,

$$\mu_n(A) \triangleq \frac{|A_n|}{|\mathbb{C}_{\leq n}|} \geq \frac{|\mathbb{C}_{n^-}|}{|\mathbb{C}_{\leq n}|} \cdot \frac{1}{m^{|P^*|+1}} \geq \frac{m^{n-m^{2r}}}{(m^n - 1)/m - 1} \cdot \frac{1}{m^{|P^*|+1}} \geq \frac{1}{m^{m^{2r}+2r+1}}.$$

Hence, $TTS^{avg}(F_\Theta^O)(n) \geq \mu(TTS(F_\Theta^O, X))_{X \in A_n}$.

Now, as noted above, $H_{O,\Theta}(P^*)$ is defined precisely that so that on any configuration $X = P \circ \triangleright \circ Y$, the trajectory of F_Θ^O on X will search through all of the elements of $H_{O,\Theta}(P^*)$, or stop at a solution, before constructing any other elements of $\Theta(\mathbb{C})$. Since for $X \in A_n$ there is no element of $|X|$ in $\widehat{H}_{O,\Theta}(P^*)$, that means for such X , the trajectory F_Θ^O will contain configurations of the form $Z \circ \triangleright \circ Y'$ for every $Z \in \widehat{H}_{O,\Theta}(P^*)$ of size n^- or less. Hence $TTS(F_\Theta^O, X) \geq g_{O,\Theta}(n^-)$.

Now, if $\widehat{H}_O(P^*)$ is infinite, then by prop. 12, there is a $k \geq 1$ and $C(r) \geq \frac{1}{m^{2r+1}}$ such that for all n , $g_O(n) \geq Cn^k$. Hence, $TTS(F_\Theta^O, X) \geq g_O(n^-) > \frac{C(r)}{2}g_O(n^-) + \frac{C(r)}{2} \cdot n^-$. On the other hand, if $\widehat{H}_O(P^*)$ is finite, then by prop. 11, it contains elements of size at most $m^{2r+1} + |P^*|$, and is of size at most $m^{m^{2r+1}}$. After finishing the completing the last element of $\widehat{H}_O(P^*)$ of size n^- or less, the right-moving turing head is at position at most $m^{2r+1} + |P^*|$. However, to complete the configuration it must move at least through the rest of the configuration, taking at least $n^- - m^{2r+1} - |P^*|$ timesteps. Now, $|P^*| \leq m^{2r+1}$. Hence $TTS(F_\Theta^O, X) \geq n^- - 2m^{2r+1} \geq \alpha g_O(n^-) + \beta n^-$ where $\alpha = (1/2)(1/m^{m^{2r+1}})$ and $\beta = 1/2$. Thus, taking $C_1 = \min(C(r)/2, \alpha)$ and $C_2 = \min(C(r)/2, \beta)$, we have

$$TTS_S(f, X) \geq C_1 g_O(n^-) + C_2 n^-$$

for all for $X \in B(P^*) \cap \mathbb{C}_{n^-}$. Since $n - n^- < m^{2r}$, we have, by prop. ref that $g_O(n) \leq \frac{1}{m^{2r+1}} g_O(n^-)$. Hence,

$$TTS_S^{avg}(F_\Theta^O)(n) \geq \frac{C_1}{m^{2r+1}} g_O(n) + \frac{C_2}{m^{2r+1}} \cdot n,$$

so we have the lower bound as desired.

Before moving on to Claim 2 and 3, we state a useful lemma. Let Y be any subconfiguration-with-right-end. Y corresponds to a initial path P_Y in $G(\Theta)$. Let P'_Y denote the acyclic subpath of Y , which is unique, and let Y' denote the configuration associated with Y' . The heart of Claim 2 and 3 is contained in the following sharpening of part 1 of prop. 15:

Lemma 3 *Suppose X is a configuration of size n of the form $X = Y \circ \triangleright \circ Z$, and $\text{ext}^\ominus(Y)$ contains a configuration of size n . Then there is a constant $A(m, r)$ such that*

$$TTS(F_\Theta^O, X) \leq A(m, r) \cdot g_{O,\Theta}(P_Y)(n).$$

Proof: (Of Lemma) The result follows from a detailed analysis of the structure of $H_{O,\Theta}(P'_Y)$.

Remark 1 *For any i.a.p, P , recall the definition of $H_{O,\Theta}(P)$ and $\alpha_{O,\Theta}^P$. A moment's thought reveals that there are two possible cases for the structure of $H_{O,\Theta}(P)$, the second of which breaks into two subcases (for notational ease, we'll drop the O, Θ and P from α and H from hereon):*

1. $\widehat{H} = \text{ext}^\ominus(P)$, in which case $\text{Sizes}(\Theta) - \text{Sizes}(\text{ext}^\ominus(P))$ is infinite.
2. $\widehat{H} \not\subseteq \text{ext}^\ominus(P)$. In this, let β be the final node of α , and $G' = \alpha \cup G(\Theta)_{\geq \beta}$; then $\text{Sizes}(\Theta) - \text{Sizes}(G')$ is finite. This case has two subcases:

- (a) α is acyclic. In this case: for some non-final node β' in α , the O -minimal node in $\text{out}(\beta, G(\Theta))$ is β' . The edge $e(P) \triangleq (\beta, \beta')$ is not in H , and so the cycle $C(P) = \alpha(\beta' : \beta) \circ \beta'$ is therefore not in H . If we let $H' = H \cup \{e(P)\}$, then $\text{Sizes}(\Theta) - \text{Sizes}(H')$ is finite.
- (b) α contains a cycle. In this case, if we let γ be the last instance of a unique node in α , then the O -minimal node in $\text{out}(\gamma, G(\Theta))$ is β , and $\beta = \alpha(i)$ for some $i < |\alpha|$. In fact, α is the line graph $\alpha[1 : i]$ followed by the single irreducible cycle $C(P) = \alpha(i : |\alpha|) \circ \beta$. The edge $e(P) \triangleq (\gamma, \beta)$, and the cycle $C(P)$, are in H . Moreover: enumerate the nodes of the cycle $C(Y')$ as $\{v_1, \dots, v_{|C(P)|}\}$, and for $v_j \in C(P)$ define $W(v_j) = \{w \in \text{out}(v_j) \mid w \succ_{O, v_j} v_{j+1}\}$. For $w \in W(v_j)$, let G'_w be the graph $\alpha[1 : i] \circ C[1 : j] \cup G(\Theta)_{\geq w}$, with the node (v_j, w) added. Then

$$\widetilde{H}(P) \triangleq \widehat{H} \cup \bigcup_{j=1}^{|C(P)|} \bigcup_{w \in W(v_j)} G'_w$$

is a subgraph of $G(\Theta)$ such that $\text{Sizes}(\Theta) - \text{Sizes}(\widetilde{H})$ is finite.

In figure 3.5, graphs 1, 2, 5, and 6 are in case 2.i, while 3 and 4 are in case 2.ii. (None of them are in case 1, since all of those in the figure are P^* , the maximal $H_{O, \Theta}(P)$.)

Now, let's take each of the three cases one at a time. In case 1, we can apply prop. 15 directly to get

$$TTS(F_{\Theta}^O, X) \leq 4|\text{ext}_{n-|Y|+|Y'|}^{\Theta}(P'_Y)| = 4|\widehat{H} \cap \mathbb{C}_{\leq (n-|Y|+|Y'|)}| = 4g_{O, \Theta}(Y')(n - |Y| + |Y'|) = 4 \cdot g_{O, \Theta}(Y)(n)$$

and so we have the result in this case.

In case 2.i: Let's compute the trajectory of X under F_{Θ}^O . If $\text{ext}^H(Y)$ contains a configuration of size n , then by 15, $TTS(F_{\Theta}^O, X) \leq 4 \cdot g_{O, \Theta}(Y)(n)$. If $\text{ext}^H(Y)$ does not a configuration of size n , then (by prop. 15) at most $4 \cdot g_{O, \Theta}(Y)(n)$ timesteps, the turing head as searched through all of $H \cap \mathbb{C}_{\leq n}$, and has come around to the end of α . It goes over the edge $e(Y')$; and for some $T_1 \leq 4 \cdot g_{O, \Theta}(Y)(n) + 3$ the configuration is now $X_{T_1} \triangleq (F_{\Theta}^O)^{T_1} = \alpha \circ \beta'(P) \circ \triangleright \circ Z'$. Now, if $\text{ext}^H(\alpha \circ \beta'(Y'))$ contains a configuration of size n , then by prop. 15, $TTS(F_{\Theta}^O, X_{T_1}) \leq 4 \cdot g_{O, \Theta}(Y)(n - |\alpha|)$, so that $TTS(F_{\Theta}^O, X) \leq 4 \cdot g_{O, \Theta}(Y)(n) + 4 \cdot g_{O, \Theta}(Y)(n - |\alpha|) + 3$. On the other hand, if $\text{ext}^H(\alpha \circ \beta'(Y'))$ does not contain a configuration of size n , after at most $4 \cdot g_{O, \Theta}(Y)(n - |\alpha|)$ timesteps, the turing head as searched through all of $\text{ext}_{n-|\alpha|}^H(\beta'(Y'))$, and has come around to the end of α again. It goes over the edge $e(Y')$ again; so for some T_2 with $T_2 \leq 4 \cdot g_{O, \Theta}(Y)(n - |\alpha|) + 3$ the configuration X_{T_2} is now $\alpha \circ C(Y') \circ \beta'(Y') \circ \triangleright \circ Z'$. If $\text{ext}^H(\alpha \circ C(Y') \circ \beta'(Y'))$ contains a configuration of size n , for some $TTS(F_{\Theta}^O, X_{T_2}) \leq 4 \cdot g_{O, \Theta}(Y)(n - |\alpha| - |C(P)|)$; while if there no such extensions, it goes around the cycle again ... Repeating this reasoning, we see that either

$$TTS(F_{\Theta}^O, X) \leq 3l + \sum_{k=0}^l g_{O, \Theta}(Y)(n - k|C(Y')|)$$

or by some $T_l \leq 3(l+1) + \sum_{k=0}^l g_{O, \Theta}(Y)(n - k|C(Y')|)$, the configuration is

$$X_{T_l} = \alpha \circ C(P)^{l-1} \circ \triangleright \circ Z'$$

for some Z' . Since $\text{Sizes}(\Theta) - \text{Sizes}(H')$ (where H' is (as above) $H \cup \{e(P)\}$) is finite, if we take n large enough (bigger than $m^{(2r+1)m^{2r}}$, although probably a better bound can be found), there must be a configuration in \widehat{H} of size n . Now, denote $H_{\geq v} = H \cap G(\Theta)_{\geq v}$. Let

$$H_1 = H_{\geq \alpha_{\leq \beta'}(Y')}$$

and

$$H_2 = H_{\geq \alpha_{\geq \beta'}(Y')}.$$

Note that

$$\text{Sizes}(H') = \text{Sizes}(H_1) \cup (\text{Sizes}(H_2) + |C(Y')| \cdot \mathbb{N}).$$

Hence, n is either in $Sizes(H_1)$ or $n = d + |C(P)|$ for $d \in Sizes(H_2)$. But now, applying Useful Fact 1, we can chose $l \leq A(m, r) \triangleq m^{(2r+1)m^{2r+1}}$. That means that $ext^H(\alpha \circ C(Y')^l \circ \beta'(Y'))$ contains a configuration of size n for $l \leq A(m, r)$. Thus,

$$TTS(F_\Theta^O, X) \leq 3A(m, r) + A(m, r)g_{O,\Theta}(Y) \cdot n.$$

So we have the result in this case.

In case 2.ii: Since we're in this case,

$$\widetilde{H}(Y') = \widehat{H} \cup \bigcup_{j=1}^{|C(Y')|} \bigcup_{w \in W(v_j)} G'_w$$

is such that $Sizes(\Theta) - Sizes(\widetilde{H}(Y'))$ is finite. (Where G'_w and $W(v_j)$ are as defined above.) Hence, if n is large enough, since n is in $Sizes(\Theta)$, there must a size- n configuration in $\widetilde{H}(Y')$. Thus, there is a size- n configuration either in \widehat{H} or in $D \triangleq \bigcup_{j=1}^{|C(Y')|} \bigcup_{w \in W(v_j)} G'_w$. First, assume $ext^H(Y)$ contains a configuration of size n . Then by 15, $TTS(F_\Theta^O, X) \leq 4 \cdot g_{O,\Theta}(Y)(n)$, and we're done. Now, let's assume D contains a size- n configuration, while \widehat{H} does not. Since there is no size- n configuration in $ext^H(Y)$, by arguments similar to previous case, for some $T \leq 4 \cdot g_{O,\Theta}(Y)(n)$, the configuration will be

$$X_T \triangleq \alpha \circ C(Y')^m \circ C(Y')[1 : K] \circ \triangleright$$

where $m = \lfloor (n - \alpha) / |C(Y')| \rfloor$ and $k = \text{mod}(n - \alpha - 1, |C(Y')|)$. Now, the trajectory of F_Θ^O on X_T is to create all configurations of the form

$$\alpha \circ C(Y')^k \circ C(Y')(1 : j) \circ x$$

for all paths in $G(\Theta)_{\geq w}$ of size $n - |\alpha| - k|C(Y')|$, for all $w \in S(v_j)$. These are precisely the configurations in D . By Useful Fact 1, there will be such a solution with $|x| \leq A(m, r) \leq m^{(2r+1)m^{2r}}$. Hence, $TTS(F_\Theta^O, X_T) \leq 4 \cdot |D \cap C_{\leq A(m, r)}| \leq 4m^{A(m, r)}$. Thus $TTS(F_\Theta^O, X) \leq 4 \cdot g_{O,\Theta}(Y)(n) + 4m^{A(m, r)}$, and we have the result. ■

Claim 3: Step 1: Let X be any configuration of size n . Let j be the minimal position in X such that $X(j) = \triangleleft_i, \triangleright$ or \triangleleft . Suppose $X(j) = \triangleleft_i$ for some i . By arguments identical to those at the beginning of the proof of cor. 1, $TTS(f, X) \leq TTS(f, \widetilde{X}) + n + 3$, where \widetilde{X} is some configuration of the form $Y \circ \triangleright \circ Z$, in which Y is Θ -consistent.

For notational convenience, for $j \leq |Y|$, let $Y_j = Y[1 : j]$ and $b_j = Y[j - 2r - 1 : j]$. Also, abbreviate $s_1 \geq_{O, b_j} s_2$ by $s_1 \geq_j s_2$. Now, let k be the maximal $l \leq |Y|$ such that there is a state s such that $ext^\Theta(Y_l \circ s)$ has a configuration of size n , and such that $s \geq_l Y(l + 1)$, and let s^* be the O -minimal state that satisfies this. Then by definition, for all $j \in [k + 1, |Y| - 1]$, $ext^\Theta(Y_j \circ s)$ does not contain a size n configuration for any s such that $s >_j Y(j + 1)$. Similarly, $ext^\Theta(Y_k \circ s)$ does not contain a size n configuration for any s such that $Y(k + 1) <_k s <_k s^*$. If n is sufficiently large, we therefore know that

$$Sizes(\Theta) - Sizes(ext^\Theta(Y_j \circ s))$$

is infinite for all $s >_j Y(j + 1)$. But then $ext^\Theta(Y'_j \circ s) = H_{O,\Theta}(Y'_j \circ s)$, for all $s >_j Y(j + 1)$, so

$$|ext_n^\Theta(Y_j \circ s)| \leq V \cdot g_{O,\Theta}(P^*)(n - j + |Y'_j|) \leq Vm^{m^{2r+1}} g_{O,\Theta}(P^*)(n - j)$$

, by definition of P^* . Similarly, $|ext_n^\Theta(Y_l \circ s)| \leq Vm^{m^{2r+1}} \cdot g_{O,\Theta}(P^*)(n - l)$ for all s such that $Y(k + 1) <_k s <_k s^*$. On the other hand by arguments identical to those in the proof of 1, by some time T ,

$$T \leq 4|ext_n^\Theta(Y)| + 4 \sum_{j=k+1}^{|Y|-1} \sum_{s >_j Y(j)} |ext_n^\Theta(Y_j \circ s)| + 4 \sum_{Y(k+1) <_k s <_k s^*} |ext_n^\Theta(Y_k \circ s)|,$$

we have

$$\widetilde{X}_T \triangleq (F_\Theta^O)^T(\widetilde{X}) = Y[1 : j] \circ s^* \circ \triangleright \circ Z'.$$

But \widetilde{X}_T is of the form to which the lemma above can be applied. Hence $TTS(F_{\Theta}^O, \widetilde{X}) \leq T + A(m, r)g_{O, \Theta}(Y')(n) < VA(m, r)g_{O, \Theta}(P^*)(n)$. Hence

$$TTS(F_{\Theta}^O, X) \leq 3n + B \sum_l^n g_{O, \Theta}(l)$$

as desired, where $B = 2VA m^{2r+1}$. Since X was arbitrary, this is a bound on the worst-case runtime.

Claim 2: The proof claim 3 actually shows that a configuration X whose left-most error with respect to Θ is at position j is solved by F_{Θ}^O in at most $3n + j \cdot A(m, r)g_{O, \Theta}(|X|)$ timesteps. Let $D_j(n)$ be the set of configurations of size n whose left-most error is at position j . It is easy to see that there that the measure of $D_j(n)$ in \mathbb{C}_n is no greater than $\beta^{j/(2r+1)}(1 - \beta)$, where β is the fraction of r -balls that are not Θ -admissible. Hence

$$\begin{aligned} TTS^{avg}(F_{\Theta}^O)(n) &\leq \sum_{k=1}^n \frac{m^k(m-1)}{m^n-1} \sum_j^k (3n + j \cdot A(m, r)g_{O, \Theta}(n)) \cdot \beta^{j/(2r+1)}(1 - \beta) \\ &= (3n + A(m, r)g_{O, \Theta}(n)) \sum_{k=1}^n \frac{m^k(m-1)}{m^n-1} \sum_j^k j \beta^{j/(2r+1)}(1 - \beta) \\ &\leq (3n + A(m, r)g_{O, \Theta}(n)) \cdot 2(1 - \beta) \beta^{1/2r+1} \frac{\beta}{(\beta - 1)^2} \end{aligned} \quad (\text{B.1})$$

as desired. ■

Proof: (Of prop. 17) Given a path P , denote by P_f the final node of P . Now suppose P is an initial acyclic path, and let $G'_p = P \cup G(\Theta)_{\geq P_f}$. Let \widehat{G}'_p denote the set of configurations corresponding to maximal paths in G'_p , and $\widehat{G}'_{p,n} = |\widehat{G}'_p \cap \mathbb{C}_{\leq n}|$. Then if

$$SIZES(\Theta) - SIZES(G(\Theta)_{\geq P_f})$$

is infinite, $H_{O, \Theta}(P) = G'_p$, and hence, $g_{O, \Theta}(n) \geq V \cdot |\widehat{G}'_{p,n}|$.

Now, by remark 1 embedded in the previous proof, any $H_{O, \Theta}(P)$ always has one of three forms: (1) one of the G'_p , or (2) a line graph (case 2.i) or "line + cycle" graph (case 2.ii) α , together with, for each point in α a union of (at most m^{2r+1}) graphs of the form G'_p . Hence, $H_{O, Th}(P^*)$ in particular has this form. In case (1), $Vg_{O, \Theta}(P^*)(n) \leq g_{O, \Theta}(n)$. In case (2).i, $g_{O, \Theta}(n) \leq m^{2r+1} |\widehat{G}'_{\bar{P}, n}|$ where \bar{P} is the path which maximizes the latter; so $g_{O, \Theta}(n) \leq m^{2r+1}/V \cdot g_{O, \Theta}(n)$. Finally, in case 2.ii,

$$g_{O, \Theta}(n) \leq m^{2r+1} \sum_l^n |\widehat{G}'_{\bar{P}_l}| \leq (m^{2r+1}/V) \sum_l^n g_{O, \Theta}(n).$$

■

Appendix C

Details of Faster Algorithm Proofs

C.1 Proofs for §4.1.2

Proof: (prop. 18). First we state a technical lemma:

Lemma 4 *For any initial configuration X_0 (live) call sequence s , either the trajectory $(\widetilde{F}_\Theta)_s^n(X_0)$ eventually converges to a solved state, or hits a configuration of the form*

$$X^* = Y \circ \triangleright \circ C_\Theta^*(l : |C_\Theta^*|) \circ (C_\Theta^*)^m \circ \mathbf{p}_\Theta \quad (\text{C.1})$$

and for which $\Gamma(B)$ holds, where $B = B_{2r+2}(|Y|, X)$. In the synchronous timing model, this event will occur within $3 + m^{2r+1} + 6|X|$ timesteps.

In words, the algorithm drives all initial conditions to the “special state” in which the “gcd criteria” is present. The proof of this lemma is included below. As a result of the lemma, however, we need only show that any configuration of the form in eq. C.1 will be solved, and, in the synchronous timing model, solved within $2|X| + A$ timesteps, where A is a constant depending only on m and r . To this end, we next demonstrate that for any configuration X of the form in eq. C.1, the trajectory of \widetilde{F}_Θ on X is effectively a *subtrajectory* of the trajectory of F_Θ^O on X , where O is the left-choice function used in the definition of \widetilde{F}_Θ .

Denote the trajectory of \widetilde{F} under some live call sequence s by $\{X, X^1, X^2, \dots\}$, and that of F^O by $\{X, X_a^1, X_a^2, \dots\}$. Now, notice that each point X^i or X_a^i in either trajectory is either solved, or of the form $Y^i \circ d \circ Z^i$ where Y^i is Θ -consistent and $d = \triangleright, \triangleleft, \text{ or } \Delta_j$ for some j . Denote the position of the left-most d in X^i by p_i and that in X_a^i by p'_i . Then the claim is that:

Lemma 5 *There is an infinite string of times $t_1^0, t_1^f, t_2^0, t_2^f, \dots$, such that for all $j \in [t_k^0, t_k^f]$, we have $p_{j-D_{k-1}} = p'_j$ and $X^{j-D_{k-1}}(1 : p_{j+D_k}) = X_a^j(1 : p'_j)$, where $D_k = t_k^0 - \sum_{i=1}^{k-1} (t_i^f - t_i^0)$.*

This lemma is proved given below. We can thus put X^{j-D_k} in 1-1 correspondence with X_a^j , and the former is then effectively an infinite subtrajectory of the latter. Moreover, since F_Θ^O is a solution to X on all live call sequences, \widetilde{F} must also be, and the time-to-solution to F_Θ^O is an upper bound for that of \widetilde{F} , for every call sequence. Since all initial conditions are forced into the form eq. C.1, \widetilde{F} is thus a solution overall.

What remains is to show that the time to solution in the synchronous model for configurations of the form in eq. C.1 is at most $2|X| + B$ for a constant B depending only on m and r . Let $A(m, r) \triangleq m^{2r+1} + m^{(2r+1)m^{2r}}$. Let's distinguish several cases:

I. $A \geq |X| - |Y|$. Define j to be the maximal l for which there is a state $s \succ_{O, Y[l-2r:l]} Y(l+1)$ such that $\text{ext}_n^\Theta(Y[1:l] \circ s)$ contains a configuration of size n . Within this case, there are two subcases:

a. $j \geq A$. In this case, lemma 5 tells us that $\text{TTS}(\widetilde{F}, X) \leq 4 \cdot |\text{ext}_n^\Theta(Y[1:j])| \leq 4m^A$.

b. $j < A$. In this case, let's distinguish two further subcases:

- i. If $\text{ext}^\Theta(Y)$ contains a configuration of size n , then again lemma 5 tells us that $TTS(\widetilde{F}, X) \leq 4m^A$.
- ii. If $\text{ext}^\Theta(Y)$ does not contain a configuration of size n , then, due to lemma 5, within $T \leq 4 \cdot |\text{ext}_n^\Theta(Y[1 : A])| \leq 4m^A$ timesteps, the configuration will have become

$$X^T = Y[1 : A] \circ \triangleleft \circ C_\Theta^*(l : |C_\Theta^*|) \circ (C_\Theta^*)^m \circ \mathbf{p}_\Theta$$

for some l and m . Then, within $2(A - j)$ timesteps, the configuration will become

$$\widetilde{X} \triangleq X^{T+2(A-j)} = Y[1 : j] \circ \triangleright \circ C_\Theta^*(l : |C_\Theta^*|) \circ (C_\Theta^*)^m \circ \mathbf{p}_\Theta$$

for some l and m , and $\Gamma(B_{2r+2}(j, \widetilde{X}))$ holds. Thus, we are in the next case.

- II. $|X| - |Y| \geq A$. Let $Y_f = Y[|Y| - 2r : |Y|]$. Let $o_1 = O^*(Y_f)$. Let $\zeta_1 = Y_f[2 : 2r + 1] \circ o_1$; then let $o_2 = O^*(\zeta_1)$ and $\zeta_2 = \zeta_1[2 : 2r + 1] \circ o_2$; \dots , and so, defining $o_i = O^*(\zeta_{i-1})$ and $\zeta_i = \zeta_{i-1}[2 : 2r + 1] \circ o_i$; until such i^* as $\zeta_{i^*} = \zeta_j$ for some $j < i^*$. The sequence $\zeta = (\zeta_1, \dots, \zeta_{i^*})$ traces out a non-self-intersecting line graph $L = (\zeta_1, \dots, \zeta_j)$ in $G(\Theta)$, attached to a cycle $C = (\zeta_j, \dots, \zeta_{i^*-1}, \zeta_{i^*})$. Let $\mathbf{o}_l = o_1 \circ o_2 \circ \dots \circ o_l$, and $\mathbf{c} = o_{j+1} \circ \dots \circ o_{i^*}$. By definition of what \widetilde{F} does when $\widetilde{\alpha}_1$ and $\widetilde{\gamma}_1$ are activated, the first j steps of the trajectory of X under \widetilde{F} are

$$X^k = Y \circ \mathbf{o}_k \circ \triangleright \circ C_\Theta^*(l_k : |C_\Theta^*|) \circ (C_\Theta^*)^{m_k} \circ \mathbf{p}_\Theta$$

for appropriate l_k and m_k and $k \leq j$. Thereafter, the configuration at timestep k is

$$X^k = Y \circ \mathbf{o}_j \circ \mathbf{c}^{n_k} \circ \mathbf{c}(1 : h_k) \circ \triangleright \circ C_\Theta^*(l_k : |C_\Theta^*|) \circ (C_\Theta^*)^{m_k} \circ \mathbf{p}_\Theta.$$

At each step, $\Gamma(B_{2r+2}(|Y| + k, X^k))$ holds. This goes on until $T = 2(|X| - |Y|)$ the configuration is

$$X^T = Y \circ \mathbf{o}_j \circ \mathbf{c}^{n_T} \circ \mathbf{c}(1 : h_T) \circ \triangleright.$$

From this point, and hereon, the trajectory constructs all configurations of the form

$$X^{T+n_p} = Y \circ \mathbf{o}_j \circ \mathbf{c}^{n_p} \circ p \tag{C.2}$$

where p are the various paths starting at ζ_j in $G(\Theta)$, and $n_p \leq 4m^{|p|}$.

Now, as j is at most m^{2r+1} , so $|\mathbf{c}|n_T + h_T > m^{(2r+1)m^{2r}}$. On the other hand, since $\Gamma(B_{2r+2}(j, X^j))$ holds, $\text{ext}^\Theta(Y \circ \mathbf{o}_j)$ contains a configuration of size n , so that for some acyclic path p from ζ_j to δ_Θ ,

$$n = |p| + |\mathbf{p}_\Theta| + |Y| + j + |\mathbf{c}| \cdot M + \sum_i d_i l_i$$

where the l_i range over the lengths of the cycles in $\text{scc}(p)$. But by Useful Fact 1 from the previous chapter, we can choose the d_i such that $\sum_i d_i l_i \leq m^{(2r+1)m^{2r}}$, with the multiple of $|\mathbf{c}|$ equal to some large enough M^* . Thus, for some path p^* of size $\leq m^{(2r+1)m^{2r}}$, from ζ_j to δ_Θ we have that $\widehat{X} \triangleq Y \circ \mathbf{o}_j \circ \mathbf{c}^{M^*} \circ p^*$ is a configuration of size n . But one of the configurations of the form in eq. C.2 will equal \widehat{X} , and system will halt in a solved stated. Since $n_{p^*} \leq 4m^A$, $TTS(\widetilde{F}, X) \leq 2(|X| - |Y|) + 4m^A$.

■

Proof: (Lemma 4) Throughout we will assume that the synchronous timing model holds, though the proof technique will work for any live timing model. Let $X^i = (\widetilde{F}_\Theta)^i(X)$. Let a configuration X be given. In what follows, let $B_j[X]$ denote the $2r + 2$ ball around agent j in configuration X . If $\Theta(X) = 1$, so that there is no j for which $\Theta[B_r(j, X)]$ fails, then $\widetilde{F}_\Theta(X) = X$. If $\Theta(X) \neq 1$, let j be the minimal l for which $\Theta[X[j - 2r : j]]$ fails to hold. Now, there are a bunch of cases:

- I. If $X(j) = \Delta_i$ for some i , so that $X = Y \circ \Delta_i \circ Z$ for some Θ -consistent Y , then

- a. $\widetilde{\delta}[B_{j-1}(X)]$ holds, then $X^1 = Y[1 : j - 2]s^*(B) \Delta_i Z'$ for some Z' , and $i \neq s^*(B)$. Then we're next case,
- b. $\widetilde{\delta}[B_{j-1}(X)]$ does not hold, then
- i. if $\widetilde{\beta}_3[B]$ then $X^1 = Y[1 : j] \circ \triangleleft \circ Z'$ for some Z' , and we're in case (II) below; while otherwise,
 - ii. $\widetilde{\alpha}_4[B_j(X)]$ must hold, and $X^1 = Y[1 : j] \circ \triangleright \circ Z'$ for some Z' , and we're in case (III) below.
- II. If $X(j) = \triangleleft$, so that $X = Y \circ \triangleleft \circ Z$ for some Θ -consistent Y , then:
- a. $j = 1$, then $\widetilde{\alpha}_3$ applies, and $X^1 = \triangleright \circ Z'$ for some Z' ;
 - b. otherwise, and $\widetilde{beta}_3[B]$ then
 - i. $\widetilde{\beta}_4[B_{j+1}[X]]$ then $X^1 = Y[1 : j - 1] \circ \triangleleft \circ b \circ Z'$ where b is a Θ -consistent r -ball; then either:
 - A. $\widetilde{\beta}_3[B_{j-1}[X]]$ will apply, so that $X^2 = Y[1 : j - 2] \circ \triangleleft \circ \triangleleft \circ b \circ Z'$, and $\widetilde{\beta}_4[B_j[X]]$, so that $X^3 = Y[1 : j - 2] \circ \triangleleft b'$ is a Θ -consistent $r + 1$ ball, and this repeats until either case (a) holds at most $2j$ timesteps later, or until at most $2j$ timesteps later, we're in the next case with $j = j'$ for $j' < j$;
 - B. $\widetilde{\epsilon}[B_{j-1}[X]]$ holds at most $t < 2j$ timesteps later, in which case $X^{t+1} = Y[1 : j' < j] \circ \Delta_{Y(j-1)} \text{circ} Z'$ and we're in case (I)(a), from whence within 2 timesteps, we're in case (I)(b)(ii), so within three timesteps, we're in case (III) below.
 - ii. or $\widetilde{\beta}_4[B_{j+1}[X]]$ does not hold, then $\Theta^-[X[j + 1 : 2r + j + 1]]$ whence $\Gamma^-(X[j - 2r - 1 : -1], B[2 + j : j + 2r + 2])$, so $\widetilde{\beta}_3[B_{j-1}[X]]$ must hold, so that $X^1 = Y[1 : j - 1] \circ \triangleleft \circ \triangleleft \circ Z'$ and $\Gamma^-(X^1[j - 2r - 2 : j - 2], X^1[j + 1 : j + 2r + 1])$, so that this same case (II)(b)(ii) applies to agent $j - 1$; and so on, until case (II)(a) holds within $2j$ timesteps.
 - c. otherwise, and $\widetilde{\beta}_3[B_j[X]]$ does not hold, then if $\widetilde{\beta}_4[B_j[X]]$ holds, then $\widetilde{\beta}_3[B_{j-1}[X]]$ must hold, and we're in case (II)(b)(i) above.
 - d. otherwise, then if $\widetilde{\epsilon}[B_j[X]]$ applies, we're in case (II)(b)(i) above,
 - e. otherwise, then $\widetilde{\alpha}_4[B]$ must apply, so $X^1 = Y[1 : j - 1] \circ \triangleright \circ Z'$, and we're in Case (III).
- III. If $X(j) = \triangleright$, so that $X = Y \circ \triangleright \circ Z$ for some Θ -consistent Y and arbitrary Z , then
- a. either there is no $x \in G(\Theta)$ in a nontrivial cycle such that $X[j - 2r - 1 : j - 1] <_{G(\Theta)} x$; that is, $X[j - 2r - 1 : -1]$ is "after all the cycles". In this case, because $G(\Theta)$ is single terminus, $X[j - 2r - 1 : j - 1] \in p$, where p is the unique path in $G(\Theta)$ from γ_Θ to δ_Θ , in which case either
 - i. $|X| - j = |p| - A$, where A is the position of $X[j - 2r : j - 1]$ in p , in which case F_Θ^O solves the configuration within $|p| - A + 1 \leq m^{2r}$ timesteps (the m^{2r} comes from the maximal length of a acyclic path).
 - ii. or $|X| - j < |p| - A$, so within at most m^{2r} steps, $\widetilde{\beta}_1$ applies, and the configuration is in case (II)(b)(ii), so within $|X|$ steps we're in case (b) below, with $j = 1$
 - iii. or $|X| - j > |p| - A$, so again within at most m^{2r} steps, $\widetilde{\beta}_2$ applies, and again the configuration is in case (II)(b)(ii), so within $|X|$ steps, we're also in case (b) below with $j = 1$.
 - b. or $X[j - 2r - 1 : -1]$ is "before a cycle" in $G(\Theta)$, that is, there is $x \in G(\Theta)$ in a non-trivial cycle, and a path from $X[j - 2r - 1 : j - 1]$ too x . In this case, C_j be the cycle that is "next" relative to O . Then with $|X|$ timesteps, applications of $\widetilde{\alpha}_2$ and $\widetilde{\gamma}_1$ alternate until the configuration has become $Y' \circ C_j^m \circ C_j(1 : l) \circ \triangleright$ for some Y' and appropriate m and l ; then $\widetilde{\beta}_2$ applies, so that the configuration becomes $Y' \circ C_j^m \circ C_j(1 : l) \circ \triangleright$; then applications of $\widetilde{\beta}_3$ and $\widetilde{\beta}_4$ alternate, producing configurations of the form:

$$\widetilde{X}^t = \widetilde{Y}^t \circ \triangleleft \circ C_\Theta^*(l : |C_\Theta^*|) \circ (C_\Theta^*)^k \circ \mathbf{p}_\Theta$$
 with $(k + 1)|C_\Theta^*| - l$ increasing by 1 every two timesteps, until within $T < |X|$ timesteps, either
 - i. Case (II)(a) arises, so within one timestep, the configuration is $\triangleright \circ C_\Theta^*(l : |C_\Theta^*|) \circ (C_\Theta^*)^k \circ \mathbf{p}_\Theta$, which has to fit the claim given $n \in \text{Sizes}(\Theta)$,

ii. or for the configuration \widetilde{X}^T satisfies

- $(\star(B) > 1)$ and $(\star(B) < |B|)$, and
- $(L_O^P)^{\neg}[B]$, and
- $\Theta_{-r-2}[B] \wedge \Theta_{-r-1}[B] \wedge (B(0) = \triangleleft)$

where $j = |\widetilde{Y}^T| + 1$, $B = B_j[\widetilde{X}^T]$, whence $\widehat{\epsilon}[B]$ and $\widehat{\delta}[B_{j-1}[\widetilde{X}^T]]$. But then

$$\widetilde{X}^{T+1} = \widetilde{Y}^T[1 : j-1] \circ \Delta_{B(-1)} \circ C_{\Theta}^*(l : |C_{\Theta}^*|) \circ (C_{\Theta}^*)^k \circ \mathbf{p}_{\Theta},$$

so

$$\widetilde{X}^{T+2} = \widetilde{Y}^T[1 : j-2] \circ s^*(B) \circ \Delta_{B(-1)} \circ C_{\Theta}^*(l : |C_{\Theta}^*|) \circ (C_{\Theta}^*)^k \circ \mathbf{p}_{\Theta}$$

so

$$\widehat{X} \triangleq \widetilde{X}^{T+3} = \widetilde{Y}^T[1 : j-2] \circ s^*(B) \circ \triangleright \circ C_{\Theta}^*(l : |C_{\Theta}^*|) \circ (C_{\Theta}^*)^k \circ \mathbf{p}_{\Theta}$$

and $\Gamma(b_1, b_2)$ where $b_1 = \widehat{X}(j-2r-1 : j-1)$ and $b_2 = \widehat{X}(j+1 : j+2r+1)$, which is the desired form.

This proof shows that all trajectories eventually wind up in case (III); adding up the time steps that each step might take, as indicated, shows that all cases wind up in case (III) within $5|X|$ timesteps. Analysis of Case (III) shows that either the trajectory winds up in (III)(a)(i) and ends before an additional m^{2r} timesteps, or within additional $|X| + 2$ timesteps, ends up in the trajectory winds up in (III)(b)(ii). Thus the desired form is achieved with $6|X| + m^{2r} + 2$ timesteps. ■

Proof: (Lemma 5) Let any configuration of the form $C_{\Theta}^*(l : |C_{\Theta}^*|) \circ (C_{\Theta}^*)^m \circ \mathbf{p}_{\Theta}$ be called ‘‘special.’’ Now suppose

$$X = Y \circ \triangleright \circ Z$$

for some special configuration Z , as in eq. C.1. Let $Y_f = Y[|Y| - 2r : |Y|]$. Recall the definition of $O^*(B)$ as:

- the O -minimal state s for which there is path from $B[-2r : -1] \circ s$ to δ_{Θ} such that

$$|p| - |\mathbf{p}_{\Theta}| - \eta(B[2 : 2r+2], C_{\Theta}^*) + r \text{ is divisible by } \gcd(p)$$

when $\Gamma(B[-2r-1 : -1], B[2 : 2r+2])$ holds.

- $O(B[-2r-1 : -1], 1)$ otherwise.

Let $t_1^0 = 0$. Let $o_1 = O^*(Y_f)$. Let $\zeta_1 = Y_f[2 : 2r+1] \circ o_1$; then let $o_2 = O^*(\zeta_1)$ and $\zeta_2 = \zeta_1[2 : 2r+1] \circ o_2; \dots, o_i = O^*(\zeta_{i-1})$ and $\zeta_i = \zeta_{i-1}[2 : 2r+1] \circ o_i$ and so on. Let $\mathbf{o}_l = o_1 \circ o_2 \dots \circ o_l$. Now, by definition of what \widetilde{F} does when $\widetilde{\alpha}_1$ and $\widetilde{\gamma}_1$ are activated, the trajectory of X under \widetilde{F} consists of configurations of the form

$$X^l = Y \circ \mathbf{o}_l \circ \triangleright \circ Z[l+1 : |Z|]$$

for $l \leq j$, until at $T = 2(|X| - |Y|)$ the configuration is

$$X^T = Y \circ \mathbf{o}_{|X|-|Y|-1} \circ \triangleright.$$

Now lets see what happens under F . Let $I_j = \{s_1, \dots, s_k\}$ denote the set of states s such that

$$O(\zeta_j, 1) \leq_{O, \zeta_j} s <_{O, \zeta_j} O^*(\zeta_j).$$

If $O(\zeta_j, 1) = O^*(\zeta_j)$ then I_j is empty. Then: (1) let t_1^f be the first timestep j for which I_j is nonempty. The trajectory of X under F is identical with that of \widetilde{F} for $t \in [t_1^0 = 0, t_1^f]$. (2) In first step after t_1^f the trajectory of F on X will come to

$$Y \circ \mathbf{o}_{t_1^f} \circ s_1 \circ \triangleright \circ Z[2 : |Z|].$$

Then, since

$$\Gamma(\zeta_{t_1^f}[2 : 2r] \circ s_1, Z[t_1^f + 2 : t_1^f + 2r + 2])$$

does NOT hold (by assumption), $\text{ext}(\zeta_{t_1^f}[2 : 2r] \circ s_1)$ cannot contain a configuration of size $|X|$. Hence the head eventually returns, and configuration becomes

$$Y \circ \mathbf{o}_{t_1^f} \circ s_1 \circ \triangleleft^{|X|-|Y|-t_1^f-1}.$$

Within two timesteps it becomes

$$Y \circ \mathbf{o}_{t_1^f} \circ s_1 \circ \triangleright \circ \triangleleft^{|X|-|Y|-t_1^f-2}.$$

Unless $s_2 = O^*(\zeta_{t_1^f})$,

$$\Gamma(\zeta_{t_1^f}[2 : 2r] \circ s_2, Z[2 : 2r + 2])$$

does not hold. Again, $\text{ext}(\zeta_{t_1^f}[2 : 2r] \circ s_2)$ cannot have a configuration of size $|X|$. Thus, the head eventually returns, and configuration now becomes

$$Y \circ \mathbf{o}_{t_1^f} \circ s_2 \circ \triangleleft^{|X|-|Y|-t_1^f-1}.$$

Within two timesteps, it becomes

$$Y \circ \mathbf{o}_{t_1^f} \circ s_3 \circ \triangleright \circ \triangleleft^{|X|-|Y|-t_1^f-2}.$$

Unless $s_3 = O^*(Y_f)$, the cycle repeats again . . . (3) Hence, eventually, we'll reach the state $O^*(Y_f)$, and the trajectory will be

$$Y \circ \mathbf{o}_{t_1^f} \circ \mathbf{o}_{t_1^f+1} \circ \triangleright \circ \triangleleft^{|X|-|Y|-t_1^f-2}.$$

Let t_2^0 denote the time at which this occurs. (4) Then repeat the calculation in steps (1)-(3). Let t_2^f denote the first timestep $j > t_2^0$ for which I_j is non-empty. The trajectory of X under \tilde{F} from $t = 2$ to $t = t_2^f - t_2^0$ is identical (up to the position of the first turing head) with the trajectory of X under F , from $t = t_2^0$ to $t = t_2^f$. Again, the Γ predicate repeatedly will not hold, and thus the extensions are empty. Eventually the trajectory becomes

$$Y \circ \mathbf{o}_{t_1^f+2} \circ \triangleright \circ \triangleleft^{|X|-|Y|-t_1^f-3}.$$

Repeating steps (1)-(4) until the head \triangleright hits the right end, we see that the trajectory becomes

$$Y \circ \mathbf{o}_{|X|-|Y|-1} \circ \triangleright$$

after

$$|X| - |Y| - 1 + \sum t_{l+1}^0 - t_l^f$$

steps.

This calculation establishes the result for trajectories during which the turing head is moving right. An analogous computation works for when it is moving left. This computation is also done assuming the synchronous timing model, but works completely for any live call sequence by replacing the phrase "the next timestep" with "the next timestep when agent $[x]$ is called" for whatever the relevant agent(s) x are. ■

C.2 Details of the General Algorithm

In the following, I describe the details of the construction of the local rule for a general graph $G(\Theta)$. Make the following preliminary definitions:

- Define the boolean

$$L_R(B) = \Theta_{r,r+1}[B] \Rightarrow (B(0) = P(B[1 : 2r + 1], |L(B[1 : 2r + 1])|)).$$

In words, $B(0)$ "maxes out" relative to P the choices for states consistent to the left $B[1 : 2r + 1]$.

- Define the boolean

$$T_R(b) = 1 \text{ if } b \in G(\Theta) \text{ and } scc(p) = 0 \text{ for all paths } p \text{ with } p_0 = b.$$

In words, there is no path from b to a cycle in $G(\Theta)$, so b is a “right-terminal” node. Let $A_0 = \{b | T_R(b)\}$.

- Let $Solv(B)$ be the boolean that is 1 if there is a state j such that $\bigwedge_{i=0}^{2r} B[-2r-i : -1] \circ j \circ B[1 : i]$ and $\Theta_{-r-1}(B)$ and $\Theta_{r+1}(B)$. In words, $Solv(B)$ holds when a single change to $B(0)$ will produce a locally solved ball. Let

$$Solv(B) = \min\{j | \bigwedge_{i=0}^{2r} B[-2r-i : -1] \circ j \circ B[1 : i]\}.$$

That is, $Solv(B)$ is the minimal that will achieve the local solution, when $Solv(B)$ holds.

- Let $A_1 = \cup\{in(b, G(\Theta)) | b \in A_0\} - A_0$. These are the “almost-terminal” nodes.
- Let the elements of A_1 be numbered 1 through $|A_1|$. For $b \in A_1$, write $m(b)$ for the associated number. For each $j \in \{1, \dots, |A_1|\}$, let $b(j) \in A_1$ be such that $m(b(j)) = j$. These numbers allow us to distinguish the various “gateway” nodes in the single-terminus components.
- For each $j \in \{0, \dots, |A_1|\}$, associate states \triangleleft_j and, for $k \in \{1, \dots, m\}$, Δ_j^k . These include the states \triangleleft_0 and Δ_0^k , which are analogous to the original \triangleleft and \triangleleft states. These are the states which allow the agent to determine which single-terminus graph it is computing the gcd criterion relative to.
- For each $b \in A_1$, make the inductive definition

$$\gamma_i \triangleq \mathbf{P}(\gamma_{i-1}, 1), \text{ and } A_b^i = (\gamma_i, \gamma_{i-1}) \circ A_b^{i-1}$$

in which $\gamma_0 = b$. Repeat this until such i^* as $\gamma_{i^*} = \gamma_i$ for some $i < i^*$. Let $A_b = A_b^{i^*}$. Each A_b is of the form of a “cycle+path” graph; let C_b denote the cyclic portion and L_b denote the linear portion. These are the “most-terminal” cycles and lines within the single-terminus graphs associated with the node b .

- For each $b \in A_1$, A_b consists of nodes γ_i . Introduce the numbering functions n_j for each $j = 1, \dots, |A_1|$, defined so that $n_{m(b)}(\gamma_i) = i$. (Thus, for example, $n_{m(b)}(b) = 0$). These numberings are the distances of the nodes the terminal cycles C_b relative to the gateway node b .
- Let $A_2 = \cup_{b \in A_1} A_b$, and $A = A_0 \cup A_1 \cup A_2$. Let $C = \cup_{b \in A_1} C_b$.

Now, we are ready to define the gcd criterion implementation, generalizing the boolean Γ defined in §4.1.2. Let B be a ball of radius $2r + 2$. Let $b_1 = B(-2r - 1 : -1)$ and $b_2 = B(2 : 2r + 2)$, considered as r -balls. Let the boolean function $\Gamma(B)$ be defined as TRUE when:

1. $b_1 \in G(\Theta)$ and $b_2 \in A$.
2. if $b_2 \in A_0$, there is a path from b_1 to b_2 of length $\star(b_1) + \star(b_2)$.
3. if $b_2 \in A - A_0 - C$, then for some $1 \leq j \leq |A_1|$, $B(0) = \triangleleft_j$, or Δ_j^k , such that $n_j(b_2)$ is defined; and there is an (acyclic) path from b_1 to $b(j)$ of length $n_j(b_2) + \star(b_1) + \star(b_2)$;
4. if $b_2 \in C$, then for some $1 \leq j \leq |A_1|$, $B(0) = \triangleleft_j$, or Δ_j^k , such that $n_j(b_2)$ is defined; and there is an acyclic path p from b_2 to $b(j)$ such that

$$|p| - n_j(b_2) - \star(b_1) - \star(b_2) \text{ is divisible by } gcd(p).$$

Let $\tilde{L}_O^P[B]$ be the boolean defined as

$$\tilde{L}_O^P[B] = \Gamma(B) \Rightarrow (\exists s >_{O, b_1} B(0) | \Gamma(B[-2r-2 : -1] \circ s \circ B[1 : 2r+2])).$$

Let $s^*(B)$ be the O -minimal $s >_{O, b_1} B(0)$ for which $\Gamma(B[-2r-2 : -1] \circ s \circ B[1 : 2r+2])$, which is defined when $\Gamma(B) \wedge (\tilde{L}_O^P[B])^\neg$.

Now, define the algorithm $F[B]$, with radius $2r + 3$, by:

- **Rule 1:** If

$$\omega_1[B] = \Theta_{-r-1}[B] \wedge \Theta_{r+1}[B] \wedge (B(0) \notin S) \wedge \text{Sol}(B)$$

then let $F[B] = \text{Sol}(B)$.

- **Rule 2:** Else if

$$\omega_2[B] = \Theta_{-r-1}[B] \wedge \Theta_{-r}^-[B] \wedge (B(0) \in S)$$

then let $F(B) = \triangleright_0$.

- **Rule 3:** Else if

$$\omega_3[B] = (\star(B) > 1) \wedge (B(-1) = \triangleright_0) \wedge \Theta_{-r-2}[B] \wedge ((B[1 : 2r + 1] \in A_0) \Rightarrow L_R(B))$$

then $F(B) = \triangleright_0$.

- **Rule 4:** Else if

$$\omega_4[B] = (\star(B) > 1) \wedge (B(-1) = \triangleright_1) \wedge \Theta_{-r-2}[B] \wedge ((B[0 : 2r] \in A_1) \wedge \Theta_{r+1}(B) \Rightarrow (B[1 : 2r + 1] \notin A_0))$$

then let $F(B) = \triangleright_1$.

- **Rule 5:** Else if

$$\omega_5[B] = (\star(B) > 1) \wedge (B(0) = \triangleright_1) \wedge (B[1 : 2r + 1] \in A_1) \wedge (B[2 : 2r + 2] \in A_0)$$

then let $F(B) = \triangleleft_{m(B[1:2r+1])}$.

- **Rule 6:** Else if

$$\omega_6[B] = (\star(B) < |B|) \wedge \Theta_{-r-1}[x] \wedge (B(0), B(1) \in \{\triangleright_0, \triangleright_1\}) \wedge (\exists j | \Theta[B[-2r : -1] \circ j])$$

then let $F[B] = O(B[-2r - 1 : -1], 1)$.

- **Rule 7:** Else if

$$\omega_7[B] = (\star(B) < B) \wedge \Theta_{-r-1}[X] \wedge ((B(0) = \triangleright_0) \vee (B(0) = \Delta_{B(1)}^R)) \wedge T_R(B[2 : 2r+2]) \wedge (B(1) \neq P(B[2 : 2r+2], |L(B[2 : 2r+2])|))$$

then let $F[B] = \Delta_{B(1)}^R$.

- **Rule 8:** Else if

$$\omega_8[B] = (\star(B) > 1) \wedge \Theta_{-r-2}[X] \wedge (B(-1) = \Delta_{B(0)}^R) \wedge T_R(B[1 : 2r+1]) \wedge (B(0) \neq P(B[1 : 2r+1], |L(B[1 : 2r+1])|))$$

, then $F[B] = P^{+1}(B)$.

- **Rule 9:** Else if

$$\omega_9[B] = (\star(B) < B) \wedge \Theta_{-r-1}[X] \wedge \Theta_{r+1}[B] \wedge (\exists j B(0) = \Delta_j^R) \wedge (B(0) \neq \Delta_{B(1)}^R)$$

let $F[B] = \triangleleft_0$.

- **Rule 10:** Else if

$$\omega_{10}[B] = (\star(B) < |B|) \wedge \Theta_{-r-1}[B] \wedge (B(0) = \triangleright_0) \wedge (\bigwedge \Theta^-[B[-2r : -1] \circ j]),$$

OR

$$(\star(B) = |B|) \wedge (B(0) = \triangleright_0) \wedge \Theta_{-r-1}[B] \wedge (\bigwedge_{j=0}^{m-1} b_j^-[B]),$$

where $\eta[B] = (\star(B) = |B|) \wedge \bigvee_{l=-r+1}^0 \Theta^-[B[l - r : 0]]$, then let $F[B] = \triangleleft_0$.

- **Rule 11:** Else if

$$\omega_{11}[B] = (\star(B) > 1) \wedge \Theta_{r+1}[B] \wedge (\exists j, j' (B(-1) = \triangleleft_j) \wedge (B(0) = \triangleleft_{j'})) \wedge (\exists j [\Theta[j \circ B[1 : 2r + 1]]]),$$

then let $F[B] = P(B[1 : 2r + 1], 1)$.

- **Rule 12:** Else if

$$\omega_{12}[B] = (B(1) = \triangleleft_j) \wedge \widetilde{L}_O^P[B] \wedge ((j = 0) \wedge (B[2 : 2r + 2] \in A_1) \Rightarrow (B[3 : 2r + 3] \notin A_0)) \wedge ((B(0) = \triangleleft_j) \vee \omega_{11}[B^{+2}]^-)$$

let $F[B] = \triangleleft_j$.

- **Rule 13:** Else if

$$\omega_{13}[B] = (\star(B) < B) \wedge (B(1) = \triangleleft_0) \wedge \widetilde{L}_O^P[B] \wedge (B[2 : 2r + 2] \in A_1),$$

let $F[B] = \triangleleft_{m(B[2:2r+1])}$.

- **Rule 14:** Else if

$$\omega_{14}[B] = (\star(B) > 1) \wedge ((B(0) = \triangleleft_j) \vee (B(0) = \Delta_{B(-1)}^j)) \wedge \Gamma(B^{-1}) \wedge (\widetilde{L}_O^P)^-[B^{-1}],$$

let $F[B] = \Delta_{B(-1)}^j$.

- **Rule 15:** Else if

$$\omega_{15}[B] = (\star(B) < |B|) \wedge (B(1) = \Delta_{B(0)}^j) \wedge \Gamma(B) \wedge (\widetilde{L}_O^P)^-(B),$$

let $\widetilde{F}[B] = s^*(B)$.

- **Rule 16:** Else if

$$\omega_{16}[B] = (\star(B) > 1) \wedge \Theta_{-r-2}[B] \wedge \Theta_{-r-1}[B] \wedge (B(0) = \Delta_k^j) \wedge (B(-1) = P(B[-2r - 2 : -2], P_{B[-2r-2:-2]}^{-1}(k + 1))),$$

let $F[B] = \triangleright_1$.

- **Rule 17:** Else if

$$\omega_{17}[B] = B(0) \notin S$$

then let $F[B] = \triangleright_0$.

C.3 Proof of Prop. 22

Recall the definition:

$$\widehat{F}(B) = \begin{cases} \widehat{F}_1(B), & \text{if } B \in \mathcal{B}_W \\ \widehat{F}_2(B), & \text{otherwise} \end{cases}$$

where \widehat{F}_1 and \widehat{F}_2 are defined as in §§4.2.1-4.2.2.

To compute the runtime of \widehat{F} , first we compute $TTS(\widehat{F}_2)$, in the three timing models. Let's tackle **the k -bounded asynchronous model**, \mathcal{S}_k . Let X be an initial condition and fix a call sequence $s \in \mathcal{S}_k(|X|)$. Let X^t denote $(\widehat{F}_2)_s^t(X)$, for any call sequence $s \in \mathcal{S}_k$. Let $B_j^t = B_r(j, X^t)$. Now, let's note several things:

1. By definition if $R(B_j) \geq 2L + |C|$, then the configuration is "solved" locally to agent j , and agent j 's state will never change under subsequence application of \widehat{F}_2 .
2. Now, if $R(B_j) \geq 2L + |C|$, then for some $t \leq k|X|$, $R(B_{j+1}^t) \geq 2L + |C|$. This is because: if we let t be the first agent $j + 1$ is called, either (a) $R(B_{j+1}^{t-1}) \geq 2L + |C|$ already, whence in light of point 1 above, $R(B_{j+1}^t) \geq 2L + |C|$, or (b) $R(B_{j+1}^{t-1}) < 2L + |C|$, in which case since $L(B_{j+1}^{t-1}) \geq R(B_j^{t-1}) \geq 2L + |C|$, the ω_3 predicate is activated, so once j is called, $R(B_j^t) \geq 2L + |C|$. Finally, since $s \in \mathcal{S}_k$, the first call to any given agent in a configuration of size $|X|$ is made within $k|X|$ timesteps, so we can assume $t \leq k|X|$.

3. Hence, repeating the reasoning in point 2 above another l times, if $R(B_j) \geq 2L + |C|$, then for some $t \leq k \cdot l \cdot |X|$, $R(B_{j+o}^t) \geq 2L + |C|$, for all $0 \leq o \leq l$.
4. Given an agent j , there is a $t_1 \leq k|X|$ there is h such that $j - |C| \leq h \leq j$ such that $R(B_h^{t_1}) > 0$. This is because, if $R(B_h) = 0$ for all $h \in [j - |C|, j]$, when the first agent in $h \in [j - |C|, j]$ is called (which must happen within $k|X|$ timesteps, the predicate ω_4 applies, and $R(B_h)$ becomes greater than 0.
5. Given an agent j in X , let a be the maximal $l \leq j$ such that $R(B_l^0) = R(B_j^0)$; and let b be the minimal $l \geq j$ such that $R(B_l^0) = R(B_j^0)$. Of course, $j - a, b - j \leq R(B_j^0)$. Suppose $R(B_j^0) < 2L + |C|$. Then let t_1 be the first time t at which an agent $c \in [a, b + 1]$ is called for which $\widehat{F}_2(c, X^t) \neq X(c)$. Such a t_1 must exist if $R(B_j^0) < 2L + |C|$, and $t_1 \leq k|X|$. Furthermore the call must be to either agent $b + 1$ or a . There are two cases: (I) If $b + 1$ is the first effective call, then $R(B_{b+1}^{t_1-1}) < R(B_b^{t_1-1})$, so $R(B_{b+1}^{t_1}) = R(B_b^{t_1}) + 1$. If (II) a is the first effective call, $R(B_a^{t_1-1}) \leq R(B_{a-1}^{t_1-1})$, so $R(B_a^{t_1}) \geq R(B_{a-1}^{t_1}) + 1$. Both cases have the commonality that, as long as $R(B_j^0) < 2L + |C|$, then there is some $t \leq k|X|$ and $j_1 \geq j - R(B_j^0)$ such that $R(B_{j_1}^t) \geq R(B_j^0) + 1$.
6. Item 5 is the inductive hypothesis of, and item 4 the base case of, the proof of the claim that: for some and j in X , there is $t_l \leq k \cdot l \cdot |X|$ and $j_l \geq j - \sum_{o \leq l} l$ such that

$$R(B_{j_l}^{t_l}) \geq \min(2L + |C|, R(B_j^0) + l).$$

Now, this implies that for some $T_1 \leq k \cdot (2L + |C|) \cdot |X|$, there is $j_l \geq j - D$ such that $R(B_{j_l}^{T_1}) \geq 2L + |C|$, where

$$D \triangleq \sum_{i=1}^{2L+|C|} i = \frac{1}{2}(2L + |C|)(2L + |C| + 1).$$

7. But now applying item 2 implies that for some $T_2 \leq k \cdot D \cdot |X|$, $R(B_{j_l}^{T_1+T_2}) \geq 2L + |C|$. But that means that the configuration is completely solved by $T_1 + T_2$, which is bounded by $k \cdot |X| \cdot (1 + 2L + |C| + D) < k(2L + |C| + 1)^2|X|$. Hence $TTS(\widehat{F}_2, X, s) \leq k(2L + |C| + 1)^2$ for all $s \in \mathcal{S}_{k,r}$ so

$$TTS_{\mathcal{S}_k}^{\text{worst}}(\widehat{F}_2)(n) \leq k(2L + |C| + 1)^2,$$

a constant independent of system size.

Because the expected round time of **the totally asynchronous model** is less than $2 \log(|X|)$, the same reasoning as in the k -bounded case applies here, replacing $k|X|$ whenever it is mentioned with $2 \cdot X \cdot \log(X)$. Hence averaging over s_r , we have that $TTS_{\mathcal{S}_r}^{\text{worst}}(\widehat{F}_2)(n) \leq 2(2L + |C| + 1)^2 \cdot \log(n)$. On other hand, almost all initial conditions will require at least one action at an agent within every r -ball. Hence, $TTS^{\text{avg}}(\widehat{F}_2)(n) \geq (1/(2r + 1)) \cdot \log(n)$. Summarizing,

$$\frac{1}{2r + 1} \log(n) \leq TTS_{\mathcal{S}_r}^{\text{avg}}(\widehat{F})(n) \leq TTS_{\mathcal{S}_r}^{\text{worst}}(\widehat{F}_2)(n) \leq 2(2L + |C| + 1)^2 \cdot \log(n).$$

Now, let's turn our attention to **the totally synchronous model**. Here the situation is a little bit more subtle. Let X be an initial condition, and lets again use the notation $X^t = (\widehat{F}_2)^t(X)$ and $B_j^t = B_r(j, X^t)$.

1. Just as in the previous cases, if $R(B_j) \geq 2L + |C|$, then the first time $j + 1$ is called $t \leq k|X|$, $R(B_{j+1}^t) \geq 2L + |C|$. But again $j + 1$ is called at every timestep in the synchronous model, so if $R(B_j) \geq 2L + |C|$, $R(B_{j+o}^t) \geq 2L + |C|$, for all $0 \leq o \leq l$.
2. Given agent j in X , let a_j^t be the maximal $l \leq j$ such that $R(B_l^t) = R(B_j^t)$; and let b_j be the minimal $l \geq j$ such that $R(B_l^t) = R(B_j^t)$. Evidently $j - a_j^t, b_j^t - j \leq R(B_j^t)$. Now, suppose that j is such that

$$R(B^0(a_j - l)), R(B_{b_j+l}^0) < R(B_j^0) \text{ for all } l \leq 2L + |C|.$$

Then, as long as $R(B_{j_l}^0) < 2L + |C|$, $B_j^l = B_j^{l-1} + 1$, $a_j^l = a_j^{l-1}$, and $b_j^l = b_j^{l-1} + 1$ for all $l \leq 2L + |C|$. Hence $R(B_j^{2L+|C|}) \geq 2L + |C|$.

3. For all agents k in X , let j_k be the maximal $l \leq k$ for which $R(B^0(a_{j_k} - l))$ and $R(B_{b_{j_k}+l}^0) < R(B_j^0)$ for all $l \leq 2L + |C|$. Then the combining points 1 and 2 above shows that $R(B_k^T) \geq 2L + |C|$ for some $T \leq 2L + |C| + k - j_k$. Hence,

$$TTS(\widehat{F}_2, X) \leq 2L + |C| + E(X) \quad (\text{C.3})$$

where $E(X) \triangleq \max_{k \in X} (j - j_k)$.

4. Now, a moment's thought will show that $E(X)$ is at most the length of the maximal contiguous substring of X with periodicity no greater than $2L + |C|$. That is, define

$$per_l(X) = \{Y \subseteq X \mid Y(1 : |Y| - l) = Y(l : |Y|)\}$$

and

$$D_k(X) = \max \left\{ |Y|, Y \in \bigcup_{l=1}^k per_l(X) \right\}.$$

Then,

$$E(X) \leq D_{2L+|C|}(X) \quad (\text{C.4})$$

5. Conversely, for all configurations X and local rules f of radius r the drive X to a configuration in \mathcal{Z} , as long as $|C| > 1$,

$$TTS_{S^s}(f, X) \geq \frac{1}{2(2r+1)} D_{|C|-1}(X) \quad (\text{C.5})$$

. To see why this holds, suppose $X = Y \circ W \circ Z$, where W is periodic with period $l < |C|$. Then, for all $j \in [|Y| + 1, |Y| + |W|]$, we have $R(B_j^0) \leq 1$. Moreover, since $W(1 : |W| - l) = W(l : |W|)$, under the synchronous application of *any* rule of radius r , $W(2r+1 : |W| - l - 2r - 1) = W(2r+l+1 : |W| - 2r - 1)$. But then for all $j \in [|Y| + 2r + 1, |Y| + |W| - 2r - 1]$, $R(B_j^1) \leq 1$. Repeating this another $|Y|/2(2r+1)$ times, we find that $R(B_{|X|+|Y|/2}^l) \leq 1$ for all $l \leq |Y|/2(2r+1)$, which yields the result.

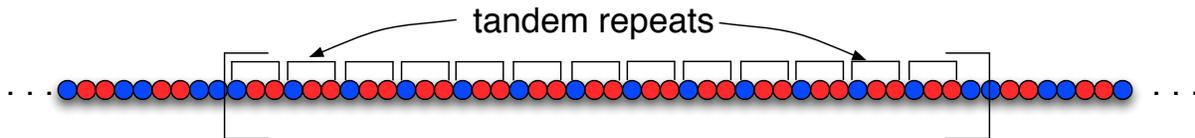
6. Combining eqs. C.3, C.4, and C.5, we find

$$\frac{1}{2(2r+1)} \langle D_{|C|-1}(X) \rangle_{X \in \mathcal{C}_n} \leq TTS_{S^s}^{avg}(\widehat{F}_2)(n) \leq 2L + |C| + \langle D_{2L+|C|}(X) \rangle_{X \in \mathcal{C}_n}$$

and

$$\frac{1}{2(2r+1)} \max_{X \in \mathcal{C}_n} D_{|C|-1}(X) \leq TTS^{worst}(\widehat{F}_2)(n) \leq 2L + |C| + \max_{X \in \mathcal{C}_n} D_{2L+|C|}(X).$$

7. Now, $\max_{X \in \mathcal{C}_n} D_{2L+|C|}(X) = n$. Hence, the worst-case scaling of \widehat{F}_2 will be linear in system size in the synchronous model. The intuitive point here is that “small tandem repeats” are in general a problem for synchronous local rules. By “tandem repeats”, I mean subconfigurations composed of many repeats of small, equally-sized subunits that are *incorrect* relative to C_1 :



Any order that a local rule wishes to establish that does not respect the shift symmetry of the tandem repeats has to wait for the symmetry to be broken from somewhere else in the system (i.e. the ends). The problems that synchronous algorithms have with tandem repeats are a specific instance of a much more general phenomenon in which “low order symmetries” – and small tandem repeats are

precisely subconfigurations that have small shift symmetry – cause difficulties in synchronous update environments. This principle is behind most of the impossibility results in distributed computing.

The worst-case behavior here is, unlike in previous local rules and situations, not representative of the average behavior. It is a standard result of analytic combinatorics that, considered as a random variable $D_i(X)$ with $X \in \mathbb{C}_n$ over m states, has mean no greater than $\log_m(n)$ ([22]). Thus the average runtime scales logarithmically.

Combining all this information, we have that \widehat{F} scales

- as $O(1)$, in the worst-case, for the k -bounded asynchronous timing model or the synchronous timing model if $|C| = 1$ for some cycle $C \in G(\Theta)$.
- as $O(\log(n))$ in the worst and average-case, for the totally asynchronous timing model
- as $O(\log(n))$ in the average and $O(n)$ in the worst case for the synchronous timing model if $|C_i| > 1$ for all cycles C_i in $G(\Theta)$.

This establishes prop. 22.

Appendix D

Details of \widetilde{P}

D.1 \widetilde{P} is More Robust

Proof: (Of Proposition 28.) First, for any pattern T and $M \leq N$, let $f_M^N(T) = |T_M|/|T_{\leq N}|$, i.e. the number of elements of size M relative to the number of all elements of size N or less. Notice that if T is infinite then $\lim_{N \rightarrow \infty} \sum_{M=1}^N f_M^N(T) \frac{1}{M} = 0$.

Assume wlog that z is empty. Now, let

$$C(N) = \frac{q \lfloor N/q \rfloor (1 + \lfloor N/q \rfloor)}{2}$$

be the normalizing function for \widetilde{P} . Then suppose X is a configuration with $|X| = n$ and consider $X \circ y$ for some finite configuration y with $|y| = d$. We have:

$$\begin{aligned} \widetilde{\mathbb{P}}[X \circ y](p/q) &= \frac{1}{C(N+d)} \sum_{m=1}^{\lfloor (N+d)/q \rfloor} \sum_{s=0}^{N+d-mq-1} \mathbb{P}[X \circ y(s : s + mq - 1)](pm) \\ &= \frac{1}{C(N+d)} \sum_{m=1}^{\lfloor (N+d)/q \rfloor} (X_m + C_m) \end{aligned} \tag{D.1}$$

where

$$X_m = \sum_{s=0}^{N-mq-1} \mathbb{P}[X \circ y(s : s + mq - 1)](pm)$$

and

$$C_m = \sum_{s=N-mq}^{N+d-mq-1} \mathbb{P}[X \circ y(s : s + mq - 1)](pm).$$

Note $|X_m| \leq \min(0, (N - mq + 1))K$ where K is the the largest possible value of the norm-square of a fourier coefficient of a signal; and note that $|C_m| \leq dK$. Now

$$\widetilde{\mathbb{P}}[X](p/q) = \frac{1}{C(N)} \sum_m^{\lfloor N/q \rfloor} X_m.$$

Hence

$$\left| \widetilde{\mathbb{P}}[X \circ y](p/q) - \widetilde{\mathbb{P}}[X](p/q) \right| \leq \left| \left(\frac{1}{C(N+d)} - \frac{1}{C(N)} \right) \sum_{m=1}^{\lfloor N/q \rfloor} X_m \right| + \left| \frac{1}{C(N+d)} \sum_{m=1}^{\lfloor (N+d)/q \rfloor} C_m \right|.$$

Now, $\left| \frac{1}{C(N+d)} - \frac{1}{C(N)} \right| = O(1/N^3)$ and $\sum_{m=1}^{\lfloor N/q \rfloor} X_m = O(1/N^2)$, so the first term is $O(1/N)$. The second term is bounded above by

$$\frac{1}{C(N+d)} \sum_{m=\lceil 1 \rceil}^{\lfloor (N+d)/q \rfloor} dK = \frac{dK \lfloor (N+d)/q \rfloor}{C(N+d)}$$

and since $C(N+d) \sim O((N+d)^2)$, we have overall that

$$|\tilde{\mathbb{P}}[X \circ y](p/q) - \tilde{\mathbb{P}}[X](p/q)| = O\left(\frac{1}{N}\right).$$

Now, $\tilde{\mathbb{P}}[T'](\omega) = \lim_{N \rightarrow \infty} \tilde{\mathbb{P}}_N[T'](\omega)$ where

$$\tilde{\mathbb{P}}_N[T'](\omega) = \frac{1}{|T'_{\leq N}|} \sum_{X' \in T'_{\leq N}} \tilde{\mathbb{P}}[X'](\omega).$$

However,

$$\begin{aligned} \frac{1}{|T'_{\leq N}|} \sum_{X' \in T'_{\leq N}} \tilde{\mathbb{P}}[X'](\omega) &= \sum_{M=1}^N \frac{|T'_M|}{|T'_{\leq N}|} \frac{1}{|T'_M|} \sum_{X' \in T'_M} \tilde{\mathbb{P}}[X'](\omega) \\ &= \sum_{M=1}^N f_M^N(T') \frac{1}{|T'_M|} \sum_{X' \in T'_M} \tilde{\mathbb{P}}[X'](\omega) \\ &= \sum_{M=1}^{N-d} f_{M-d}^{N-d}(T) \frac{1}{|T_{M-d}|} \sum_{X \in T_{M-d}} \tilde{\mathbb{P}}[X](\omega) + O\left(\frac{1}{M-d}\right). \end{aligned} \quad (\text{D.2})$$

Hence

$$\lim_{N \rightarrow \infty} |\tilde{\mathbb{P}}_N[T'](\omega) - \tilde{\mathbb{P}}_{N-d}[T](\omega)| \leq \lim_{N \rightarrow \infty} \sum_{M=1}^{N-d} f_{M-d}^{N-d}(T) \frac{C}{M-d} = 0.$$

■

D.2 Computing $\tilde{\mathbb{P}}$ from \mathcal{F}

Proof: (Of Proposition 29.) For notational ease, define $a_l = \mathcal{F}_l[X]$. Now, by definition,

$$\tilde{\mathbb{P}}[X](p/q) = \frac{1}{C(N)} \sum_m^{\lfloor N/q \rfloor} \sum_{s=0}^{N-mq-1} |\mathcal{F}_{mp}(X(s : s + mq - 1))|^2.$$

Now, let's concentrate on $\mathcal{F}_{mp}(X(s : s + mq - 1))$. Because of the Fourier inversion property,

$$X(k) = \sum_{l=0}^{N-1} a_l e^{2\pi i l k / N}.$$

Plugging this into the definition of $\mathcal{F}_{mp}(X(s : s + mq - 1))$ we get

$$\mathcal{F}_{mp}(X(s : s + mq - 1)) = \frac{1}{mq} \sum_{k=0}^{mq-1} X(k+s) e^{-2\pi i k \frac{mp}{mq}} = \frac{1}{mq} \sum_{k=0}^{mq-1} \sum_{l=0}^{N-1} a_l e^{2\pi i l (k+s)/N} e^{-2\pi i k p/q},$$

and changing the order of summation gives

$$\mathcal{F}_{mp}(X(s : s + mq - 1)) = \frac{1}{mq} \sum_{l=0}^{N-1} a_l e^{2\pi i l s / N} \sum_k^{mq-1} e^{2\pi i k \left(\frac{l}{N} - \frac{p}{q}\right)}.$$

Now,

$$\sum_{k=0}^{mq-1} e^{2\pi i k \left(\frac{l}{N} - \frac{p}{q}\right)} = \frac{e^{2\pi i l m q} - 1}{e^{2\pi i \left(\frac{l}{N} - \frac{p}{q}\right)} - 1} = e^{\pi i (p/q - l/N + l m q/N)} \frac{\sin(\pi q l m/N)}{\pi \left(\frac{l}{N} - \frac{p}{q}\right)}.$$

Putting this in, and taking the norm-square, we get

$$|\mathcal{F}_{mp}(X(s : s + mq - 1))|^2 = \left(\frac{1}{mq}\right)^2 \sum_{l,k} (a_l \otimes \bar{a}_k)^\dagger e^{-\pi i (k-l)(2s+mq-1)/N} \frac{\sin(\pi m q \frac{l}{N}) \sin(\pi m q \frac{k}{N})}{\sin(\pi \left(\frac{l}{N} - \frac{p}{q}\right)) \sin(\pi \left(\frac{k}{N} - \frac{p}{q}\right))}.$$

Summing over s gives

$$\sum_{s=0}^{N-mq-1} |\mathcal{F}_{mp}(X(s : s + mq - 1))|^2 = \left(\frac{1}{mq}\right)^2 \sum_{l,k} (a_l \otimes \bar{a}_k)^\dagger A(s) \frac{\sin(\pi q l m/N) \sin(\pi q k m/N)}{\sin(\pi \left(\frac{l}{N} - \frac{p}{q}\right)) \sin(\pi \left(\frac{k}{N} - \frac{p}{q}\right))}$$

where

$$A(s) = \sum_{s=0}^{N-mq-1} e^{-\pi i (k-l)(2s+mq-1)/N}.$$

It is easy to compute that

$$A(s) = \begin{cases} N - mq, & l = k \\ e^{2\pi i (k-l)/N} \frac{\sin((k-l)\pi m q/N)}{\sin((l-k)\pi/N)}, & l \neq k \end{cases}.$$

Hence, summing over m and collecting conjugate terms,

$$\begin{aligned} \tilde{\mathbb{P}}[X](p/q) &= \sum_l \|a_l\|^2 \frac{1}{C(N)} \sum_m \frac{N - mq}{(mq)^2} \left(\frac{\sin(\pi m q \left(\frac{l}{N} - \frac{p}{q}\right))}{\sin(\pi \left(\frac{l}{N} - \frac{p}{q}\right))} \right)^2 \\ &+ \sum_{k \neq l} \frac{(a_l \otimes \bar{a}_k)^\dagger e^{2\pi i (k-l)/N} + (a_k \otimes \bar{a}_l)^\dagger e^{2\pi i (l-k)/N}}{2C(N)} \sum_m \frac{1}{(mq)^2} \frac{\sin((k-l)\pi m q/N)}{\sin((l-k)\pi/N)} \frac{\sin(\pi m q \left(\frac{l}{N} - \frac{p}{q}\right))}{\sin(\pi \left(\frac{l}{N} - \frac{p}{q}\right))} \frac{\sin(\pi m q \left(\frac{k}{N} - \frac{p}{q}\right))}{\sin(\pi \left(\frac{k}{N} - \frac{p}{q}\right))} \end{aligned}$$

where $C(N)$ is the normalization function from above. Now,

$$(a_l \otimes \bar{a}_k)^\dagger e^{2\pi i (k-l)/N} + (a_k \otimes \bar{a}_l)^\dagger e^{2\pi i (l-k)/N} = \cos\left(\frac{2(k-l)\pi}{N}\right) (a_l \otimes \bar{a}_k + a_k \otimes \bar{a}_l)^\dagger.$$

Hence,

$$\tilde{\mathbb{P}}[X](p/q) = \sum_{k,l} (a_l \otimes \bar{a}_k + a_k \otimes \bar{a}_l)^\dagger f_{N,p/q}(l/N, k/N)$$

where

$$f_{N,p/q}(l/N, k/N) = \frac{1}{2C(N)} \begin{cases} \sum_m \frac{N-mq}{(mq)^2} \left(\frac{\sin(\pi m q \left(\frac{l}{N} - \frac{p}{q}\right))}{\sin(\pi \left(\frac{l}{N} - \frac{p}{q}\right))} \right)^2, & l = k \\ \sum_m \frac{1}{(mq)^2} \frac{\sin((k-l)\pi m q/N)}{\sin((l-k)\pi/N)} \frac{\sin(\pi m q \left(\frac{l}{N} - \frac{p}{q}\right))}{\sin(\pi \left(\frac{l}{N} - \frac{p}{q}\right))} \frac{\sin(\pi m q \left(\frac{k}{N} - \frac{p}{q}\right))}{\sin(\pi \left(\frac{k}{N} - \frac{p}{q}\right))}, & l \neq k \end{cases}.$$

Now, we have to show that f is a delta function as advertised. First let's take the case $k = l$. Consider the function

$$G(N, t, q) = \frac{1}{N} \sum_{m=1}^{\lfloor N/q \rfloor} (\sin(\pi m q t))^2 \frac{(N - mq)}{(mq)^2}.$$

Putting aside issues of differentiability for the moment, the second derivative of G with respect to t is easily computed to be:

$$\frac{\partial^2 G}{\partial t^2} = \frac{1}{N} \sum_{m=1}^{\lfloor N/q \rfloor} 2\pi^2 (N - mq) \cos(2\pi m q t).$$

This is easily evaluated in closed form using standard techniques, although the formula is unwieldy. If tq is integral, there is a singularity; for tq non-integral, for large N we can approximate $\lfloor N/q \rfloor$ by N/q , and compute

$$\frac{\partial^2 G}{\partial t^2} \sim -\pi^2 + \frac{\pi^2 q \csc^2(\pi qt) \sin^2(\pi Nt)}{N} \sim -\pi^2.$$

Hence, in the large N limit,

$$G(N, t, q) \sim -\frac{\pi^2}{2} t^2 + Bt + C$$

for some constants B and C . To evaluate these constants, notice that $t = 0, 1/q, 2/q, \dots$, $G(N, t, q) = 0$, so $C = 0$ and $B = \frac{\pi^2}{2q}$. Plugging this information back into the formula for f , we see that

$$f_{N,p/q}(l/N, l/N) \sim \frac{\pi^2 q}{q+N} \left| \frac{1}{q} - \frac{l}{N} \right| \frac{l}{N} \frac{1}{\sin^2 \left[\pi \left(\frac{l}{N} - \frac{p}{q} \right) \right]}$$

where l is calculated modulo N/q . This is a discrete delta function, for on the one hand, the second-order polynomial in the numerator is bounded over its whole range, and scales with N as $O(1/N)$. When bounded from p/q , the sine term in the denominator is bounded, so the ratio scales as $O(1/N)$ away from p/q . At $l/N \sim p/q$, both the numerator and the denominator go to zero, at the same rate. To see this, consider

$$f_{N,p/q}((p/q)N + k, (p/q)N + k) \sim \frac{k\pi^2(N + kq)}{N^2(N + q)} \csc^2(\pi/q + \pi(k/N - p/q)).$$

Taking the Taylor series of the RHS about $N = \infty$, we get

$$f_{N,p/q}((p/q)N + k, (p/q)N + k) \sim \frac{1}{k} + q(1 - (1/k)) \frac{1}{N} + O(1/N^2).$$

Notice that at $k = 0$, $f_{N,p/q}(p/q, p/q) = 1$.

In the $k \neq l$ case, it is again easy to see that the numerator is bounded over its range and scales as $O(1/N^2)$ with N , while denominator is bounded away from either k or $l = (p/q)N$ and does not scale with N , so away from p/q , the function goes to zero with N . The computation for asymptotics near either k or $l = (p/q)N$ for the $k \neq l$ case is somewhat more onerous than the $k = l$, but delivers essentially the same result. In particular, consider the function

$$H(N, q, t, s) = \sum_{m=1}^{\lfloor N/q \rfloor} \frac{1}{(mq)^2} (\sin(\pi mqt)) (\sin(\pi mqs)) (\sin(\pi mqt - s)).$$

Computing its second partial derivatives, one finds that

$$\frac{\partial^2 H}{\partial s^2} = -2\pi^2 \sum_{m=1}^{\lfloor N/q \rfloor} \cos(\pi(s - 2t)mq) \sin(\pi smq),$$

$$\frac{\partial^2 H}{\partial t^2} = 2\pi^2 \sum_{m=1}^{\lfloor N/q \rfloor} \cos(\pi(2s - t)mq) \sin(\pi tmq),$$

and

$$\frac{\partial^2 H}{\partial s \partial t} = \pi^2 \sum_{m=1}^{\lfloor N/q \rfloor} \sin(2\pi(s - t)mq).$$

These sums can be evaluated in closed form, though unlike in the case of $k = l$ it turns out to be impossible here to take a simple limit as $N \rightarrow \infty$. Instead, it turns out to be useful to define the following function, known as the Clausen function:

$$Cl(\theta) = \sum_{n=1}^{\infty} \frac{\sin(n\theta)}{n^2}.$$

Now if we consider

$$\tilde{H}(t, s) = \frac{1}{4q^2} [Cl(2\pi qt) + Cl(2\pi(s-t)q) - Cl(2\pi s)]$$

and compute its second partials, they are identical to those computed for H , except with $\lfloor N/q \rfloor$ replaced with ∞ . (Of course, those sums are seemingly diverging, but this relation to the Clausen function gives them meaning.) Hence, we can take \tilde{H} as an approximation to H . The key fact is that there is a standard integral representation due to Clausen:

$$Cl(\theta) = - \int_0^\theta \log \left[2 \sin \left(\frac{t}{2} \right) \right] dt.$$

Using this allows for the computation of the asymptotic expansion of H around $t = 1/q$ of the form

$$H(N, q, t, s) \sim (C_1(f) + C_2(f) \text{Log}[g - 1/q])(tq - 1) + O((tq - 1)^2)$$

where $C_i(f)$ are functions of f (that are somewhat unwieldy but nonetheless obtainable in closed form). Hence, the upshot is the same as in $k = l$ case. ■

In the above argument, I used a variety of infinite series without regard to convergence, differentiated using series formulae without regard to differentiability, and committed a variety of related sins. To see why these were generally OK (say, in the $k = l$ case), notice that we could have first rewritten the function $G(N, t, q)$, via some pretty nasty algebra (using the Taylor series of the exponential function) as:

$$\begin{aligned} G(N, t, q) = & \frac{1}{4q^3} (2NH_2(\lfloor N/q \rfloor) + e^{-xy}\Phi(e^{-x}, 2, y) + e^{xy}\Phi(e^x, 2, y) + Li_2(e^{-x}) - Li_2(2, e^x)) + \\ & + \frac{1}{4q} \left(2H_1(\lfloor N/q \rfloor) + \frac{e^{-xy}}{y} \mathcal{H}(1, y, 1 + y, e^{-x}) + \frac{e^{xy}}{y} \mathcal{H}(1, y, 1 + y, e^x) + \text{Log}[1 - e^{-x}] + \text{Log}[1 - e^x] \right) \end{aligned} \quad (\text{D.3})$$

where $x = 2\pi itq$, $y = 1 + \lfloor N/q \rfloor$, and $H_{i=1,2}(N) = \sum_n^N (1/n^i)$ are the harmonic numbers of order 1 and 2, $\Phi(x, a, s) = \sum_{k=0}^\infty \frac{x^k}{(a+k)^s}$ is the so-called Lerch generalized polylogarithm, $Li_2(x) = \Phi(x, 1, 2)$ is the dilogarithm function, and \mathcal{H} is the standard hypergeometric function. The asymptotics of the two harmonic number terms are independent of t , while the other things are well understood in the literature to have good enough convergence derivatives for our purposes. The $k \neq l$ case is similar.

It is interesting to note that this re-writing in terms of well-known series functions is simply a more sophisticated version of the computation done above. On the one hand, the Lerch and hypergeometric function terms are very small and go away in the limit, and the $H_1(N)$ is also small ($\sim \log(N)/N$), as is as the pure logarithm term. The $H_2(N)$ term contributes a constant $\pi^2/3$, and the difference of dilogarithm terms yields the second order polynomial $2\pi^2(|f| - f^2 + (1/6))$. The $\pi^2/3$ constants cancel, leaving the homogenous quadratic.

D.3 Computing Examples of \tilde{P}

Finally, we'll compute some examples of \tilde{P} .

Example 47 Consider $T = \{(100)^n | n \in \mathbb{N}\}$. Notice that for $X \in T$, the original Fourier coefficients are $a_0 = a_{N/3} = a_{2/3} = 1$ and 0. Hence, by the theorem $\tilde{P}[X] \sim \frac{1}{3}(\delta_0 + \delta_{1/3} + \delta_{2/3})$. Since this is the same for all X , this is also the value of $\tilde{\mathbb{P}}[T]$. Now consider T' consisting of substrings of elements of $\{(100)^n | n \in \mathbb{N}\}$. Notice that $T' = \bigcup_{i,j=0,1,2} 0^i \circ T \circ 10^j$, i.e. T' is obtained from T by concatenating a finite number of finite strings both ends. Hence by the proposition $\tilde{\mathbb{P}}[T'] = \tilde{\mathbb{P}}[T]$. This computation says that for a pattern that "intuitively really has" a frequency correctly captured by the original definition of spectral density $\tilde{\mathbb{P}}$ it will still be captured by $\tilde{\mathbb{P}}$, and that $\tilde{\mathbb{P}}$ captures more things that intuitively make sense.

Example 48 Now let $T = \{0^n 1^n | n \in \mathbb{N}\}$, in the words, the half-proportion pattern $T_{1/2}$. For $X = 0^n 1^n$, recall that

$$\mathbb{P}[X] \left(\frac{k}{2n} \right) = \begin{cases} \frac{(\csc(\frac{k\pi}{2n}))}{n^2}, & k \text{ odd} \\ 0, & k \text{ even} \end{cases}.$$

Hence, for k/N bounded away from 0, these go to 0, and by the theorem, $\tilde{\mathbb{P}}[X](\omega) = 0$ for $\omega > 0$. At 0, it is easier to compute directly from the definition:

$$\tilde{P}[0^n 1^n] = \frac{1}{n(1+2n)} \left(\sum_{k=1}^n \left(2(n-k+1) + \frac{1}{k^2} \sum_{j=1}^{k-1} ((k-j)^2 + j^2) \right) + \sum_{k=n+1}^{2n} \frac{1}{k^2} \sum_{j=0}^{2n-k} ((k-n+j)^2 + (n-j)^2) \right).$$

Expanding the two main terms in the above expression separately, using simple formulas for sum-of-squares, &c, we find that

$$\begin{aligned} \tilde{P}[0^n 1^n] &= \frac{n(1+4n) + \mathcal{H}_n}{3n(1+2n)} + \frac{1}{n(1+2n)} \sum_{k=n+1}^{2n} \frac{(2n-k+1)(2k^2 + 2n(1+n) - k(1+2n))}{3k^2} \\ &= \frac{n(2+3n) + (1+6n(1+n))(\mathcal{H}_n - \mathcal{H}_{2n}) + 2n(1+n)(1+2n)(\psi_1(n+1) - \psi_1(1+2n))}{3n(1+2n)} \end{aligned}$$

where \mathcal{H}_n is the n -th harmonic number and ψ_1 is the first-order ‘‘polygamma’’ function (defined as the derivative of the digamma function ψ , which itself is the log derivative of the Γ function). Now, as $n \rightarrow \infty$, well-known asymptotics (as shown, for example, in Abramowitz and Stegun) show that

1. $2n(\psi_1(n+1) - \psi_1(1+2n)) \rightarrow 1$ and
2. $\mathcal{H}_n \sim \log(n)$.

Hence, as $n \rightarrow \infty$,

$$\tilde{P}[0^n 1^n] \sim \frac{n(2+3n) + (1+6n(1+n))(\log(n) - \log(2n))}{3n(1+2n)} \rightarrow \frac{3}{2} - \log(2).$$

Putting all this together shows that $\bar{P}[T_{1/2}] = \left(\frac{3}{2} - \log(2)\right) \delta_0$.

Appendix E

Generic Dynamic Encoding

The radius-state tradeoff defined above works only for a very specific class of algorithms. It is desirable to have a generic procedure for making such tradeoffs for any local rule, like the algorithms for the static tradeoff. This would involve finding a generally-applicable process for encoding the dynamics of one local rule in the operation of another, with different states and radius.

Even *defining* the notion of dynamic encoding properly, much less finding such encodings algorithmically, turns out to be difficult. The discussion in the previous section indicates that the size of the encoded structures, and the complexity of their internal substructures, increases with the number of different states to be encoded and the complexity of the transitions between them. As a result, the dynamic motion of these larger encoded structures will be somewhat slower than the dynamics of the original states they encode, at least in asynchronous timing models. For instance, the encoded trajectory shown in eq. 6.5 requires $2r + 6$ (asynchronous) timesteps in place of 2 (asynchronous) timesteps in the original trajectory 6.4. This is because the larger encoded structures contain internal substructures, and when only a few agents at a time are updated, the multiple parts of these substructures will require extra time to propagate and transition between.

This observation introduces some complication in properly defining the notion of a dynamic encoding. Suppose we're given a local rule F of radius r operating on configurations over state set S – that is, a mapping $F : \mathcal{B}_{r,S} \rightarrow S$. Then an encoding of F of radius r' over state set S' will of course have two elements: an encoded rule $F' : \mathcal{B}_{r',S'} \rightarrow S'$, and a decoding function $\phi : \mathcal{B}_{r',S'} \rightarrow S$. The dynamics of F' , once suitably decoded by ϕ , should track those of F . This means that some sort of commutation relationship should hold between ϕ , F , and F' :

$$\begin{array}{ccc} \mathcal{C}_{S'} & \xrightarrow{F'} & \mathcal{C}_{S'} \\ \downarrow \phi & \circlearrowleft & \downarrow \phi \\ \mathcal{C}_S & \xrightarrow{F} & \mathcal{C}_S \end{array}$$

One might be tempted to make this relationship precise by requiring commutation of the above diagram. That is, for all configurations X and agent calls $s \subset \{1, \dots, |X|\}$,

$$\phi(F'_s(X)) = F_s(\phi(X)). \tag{E.1}$$

This might be reasonable for the synchronous timing model, i.e. when s is the call to all agents. However exact commutation is usually, however, too strong a requirement for asynchronous environments. If we did mandate exact commutation, implying an exact matching of the trajectories, then the encoding described the previous section would be invalidated since the encoded version would, at least in asynchronous models, fall behind the original trajectory by several timesteps every time a $\triangleright \rightarrow \triangleright \circ \triangleright \rightarrow \triangleright$ transition occurred.

Put another way, a dynamic encoding can only really be expected to provide a “weak bisimulation” between the original rule and its encoded version. Given an exact accounting of time, nontrivial asynchrony will reveal gaps caused by the hidden operations with regard to which F and F' fail to be equivalent. Thus, we need a definition of encoding which allows for a looser tracking of the original trajectory by the encoded

trajectory. We can loosen the tracking quite a lot by simply asking that for any live call sequence s , there is some live subsequence s' such that the trajectory of the encoded system under s is a subtrajectory of the original system under s' . In other words, there is some way to selectively “stop and start the clock” of the original system, by calling it on a subsequence, such that the resulting trajectory can be viewed as a “slowed down” version of the encoded system’s trajectory. Formally,

Definition 37 [Loose Dynamic Encoding] *Given a call sequence $s = (s_1, s_2, \dots)$, a sub-call sequence of s is a call sequence $s' = (s'_1, s'_2, \dots)$ for which for some monotone increasing sequence j_1, j_2, \dots of natural numbers, $s'_i \subset s_{j_i}$. A pair (F', ϕ) , where $F' : \mathcal{B}_{r,s'} \rightarrow S'$ and $\phi : \mathcal{B}_{r,s'} \rightarrow S$ is a loose dynamic r' -encoding of $F : \mathcal{B}_{r,s} \rightarrow S$ if for all configurations X and all live call sequences s over X , there is a live sub-call sequence s' of s such that trajectory $\phi((F')_{s'}^n(X))$ is a subtrajectory of $F_s^n(\phi(X))$.*

It is not hard to verify that the replacement procedure defining F' from F_{Θ}^O is a loose dynamic encoding, with the decoding function ϕ simply codifying the translations described in words in the previous section. Moreover, we can verify that def. 37 is reasonably satisfactory for the problem of dynamic radius-state tradeoff by virtue of the following simple result.

Proposition 49 *Suppose $S = S_1 \cup S_2$ is a state set, and $T \subset \mathcal{C}_{S_1}$. Suppose also that $f : \mathcal{B}_{r,S} \rightarrow S$ is a solution to T in all live timing models and that (F, ϕ) is an R -encoding of f where $F : \mathcal{B}_{R,S_1} \rightarrow S_1$ and $\phi : \mathcal{B}_{R,S_1} \rightarrow S_1 \cup S_2$. Finally, suppose ϕ has the property that*

$$\phi(X) \in \mathcal{C}_{S_1} \Rightarrow \phi(X) = X. \quad (\text{E.2})$$

Then F is a solution to T in any live timing model.

In words, if we have a local rule f that solves a pattern using some extra state, an encoded version F of f that uses no extra states, and for which an decoding ϕ can be found that is the identity when ϕ detects no encodings of those excess states (eq. E.2), then the F must also solve the pattern.

Proof: Let X_0 be an initial configuration in \mathcal{C}_{S_1} such that $|X_0|$ is a T -admissible size, and let s be a live call sequence. There must be a live call sequence s' for which $\phi(F_s(X_0))$ is a subsequence of $f_{s'}(\phi(X_0))$, so $\{\phi(y) | y \in \Omega(F, s, X_0)\} \subset \Omega(f, s', X_0)$. Since f is a solution to T under all live call sequences, $\Omega(f, s', X_0)$ is a singleton subset of \mathcal{C}_{S_1} , so therefore there is a $z \in T \subset \mathcal{C}_{S_1}$ such that $\phi(y) = z$ for all $y \in \Omega(F, s, X_0)$. But then by the assumption in eq. E.2, $y = z$ for all such y , so that $\Omega(F, s, X_0) = \{z\}$, implying F is a solution. ■

One flaw in the definition 37, however, is that resulting encoded solution is not guaranteed to have the same runtime scaling as the original solution. To achieve this, we can add an additional requirement to def. 37 that the subsequences s' of s and $\phi(F_{s'})$ of $F(\phi(X))$ can always be taken to be reasonably dense. This would imply that the number of inequivalent hidden operations between the encoded and original systems is sufficiently small that their clocks need not be very distorted before their trajectories are expected to line up. In the encoding described above for the Naive backtracking algorithm, for example, the subsequence s' can always be taken to contain at least one complete round of agent calls in every $2r + 6$ complete rounds in s , and the resulting $\phi(F_{s'})$ will always contain one configuration in every $2r + 6$ timesteps of $F_{\Theta}^O(\phi(X))$. Hence, the runtime of F' is at worst a multiple of $(2r + 6)^2$ times the worst-case runtime of F_{Θ}^O . On the other hand, adding a “density” requirement of this kind introduces a parameter into the definition, which is itself unpleasant. It may be difficult to find a “clean” notion of dynamic encoding that is loose enough to be generally applicable but strong enough to guarantee runtime equivalence.

Whatever the ultimate definition of dynamic turns out to be, what I’d really like is to find an algorithm for making a dynamic radius state tradeoff. That is,

Wish 1 *I wish I could find computable functions*

$$\Gamma : \mathcal{D} \times \mathbb{N} \rightarrow \mathcal{D} \text{ and } \gamma : \mathbb{N} \rightarrow \mathbb{N},$$

where \mathcal{D} is the set of all local rules, such that $\Gamma(f, m)$ is a radius- $\gamma(r(f))$ encoding of f over m states.

Experience has shown me that this is probably hard, and may be impossible. The reason I have found it difficult is that the only approach I know to making dynamic radius state tradeoffs is to locate the “emergent

turing heads" induced by the local rule and, having isolated them, encode using a technique along the lines of the previous section. Making this approach algorithmic would thus include as a subproblem finding an algorithm to detect and analyze the induced turing heads that may arise in configurations of all sizes. Since there are finitely many local rules of a given radius and number of states, there must be a finite upper bound for each m and r on the "size" of the largest induced turing head, whatever that formally means. However, intuitive comparison to a number of other problems from the theory of cellular automata suggests that this bound will be an uncomputable function. This is why I've had to work on a case-by-case basis to achieve the goal of removing the requirement of extra state from the local rules defined in previous chapters.

Appendix F

Other Approaches to Pattern Description

The choice of description language made in §8.1.2 has a large impact on the nature of the compilers defined in §8.1.3. There are other approaches to pattern description aside from local feature invariance.

F.1 Direct Specification

One obvious candidate would be to just have the user specify a local check scheme directly, inputting which local neighborhoods were to be accepted and which were to be rejected. After all, a check scheme *is* in the end a finite description of a potentially infinite pattern set. However, as examples 12 and 13 in chapter 2 make clear, it would be enormously laborious and confusing to do this. The typical number of accepted neighborhoods even in a simple pattern is large, and it is often not intuitive to think about all the separate local agent views. Local feature invariance generates the same output, with a much smaller and more intuitive input.

F.2 Formal Logic

Another approach would be the use of formal logic. For each pattern, this would mean writing a logic formula that defined the pattern as a set. Given the structure of locally checkable patterns, first order logic would be a natural choice.

Roughly speaking, a *first order formula in variable* x_1, \dots, x_n is a symbol string composed of the symbols

$$\{', \neg', \wedge', \forall', \in', x_1, \dots, x_n\}$$

that encapsulates a “logical proposition” about the object x . The simplest propositions are those that express equality between two objects, i.e.

$$\phi(x_1, x_2) \triangleq [x_1 = x_2]$$

expresses the idea that the two objects x_1 and x_2 are the same.¹ Formulas are built up by composing the various symbols above. The ‘ \neg ’ symbol is “negation”, so that for any initial statement $\phi(x)$, the statement $\neg(\phi(x))$ expresses that “not $\phi(x)$ ” holds. The ‘ \wedge ’ symbol is the logical “AND”, so that $\phi(x_1) \wedge \psi(x_2)$ expresses that both $\phi(x_1)$ and $\psi(x_2)$ hold true. The ‘ \forall ’ symbol declares that some statement must hold for all x , so that $\forall x \phi(x)$ means that any possible x ’s (in whatever domain is understood), the statement $\phi(x)$ holds. The ‘ \in ’ symbol is a domain restrictor, so that $\forall y \in x, \phi(x)$ means that $\phi(x)$ is only required to hold for those objects y in x . The notions of “there exists” and the logical “OR” can be built out of these symbols. The notation

¹The symbol ‘ \triangleq ’ means “ ϕ is defined to be”, which is used instead of the usual ‘ $=$ ’ sign to prevent confusion with the logical symbol.

$\exists x\phi(x)$ is a shorthand for $\neg(\forall x(\neg\phi(x)))$ and $\phi(x) \vee \varphi(y)$ is shorthand for $\neg((\neg\phi(x_1)) \wedge \neg\varphi(x_2))$.

For example, the formula $\phi(x)$ defining x to be a set containing at least 2 distinct elements is:

$$\phi(x) \triangleq \exists y_1, y_2 \in x, \neg(y_1 = y_2).$$

A similar formula $\psi(x)$ could be used to describe x as having no more than 2 elements. Hence, $\psi(x) \wedge \phi(x)$ describes x as having size exactly 2. A formula built up from these kinds of operations gets longer and longer. The *length* of ϕ , $|\phi(x)|$ simply counts how many symbols were used to compose it.

Now, to make first-order formulas “speak about” multi-agent configurations, we have to “enrich” the logic to the point where it could describe multi-agent system configurations. To do this, we’d add symbols describing the internal agent states, and perhaps the basic spatial relationships between agents.² Once this is accomplished, we could, for example, write a formula $\phi(X)$ which held true if and only if configuration X contained at least 12 agents in state 2. Such formulas can describe any local check scheme Θ , simple by taking the “OR” over all Θ -accepted neighborhoods.

Just as with the local feature invariance approach, we’d then have to find some way to translate logical formulas into local check schemes. The key to doing this is to use results about the *inherent locality* of first-order logic. In the 1970s, Hanf, Gaifman, and others realized that any first-order logic formula $\phi(x)$ can be understood as a series of “local conditions” [20]. The basic intuition behind this is that the simplest, shortest, formulas only speak about objects that are very “close” to each other. To make a formula describe a relationship between two structures that are “far away” from each other, requires us to use several rounds of composition. In our model, this literally means that a short first-order formula can only describe relations between neighboring agents, while a longer one is required to increase the radius of the relationship. If we let $R(\phi)$ denote the radius at which formula ϕ can describe inter-agent relations, Hanf and Gaifman’s main point was to show $R(\phi)$ is bounded by a simple function of the length $|\phi(x)|$ of the formula. The reason for this is that:

- The logical operations \wedge and \neg do not change the locality radius. That is, $R(\neg\phi) \leq R(\phi)$ for all formulas ϕ , and $R(\phi \wedge \psi) \leq \max(R(\phi), R(\psi))$.
- And the quantifier \exists at most doubles the radius, so that $R(\exists x\phi(x)) \leq 2R(\phi)$.

Thus, building up formulas by composing simpler formulas, we see inductively that $R(\phi) \leq 2^{|\phi|}$. Intuitively, what is going on here is that the only way to make a formula describe a new relationship between one agent and another one nearby is to introduce an existential quantifier. Chains of such quantifiers are required to make the formula have a long-range radius.

These locality results were proved in the context of first-order logic without added constants like those required to make formulas talk about multi-agent configurations. However, it is easy to show that the results extend at the expense of small addition to the radius. As a result, we have that:

Proposition 50 *Any pattern defined by a first-order logic formula ϕ is locally checkable by a local check scheme $\Theta(\phi)$ with radius $r \leq 3^{|\phi|}$; and conversely, all patterns of the form $\Theta(\mathbb{C})$ for some local check scheme Θ are first-order definable.*

Because the inductive procedure for constructing the locality argument is computable at each step, $\Theta(\phi)$ can be algorithmically derived from ϕ . Hence, we could use formulas as the input to a global-local compiler by defining:

$$GC_{Logical}(\phi) = F_{\Theta(\phi)}.$$

This translation from logic to check scheme to local rule is elegant. But I have not focused on this approach because:

Experience shows logic is not a particularly efficient representation of patterns.

²The references [39, 40] describe a systematic way to do this.

That is, it's no easier to write a formula ϕ for a spatial pattern T than it is to use the local feature invariance approach, and in fact, usually much more laborious. In fact, the local feature invariance approach might be thought of as an easy way to generate logical formulas, in which the set-theoretic abstract of the logic can be forgotten. Perhaps for more complicated logics describing more sophisticated patterns the full logical approach will be useful.³ Logic descriptions can be useful in describing other self-organizing goal spaces [16].

³And then, the question becomes: Will the locality results still hold for the more complicated logics? After all, not all meaningful logical concepts are local. For instance, the concept of "connectedness" is nonlocal. There is no way to use local conditions, and therefore, first-order logic, to express the condition that the set of agents in configuration with a certain internal state are in an unbroken consecutive line.

Appendix G

Details of Pattern Space Analysis

G.1 Gross Structure of Pattern Space

It is useful to separate the question in terms of the isomorphism classes of balls described in appendix §A.1, which broke down into five classes: (i) Central balls, which have size $2r + 1$ – and only arise in configurations of size $\geq 2r + 1$; (ii) Left balls, of size between $r + 1$ and $2r$ or less – which arise at the left ends of in configurations of size $r + 1$ and greater; (iii) Right balls, also of size between $r + 1$ and $2r$ or less – arising at the right end of configurations of size $r + 1$ and greater; (iv) Small central balls, arising at the center of configurations of size between $r + 1$ and $2r - 1$; and (v) small balls, of size between 1 and r , arising as configurations of size less than r .

Let's focus for a moment on the central balls. They are equivalent to m -ary sequences of length $2r + 1$. Recall in §7.1, we defined the *DeBruijn graph* $DB(n, m)$ to be the directed edge-labeled graph

$$(V_{n,m}, E_{n,m}),$$

whose nodes $V_{n,m}$ are $[m]^n$, the m -ary sequences of length n , and whose edges are

$$E_{n,m} = \{(v_i, v_j, k) \in V(n, m) \times V(n, m) \times [m] \mid v_j = (v_i(2), v_i(3), \dots, v_i(n), k)\}.$$

It is immediately apparent from the definition that:

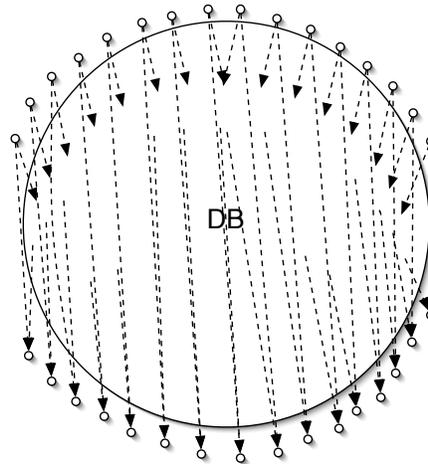
$$\mathbb{D}_{central}(r, m) = DB(2r + 1, m),$$

i.e. when we just focus on central balls $\mathbb{D}(r, m)$ reduces to the DeBruijn graph. Now, what about the rest? Let's add in just left and right balls of size exactly $2r$, i.e. considering nodes in

$$\mathcal{B}_{central}(r) \cup \mathcal{B}_{left,2r}(r) \cup \mathcal{B}_{right,2r}(r).$$

What are the induced edges on this? A left ball of size $2r$ consists of a pair $(x, (r, \star))$ where x is any m -ary sequence of length $2r$, while a right ball consists of $(x, (r + 1, \star))$. The only edges incident on a left ball are not from right balls or central balls, so we need only focus on out-going edges. There are m outgoing edges to the nodes $(xi, (r + 1, \star))$ in $\mathcal{B}_{central}(r)$. There is also an edge to $(x, (r + 1, \star))$, the right ball with the same underlying sequence. Similarly, the only incident edges on a right ball $(x, (r + 1, \star))$ are the left ball with the same underlying sequence, and the m edges from $(ix, (r + 1, \star))$.

Thus, the structure looks like:



Now, we add in the left and right balls, and small central balls, of size exactly $2r - 1$. Each left $2r - 1$ ball $(x, (r - 1, \star))$ has

- m left-end balls of size $2r$, i.e. $(x_i, (r, \star))$
- 1 small central ball of size $2r - 1$, i.e. $(x, (r + 1, \star))$.
- no incoming edges from any central or right-end balls or left-end balls of size $2r - 1$ or greater.

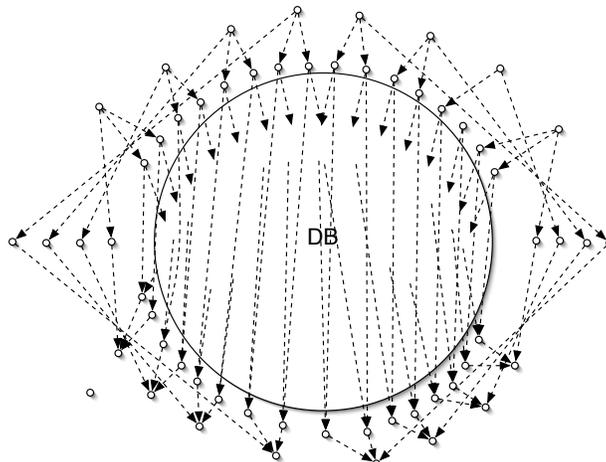
Each small central $2r - 1$ ball $(x, (r + 1, \star))$ has:

- one incoming edge from a left-end ball (account for above)
- one outgoing edge to a right-end ball $(x, (r + 1, \star))$.
- No other edges, whatsoever

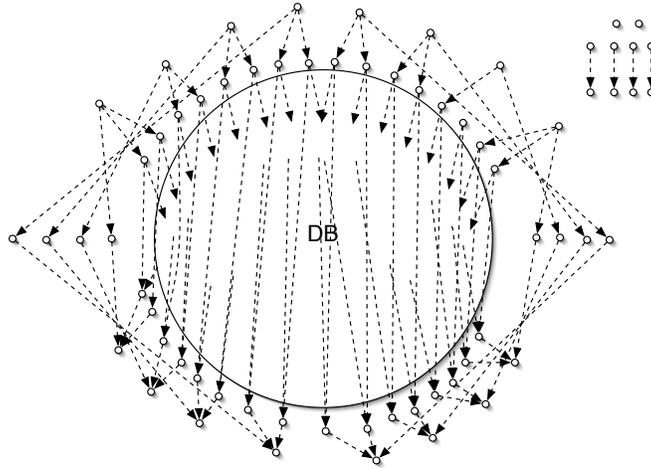
Each right-end ball of size $2r - 1$ $(x, (r + 2, \star))$ has:

- one incoming edge from a small central ball (accounted for above)
- m incoming edges from right-end balls $(ix, (r + 1, \star))$ of size $2r$
- no outgoing edges to any central, left-end balls or right-end balls of size $2r - 1$ or greater.

Hence, the picture is now:



Repeating this same computation $r - 2$ more times, exhausts all the balls except for the very small balls. These fit into line graphs that are not connected to the main body of the graph. Hence, overall we have:



In summary, $\mathcal{D}(r, m)$:

- Has one strongly connected component, namely the copy of $DB(n, m)$ corresponding to the m^{2r+1} central balls.
- Has $1 + \sum_{i=1}^r m^i = (m^{r+1} - 1)/(m - 1)$ connected components, one for the main component and the others for each of the small balls.
- All nodes in the strongly connected component have indegree and out-degree m .
- All nonterminal nodes corresponding to left-end balls have indegree 1 and out-degree $m + 1$.
- All nonterminal nodes corresponding to right-end balls have indegree $m + 1$ and outdegree 1.
- All nodes corresponding to small central or very small balls are belong to a single weakly connected component, having indegree and outdegree 1.
- Initial nodes (with indegree 0) correspond to left-most balls with the \star at position 1, and terminal nodes to right-most balls b with the \star at position $|b|$.

G.2 Proof of Prop. 40

Proof: Suppose C_1 and C_2 are k -flat cycles in $DB(n, m)$ such that $d = \text{dist}(C_1, C_2) \geq k + 1$. Let $x \in C_1$, $y \in C_2$ be points of closest approach, and let $y' = p^{d-1}(y, C_2)$. Notice that $d \leq n$. Let C'_1 and C'_2 be cycles in $DB(n - k + 1, m)$ such that $C_1 = \gamma^{k-1}(C'_1)$ and $C_2 = \gamma^{k-1}(C'_2)$ (which we know exist and are unique by the second part of proposition 39). C'_1 and C'_2 are 1-flat, i.e. irreducible, and $d' = \text{dist}(C'_1, C'_2) = d - k + 1 \geq 2$. If we could prove the result just for irreducible cycles (i.e. the $k = 1$ case), then we'd have that $C'_1 \circledast_b^a C'_2$ is an irreducible cycle, where $x = \gamma^{k-1}(a)$ and $y' = p^{d-1}(y, C_2) = \gamma^{k-1}(b)$. But then since γ commutes with \circledast , by the first part of proposition 39, we have

$$\gamma^{k-1}(C'_1 \circledast_b^a C'_2) = C_1 \circledast_{y'}^x C_2$$

is a k -flat cycle, as desired.

We must now show the $k = 1$ case. Let C_1 and C_2 be two irreducible cycles of $DB(n, m)$ such that $d(C_1, C_2) > 1$, with x, y , and y' defined as above. Then there is a path α from x to y with $|\alpha| = d$ and a path β from $y' = p^d(y^*, C_2)$ to $x' = c^d(x^*, C_1)$ of length d as constructed in corollary 4. Now, there are three cases:

Case (i): $d \leq \min(|C_1|, |C_2|)$. In this case, our operation is given by:

$$C_1 \circledast_{y'}^x C_2 = C_1(x' : x) \circ \alpha \circ C_2(y : y') \circ \beta.$$

Graphically, this is the situation illustrated in figure 7.9.

Now, for any set $X \subset DB$ let $N_1(X) = \{y \in DB \mid \exists x \in X \mid p(y) = p(x)\}$. Since $\alpha(1) = x$, irreducibility of C_1 implies $N_1(\alpha(1)) \cap C_1 = \{x\}$ and $\text{dist}(C_1, C_2) > 1$ implies $N_1(\alpha(1)) \cap C_2 = \emptyset$. Analogously, since $\alpha(|\alpha|) = y$,

$N_1(\alpha(|\alpha|)) \cap (C_1 \cup C_2) = \{y^*\}$. Then suppose $z \in N_1(\alpha(i)) \cap (C_1 \cup C_2)$ for $2 \leq i \leq |\alpha| - 1$. Then if $z \in C_2$, the path $\alpha(1 : i - 1) \circ z$ is a path of length $i < |\alpha| = d$ from x to z , contradicting the minimality of d . On the other hand, if $z \in C_1$ then $z \in P^{d-i}(y)$, yielding a path of length $d - i$ from C_2 to C_1 , which by the previous lemma is a contradiction to minimality. Thus:

$$N_1(\alpha) \cap (C_1 \cup C_2) = \{x, y\}.$$

For the same reasons, the path β satisfies

$$N_1(\beta) \cap (C_1 \cup C_2) = \{y', x'\}.$$

Finally, $N_1(\alpha) \cap N_1(\beta) = \emptyset$. To see this, suppose otherwise. If $(\alpha(i), \beta(j))$ is a valid edge in $DB(n, m)$; then $(\beta(j - 1), \alpha(j + 1))$ is also a valid edge. Hence $\alpha' = \alpha(1 : i) \circ \beta(j : d)$ and $\beta' = \beta(1 : j - 1) \circ \alpha(i + 1 : d)$ are paths from x to $c^d(x, C_1)$, and $p^d(y, C_2)$ to y , respectively. Obviously

$$|\alpha'| + |\beta'| = |\alpha| + |\beta| = d + d = 2d.$$

On the other hand, because C_1 and C_2 are irreducible cycles of length $> d$, the paths $C_1(x : x')$ and $C_2(y' : y)$ are acyclic, and because $d \leq n$, these must be the unique shortest paths by proposition 37 part i. Hence $|\alpha'| > d$ and $|\beta'| > d$ which contradicts $|\alpha'| + |\beta'| = 2d$. Putting all this together, yields that $C_1 \otimes_{y'}^x C_2$ is an irreducible cycle.

Case (ii): $\min(|C_1|, |C_2|) < d \leq \max(|C_1|, |C_2|)$. Then suppose wlog that $|C_1| > |C_2|$. Then note that $\text{dist}(x^*, \beta(d - |C_2|)) = d$. So let $y_1 = \beta(d - |C_2|)$ and define $\alpha_1 = \text{sp}(x^*, y_1)$. Note that $|\alpha_1| = d$ and $N_1(\alpha_1) \cap (C_1 \cup C_2) = \{x^*\}$ by similar arguments as above. In this case,

$$C_1 \otimes_{y'}^x C_2 = C_1(c^d(x^*, C_1) : x^*) \circ \alpha_1 \circ \beta(y_1 : c^d(x^*, C_1))$$

which is an irreducible cycle of length $|C_1(c^d(x^*, C_1) : x^*)| + |\alpha_1| + |\beta(y_1 : c^d(x^*, C_1))| = |C_1| - d + d + |C_2| = |C_1| + |C_2|$.

Case (iii): Suppose $d > \min(|C_1|, |C_2|)$. Retaining the definitions of y_1 and α_1 as above, now define $x_1 = \alpha_1(d - |C_1|)$ and $\beta_1 = \text{sp}(p^d(y^*, C_2), x_1)$. Again, $N_1(\beta_1) \cap (C_1 \cup C_2) = \{p^d(y^*, C_2)\}$ and $|\beta_1| = d$. Repeating these steps, we recursively define

$$y_i = \beta_{i-1}(d - |C_2|); \alpha_i = \text{sp}(x^*, y_i); x_i = \alpha_i(d - |C_1|); \beta_i = \text{sp}(p^d(y^*, C_2), x_i)$$

noting that in each case $|\alpha_i| = |\beta_i| = d$ and

$$N_1(\alpha_i) \cap (C_1 \cup C_2) = \{x^*\}, N_1(\beta_i) \cap (C_1 \cup C_2) = \{p^d(y^*, C_2)\},$$

preserving irreducibility at each step. Now applying the second part of proposition 38 inductively to α_i and β_i , to get (from the equality)

$$\beta_{i+1}(1 : 1 + |C_1| + i(|C_1| + |C_2|)) = \beta_i(1 : 1 + |C_1| + i(|C_1| + |C_2|))$$

and

$$\beta_{i+1}(2 + |C_1| + i(|C_1| + |C_2|)) \neq \beta_i(2 + |C_1| + i(|C_1| + |C_2|))$$

whenever $i < \lfloor d / (|C_1| + |C_2|) \rfloor$ and

$$\alpha_{i+1}(1 : 2 + (i + 1)(|C_1| + |C_2|)) = \alpha_i(1 : 2 + (i + 1)(|C_1| + |C_2|)),$$

and

$$\alpha_{i+1}(3 + (i + 1)(|C_1| + |C_2|)) \neq \alpha_i(3 + (i + 1)(|C_1| + |C_2|))$$

whenever $i < \lfloor d / (|C_1| + |C_2|) \rfloor - 1$. Hence for $i^* = \lfloor d / (|C_1| + |C_2|) \rfloor$, $\alpha_{i^*+1} = \alpha_{i^*}$ and $\beta_{i^*+1} = \beta_{i^*}$, implying that $\alpha_{i^*}(d) = \beta_{i^*}(d - |C_2|)$ and $\beta_{i^*}(d) = \alpha_{i^*}(d - |C_1|)$. Thus,

$$C_1 \otimes_{y'}^x C_2 = \alpha_{i^*}(d - |C_1| : d - 1) \circ \beta_{i^*}(d - |C_2| : d - 1).$$

Since irreducibility is preserved at each step, this is an irreducible cycle of length $|C_1| + |C_2|$. ■

Notice that by construction $N_1(C_1 \otimes_{y'}^x C_2) \cap (C_1 \cup C_2) \subset \{x, y'\}$.

G.3 Proof of Prop. 43

Proof: We need only construct irreducible cycles of all sizes between 1 and m^{n-1} , for then we can apply γ . To prove this, it will be useful to have the following:

Lemma 6 *If C is an irreducible cycle in DB , then there is an irreducible cycle of length $|m^{n-1}| - |C|$.*

Proof: (of lemma) If $|C| = |m^{n-1}|$, then the result is trivial. Thus, assume $|C| < |m^{n-1}|$, and let C^{big} be an irreducible cycle of size m^{n-1} , which exists as per proposition 42. Of course, $C \not\subseteq C^{big}$, since any irreducible cycle cannot contain any other cycle properly.

First let's suppose $|C \cap C^{big}| = |C| - 1$, i.e. all but one element of C – call it x – are in C^{big} . Let x_p be the unique element of $P(x) \cap C^{big}$ and x_c the unique element of $C(x) \cap C^{big}$. Then $x_c, x_p \in C$. Let $y = p(x_c, C^{big})$ and $z = c(x_p, C^{big})$. x is the sibling of z and let x' be the sibling of y . Now let $D = (C^{big} \setminus \{y, z\}) \cup \{x, x'\}$. This is essentially the inverse of the operation done in proposition 42. For this reason, D is easily seen to be union to two disconnected components, one of which contains, and thus is equal to, C . For that reason, the other component of D is of size $|C^{big}| - |C| = m^{n-1} - |C|$, proving the desired result.

Now suppose that $|C \cap C^{big}| = |C| - 2$, i.e. C^{big} is missing two elements of C . If the two gaps are separated, i.e. not consecutive, fill in one at a time as in the previous case. This will result in a final D of with three disconnected irreducible components, $C \cup C_1 \cup C_2$, such that $|C| + |C_1| + |C_2| = m^{n-1}$. But then apply corollary 8 to $D \setminus C$ to get a single irreducible cycle of size $|C_1| + |C_2| = m^{n-1} - |C|$ as desired. On the other hand if the two missing elements of C are consecutive, then a similar cut procedure can be done – except now the result will, like the case of one missing element, be two components, one of which is C and other of which has size $m^{n-1} - |C|$.

The case of several missing elements is a straightforward generalization. ■

In light of the lemma, we need only prove the existence of irreducible cycles of all sizes between 1 and m^{n-2} . So now consider the set H_0 consisting of the nodes whose numbers have residue 1 modulo m . In terms of m -ary sequences, $H_0 = \{ixi | x \in [m]^{n-2}\}$. The edges in H_0 are $(ijxi) \rightarrow (jxi j)$. Thus, for $y = ixi \in H_0$ the sequence of edges specified by reading off the digits of ix are valid in H_0 . But therefore the connected component containing y also contains $(y_1 y_2 \dots y_{n-1} y_1)$, $(y_2 y_3 \dots y_{n-1} y_1 y_2)$, \dots , and $(y_{n-1} y_1 \dots y_{n-2} y_{n-1})$; and infact, this is the whole connected component, comprising a loop of length n . If $y = (z_1, \dots, z_l)^k$ for $kl = n - 1$, then the length of underlying cycle is l . Hence the sizes of the connected components of H_0 are all divisors of $n - 1$, and every divisor of $n - 1$ is represented as the size of least one component of H_0 .

Recall the loops t_i which had length i , for $i < n$. If i is a divisor of $n - 1$, then t_i is a component of H_0 . If i is not a divisor of $n - 1$, then it is connected to precisely two components in H_0 , both of length $n - 1$. Modify H_0 by removing these two components and replacing them with t_i . The result, which we'll denote H_1 , is a set of irreducible cycles whose mutual distances are at least 2, all of whose sizes are at most $n - 1$, and all sizes $i \leq n - 1$ are representing. The total number of nodes in H_1 is

$$m^{n-1} - 2(n-1)\psi(n-1) + \sum_{d \nmid n-1} d,$$

where $\psi(n)$ is the number of integers between 1 and n not dividing n .

Now, let $A = \{a_1, a_2, \dots, a_N\}$ is a finite multiset of natural numbers $\leq l$, i.e. a collection of numbers in $[l]$ such that any element might appear several times. Then if each $i \in [l]$ is represented at least once in A , any number less than (or equal to) $\sum A = \sum_i a_i$ can be written as a sum of elements of A . Notice that for all n, m except for the case $n = 6, m = 2$,

$$2(n-1)\psi(n-1) - \sum_{d \nmid n-1} d \leq m^{n-2}.$$

Thus, in all cases except $n = 6, m = 2$, the sizes of H_1 are a multiset in which all elements are at most $n - 1$, all $i \leq n - 1$ appear at least once, and the total sum is bigger than m^{n-2} . Hence applying proposition 8, we are done except for the case $n = 6, m = 2$.

In the case $n = 6$ and $m = 2$, we must make irreducible cycles of size 16 or less. The cycles $t_i, i = 1, 2, 3, 4, 5$, combine to produce irreducible cycles of all sizes up to $1 + 2 + 4 + 5 = 15$; since there is a hamiltonian cycle H in $DB(5, 2)$, which has size 16, $\gamma(H)$ is an irreducible cycle of size 16 in $DB(6, 2)$. ■

G.4 Proof of Prop. 45

Proof: We need only compute $N(n, m)$, the number of hamiltonian cycles in $DB(n, m)$ and apply γ .

The $m = 2$ case is representative, and is simpler to describe. $DB(n, 2)$ is pair of trees (figure G.1). Now, suppose D is a hamiltonian cycle in $DB(n, 2)$ (red arrows in figure G.1 A). Our overall strategy is: i) show that the hamiltonian cycles in $DB(n, 2)$ are in 1-1 correspondence with irreducible covers of $DB(n, 2)$; ii) then, given any irreducible cover I of $DB(n, 2)$ we turn it into a 1-dispersed covering of $DB(n + 1, 2,)$ by applying γ ; we build 2^{n-1} distinct hamiltonian cycles around each 1-dispersed covering; iii) this establishes a recursive relationship for $N(n, 2)$ in terms of $N(n - 1, 2)$.

Step i): To make an irreducible cover from a hamiltonian cycle H , first cut the self-loop node off the cycle by removing the edges $(1, 2)$ and $(2^n, 1)$ and replacing with $(1, 1)$ and $(2^n, 2)$. The resulting object H_1 is a disjoint covering with two elements: the self-loop, and everything else. Now, move on to node 2, which is connected either to node 3 or to node 4. Let c_2 denote which one, i.e. $c_2 = c(2, H_1)$, the child of node 2 in H_1 . Let c'_2 be the other child, the one not connected to 2. Let $y_2 = p(c'_2, H_1)$, the parent of c'_2 . Remove the edges $(2, c_2)$ and (y_2, c'_2) and replace with $(2, c'_2)$ and (y_2, c_2) . The resulting object H_2 is a disjoint cover with three elements. (See figure G.1 B). What we've been doing in these cuts is applying \otimes^{-1} , the inverse of the sum operation.

To continue this process in general, suppose we've made i cuts, forming H_i , a disjoint cover with $i + 1$ components. Let $cp(x, H_i)$ denote the component of x in H_i . Find the first (with respect to the numbering) node $x_{i+1} \in H_i$ such that both children in $C(x_i)$ are in the same component, i.e. if we write $C(x_{i+1}) = \{c_{i+1}^1, c_{i+1}^2\}$, then $cp(c_{i+1}^1, H_i) = cp(c_{i+1}^2, H_i)$. Let $c_{i+1} = c(x_i, H_i)$ and c'_{i+1} be the unique element of $C(x_i) - \{c_{i+1}\}$. Let $p_{i+1} = p(c'_{i+1}, H_i)$. Then let

$$H_{i+1} = H_i - \{(x_{i+1}, c_{i+1}), \{(p_{i+1}, c'_{i+1})\} + \{(x_{i+1}, c'_{i+1}), \{(p_{i+1}, c_{i+1})\}$$

so that

$$H_i = H_{i+1} - (cp(x_{i+1}, H_{i+1}) \cup cp(c_{i+1}, H_{i+1})) + cp(x_{i+1}, H_{i+1}) \otimes_{p_{i+1}}^{x_{i+1}} cp(c_{i+1}, H_{i+1}).$$

Repeat the above steps until the end of the first tree has been reached, producing the cover $I(H)$ (see figure G.1 C). $I(H)$ has no remaining point both of whose children are in the same component, and so is an irreducible cover. The process as described is evidently reversible, showing that hamiltonian cycles in $DB(n, 2)$ are equivalent to irreducible covers of $DB(n, 2)$.

Step ii): $\gamma(I(H))$ is a 1-dispersed cover of $DB(n, 2)$ whose elements are 2-flat cycles (see figure G.2 A). Define a disjoint cover D of $DB(n, 2)$ according to the following procedure:

- On the nodes in the main body the first tree, let D simply replicate the edges implied by $\gamma(I)$ (green edges in figure G.2B).
- On the nodes in the main body of the second tree, let children for each tree be chosen arbitrarily (red edges in figure G.2B).
- The leaf edges are now determined by disjointness (purple edges in figure G.2B).
- The resulting object D may have several disjoint cycles (figure G.3A). Consider the leaf-nodes in the second tree that map to leaf nodes in the first tree, call the set L' . All of the cycles in D are accessible by edges from L' . Pick the the first node (lowest number) x_1 in L' such that the two children of x_1 are in different components. Let $c_1 = c(x_1, D)$ and c'_1 be the other child of x_1 and $p_1 = p(c'_1, D)$, and now sum at (x_1, p_1) , i.e.

$$D_1 = D - (cp(x_1, D) \cup cp(p_1, D)) + cp(x_1, D) \otimes_{p_1}^{x_1} cp(p_1, D).$$

Now find the next node x_2 in L' whose children are different components, and sum. Repeat this process until one component remains, a hamiltonian cycle (figure G.3B).

In the four items above, there was choice only the in the second step. Since there are 2^{n-1} nodes in the main body of the second tree, and each non-self-loop node has 2 possible choices (while the self-loop node has 1), there are a total of $2^{2^{n-1}-1}$ ensemble choices. Denote the ensemble choice u , and $D(I, u)$ the hamiltonian cycle the results from the above procedure.

Step iii:) The hamiltonian cycle $D(I, u)$ determines I and u uniquely. On the one hand, the edges in the main tree body of the second tree, $D(I, u)$ simple is equal to the edges chosen in u ; hence u can be read off directly. On the other hand, the edges in the top $n - 1$ rows of $D(I, u)$ are equal to the edges implied by $\gamma(I)$, while the edges between the $n - 1$ st and n th row may have changed in order to accommodate the sums in step 4 of the construction procedure. Call the the top $n - 1$ rows T_{n-1} . No image $\gamma(I)$ will contain node 2, so whichever node is not pointed to by 2 is the top node in $\gamma(I)$ – and by excluding neighbors, all the nodes in $T_{n-1} \cap \gamma(I)$ can be determined. The key thing is that these nodes actually uniquely determine I itself. This is because $T_{n-1} \cap \gamma(I)$ determines the edges of I in the entire first tree of the $DB(n - 1, m)$ (a node in $DB(n, m)$ determines an edge in $DB(n - 1, m)$ under γ^{-1}), and the requirement of irreducibility determines the rest of the edges).

Hence for each hamiltonian cycle H of $DB(n - 1, 2)$ and each choice ensemble u , $D(I(H), u)$ is a distinct hamiltonian cycle of $DB(n, 2)$. This means that there are $2^{2^{n-2}-1}$ distinct hamiltonian cycles of $DB(n, 2)$ for each one in $DB(n - 1, 2)$, and thus:

$$N(n, 2) = 2^{2^{n-2}-1}N(n - 1, 2) = 2^{2^{n-1}-n}N(2, 2) = 2^{2^{n-1}-n}$$

as claimed. Having seen the $m = 2$ case, the general case is similar. ■

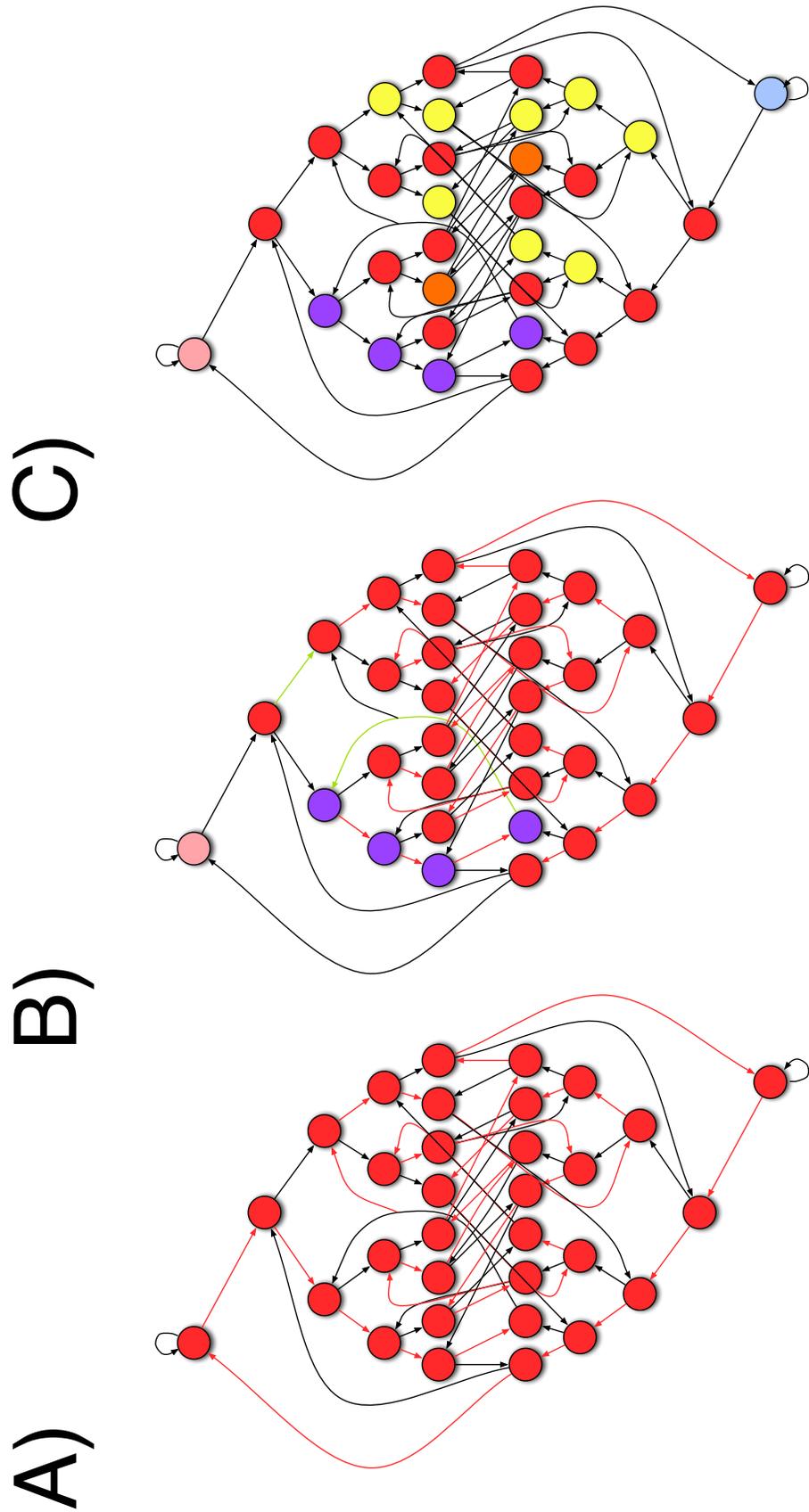


Figure G.1: A) $DB(5, 2)$ with edges of hamiltonian cycle highlighted in red. B) First two steps of process converting hamiltonian cycle to irreducible cover. C) Completed irreducible cycle resulting from hamiltonian cycle shown in A).

Appendix H

Drosophila Regulatory Networks

H.1 The Chao-Tang Model

The Chao-Tang model of the *Drosophila melanogaster* Segment Polarity Network [24] has the following components:

- *Engrailed* protein (square node labeled En) and mRNA (ellipse labeled en)
- *Mid* protein and mRNA
- *Sloppy* (Slp) protein and mRNA.
- *Wingless* (Wg) protein and mRNA
- *Hedgehog* (Hh) protein and mRNA
- *Cubitus interruptus* (CI) protein and mRNA,
- and *cubitus interruptus fragment* (CN).

Cubitus interruptus is produced basally, i.e. automatically without stimulation; whereas all other substances in the network are only produced if induced by some other substance in the network. All substances are degraded with first-order degradation processes. The interactions between components may be taken to be:

- The mRNA molecules induce production of associated proteins.
- *Engrailed* protein induces production of *hedgehog* mRNA.
- *Engrailed* protein inhibits product of *cubitus interruptus* mRNA.
- *Hedgehog* protein from a neighboring cell inhibits the cleavage of *Cubitus interruptus* protein into CN.
- *Cubitus interruptus* protein induces production of *wingless* mRNA.
- *Cubitus interruptus* repressor fragment inhibits production of *wingless* mRNA.
- *Wingless* protein from a neighboring cell induces production of *engrailed* and *sloppy* mRNAs.
- *Mid* protein inhibits production of *wingless* mRNA.
- *Sloppy* protein inhibits production of *mid* mRNA.

This model is summarized by the following regulatory network:

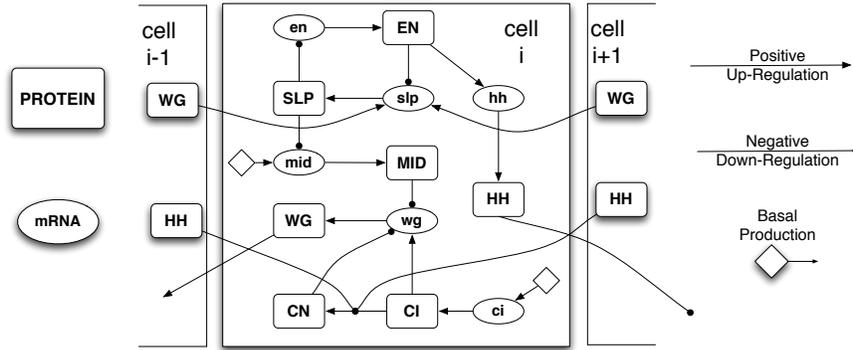


Figure H.1: The Chao-Tang SPN model network graph.

This network contains links between its own nodes, representing intra-cellular interactions, as well as links between nodes of other “copies” of the same graph, representing inter-cellular interactions.

Such a model can be captured mathematically in some generality. Let $[A]_i$ denote the concentration of substance A in cell i at time t . Let

$$[A - na]_i = \frac{[A]_{i-1} + [A]_{i+1}}{2}$$

denote the average of substance A 's concentration values from neighboring cells to cell i . Then the ODE equations corresponding to Chao-Tang model are generated according to the rules:

- First order degradation processes are represented by terms of the form:

$$\frac{d[A]_i}{dt} = -\frac{1}{\tau_A}[A] \quad (\text{H.1})$$

where τ_A is the half-life of the substance.

- The (positive) induction of substance B by substance A from the same cell is represented by:

$$\frac{d[B]_i}{dt} = \frac{1}{\tau_B} \frac{[A]_i^{n_{AB}}}{[A]_i^{n_{AB}} + k_{AB}^{n_{AB}}} \quad (\text{H.2})$$

where n_{AB} is the cooperative coefficient for the interaction and k_{AB} is the reaction rate (these are, theoretically, biologically measurable parameters).

- The (negative) inhibition of substance B by substance A from the same cell is represented by

$$\frac{d[B]_i}{dt} = \frac{1}{\tau_B} \frac{k_{AB}^{n_{AB}}}{[A]_i^{n_{AB}} + k_{AB}^{n_{AB}}} \quad (\text{H.3})$$

- The induction/inhibition of substance B by substance A from neighboring cells is represented by replacing $[A]$ with $[A - na]$ in eqs. H.3 and H.2, respective.
- When multiple interactions impinge on the same component, the effect is multiplicative, e.g. if A inhibits C and B induces C , then:

$$\frac{d[C]_i}{dt} = \frac{1}{\tau_C} \frac{k_{AC}^{n_{AC}}}{[A]_i^{n_{AC}} + k_{AC}^{n_{AC}}} \cdot \frac{[B]_i^{n_{BC}}}{[B]_i^{n_{BC}} + k_{BC}^{n_{BC}}} \quad (\text{H.4})$$

For example, the evolution of *sloppy* mRNA from the interaction model described above generated by these rules is:

$$\frac{d[slp]_i}{dt} = \frac{1}{\tau_{slp}} \left(\frac{[Wg - na]_i^{n_{Wg-slp}}}{[Wg - na]_i^{n_{Wg-slp}} + k_{Wg-slp}^{n_{Wg-slp}}} \cdot \frac{k_{En-slp}^{n_{En-slp}}}{[En]_i^{n_{En-slp}} + k_{En-slp}^{n_{En-slp}}} - [slp]_i \right).$$

The result of the choice in eq. H.4 to model multiple interactions as multiplicative is to “prefer” inhibitions over inductions, i.e. when an inhibitory link is active, it will overwhelm active inductive links. In terms of circuit logic, this says that multiple interactions coming into a node are always interpreted as AND gates.

The five above rules described in equations H.1-H.4 can be applied more generally than just to the specific network structure of the Segment Polarity Network described above. Given any network graph \mathcal{G} , and associated parameters n_{ij}, k_{ij} , there are the associated model of ODEs. Let’s call this the “Chao-Tang-form model associated with $\mathcal{G}, n_{ij}, k_{ij}$ ”, and denote it $F(\mathcal{G}, (n_{ij}, k_{ij}))$.

H.2 Analysis

A simple observation about the Chao-Tang-form models is that substances that only interact with themselves within the cell or respond to the concentrations of the same substance in neighboring cells only have at most one spatially uniform stable fixed point configuration. Suppose A were such a substance, and wlog suppose one spatially uniform stable fixed point is the 0 point. Then, the ODE describing it is either of the form

$$\frac{d[A]}{dt} = K_1 \frac{[A]^n}{[A]^n + K_2^n} \frac{K_3^m}{[A - na]^m + K_3^m} - K_4[A]$$

or

$$\frac{d[A]}{dt} = K_1 \frac{[A - na]^n}{[A - na]^n + K_2^n} \frac{K_3^m}{[A]^m + K_3^m} - K_4[A]$$

where K_{1-4} are positive constants and $n, m \geq 1$. In the former case, there would have to be some $\epsilon > 0$ that for all $a \in [0, \epsilon]$ that

$$\frac{K_1}{a^n + K_2^n} \frac{K_3^m}{a^m + K_3^m} < K_4 a^{n-1}.$$

But then $K_4 \geq K_1/K_2^n$, which makes 0 a global basin of attraction. The latter case is analyzed similarly.

A more sophisticated and general type of analysis is possible. Let \mathcal{G} denote a network graph (as in fig. H.1). Let a *real configuration* Y with n cells over \mathcal{G} denote a function Y whose arguments are pairs (i, A) , assigning the numerical concentration value for substance $A \in \mathcal{G}$ in cell i . Formally, Y is a function $Y : [n] \times \mathcal{G} \rightarrow \mathfrak{R}^{\geq 0}$, where $[n] = \{1, \dots, n\}$. A *binary configuration with n cells* is a function $X : [n] \times \mathcal{G} \rightarrow \{0, 1\}$. Given a threshold $\epsilon > 0$, define the ϵ -*thresholding of real configuration* Y , denoted Y_ϵ , to be the binary configuration given by $Y_\epsilon(i, A) = 0$ if $X(i, A) < \epsilon$ and 1 otherwise.

Given a network graph \mathcal{G} , and a binary configuration X over \mathcal{G} , we say X is *consistent with \mathcal{G}* if for all cells i and nodes A , $X(i, A) = 1$ if and only if

- There is at least one cell j neighboring i (or $j = i$) and one substance B in j such that $X(j, B) = 1$ and (j, B) is linked by a inductive edge to (i, A) .
- There are no j, B such that $X(j, B) = 1$ and (j, B) is linked by an inhibitory edge to (i, A) .

The key point is that:

Proposition 51 *Given a network graph \mathcal{G} and $\epsilon > 0$, there is an open set $\Omega(\mathcal{G}, \epsilon)$ in parameters space such that for each $q \in \Omega(\mathcal{G}, \epsilon)$, the associated Chao-Tang-form model $F(\mathcal{G}, q)$ satisfies:*

- A real configuration Y (of whatever number of cells) can only be a stable fixed point of $F(\mathcal{G}, q)$ whenever Y_ϵ is consistent with \mathcal{G} ; and

- For every binary configuration X consistent with \mathcal{G} , there is a stable fixed-point configuration Y of $F(\mathcal{G}, q)$ such that $Y_\epsilon = X$.

The proof of this result is a straightforward induction on the number of nodes and edges in \mathcal{G} . Now, the “pathway” identified in §8.2.3 is a periodic binary configuration that is consistent with the network shown in figure H.1 (for any number of cells). Hence, there is a nontrivial parameter regime for the Chao-Tang segment-polarity model on which it corresponds to a stable fixed point.