

# Tutorial Notes: Programming a One-Dimensional Amorphous Computer

Dan Yamins

July 5, 2007

## Contents

<b>1</b>	<b>A Simple One-Dimensional Model</b>	<b>2</b>
1.1	Agents and Configurations . . . . .	2
1.2	Local Rule Dynamics . . . . .	2
1.3	Timing Models . . . . .	3
1.4	Patterns . . . . .	4
1.5	Solutions . . . . .	4
1.5.1	Run-Time Complexity . . . . .	5
1.6	A Very Simple Example . . . . .	5
<b>2</b>	<b>Local Checkability</b>	<b>7</b>
2.1	A Necessary Condition for Solvability: Local Checkability . . . . .	7
2.1.1	Local Check Schemes as Parts List . . . . .	8
2.2	Local Checkability is Sufficient: A General Construction . . . . .	9
2.2.1	Step 1: Propagating Correct Structure . . . . .	9
2.2.2	Step 2: Recovering from Error . . . . .	10
2.2.3	Putting It Together . . . . .	11
<b>3</b>	<b>Graph-Based Analysis</b>	<b>12</b>
3.1	Local Check Schemes as Graphs . . . . .	12
3.2	Detailed Analysis . . . . .	12
3.2.1	Line Graph . . . . .	12
3.2.2	Single Cycle Graph . . . . .	13
3.2.3	Multi-Cycle Linear Graph . . . . .	13
3.2.4	One Non-Trivial SCC . . . . .	14
3.2.5	Multi-SCC Linear Graphs . . . . .	15
3.3	The General Case . . . . .	15
<b>4</b>	<b>Faster Algorithms</b>	<b>17</b>
4.1	A Linear-Time Algorithm . . . . .	17
4.2	Patterns that Admit Local Patching . . . . .	19
4.2.1	The Trivial Case . . . . .	22
4.3	Optimality . . . . .	22
<b>5</b>	<b>The Radius-State Tradeoff</b>	<b>23</b>
5.1	Coordinates and The Static Tradeoff . . . . .	23
5.2	The Dynamic Tradeoff . . . . .	26
<b>6</b>	<b>“Glocal” Compilation</b>	<b>27</b>
6.1	Sampling for Local Features . . . . .	28
6.2	Logic . . . . .	28
<b>7</b>	<b>Appendices</b>	<b>29</b>

# 1 A Simple One-Dimensional Model

## 1.1 Agents and Configurations

**Definition 1 [Agent Configurations]** 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. Let  $S$  be a finite set. 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  $s_i \in S$ , i.e. each node  $i \in \{1, \dots, n\}$  has been assigned a label  $s_i \in S$ . Let  $C_{n,S}$  be the set of all  $S$ -configurations of size  $n$ , and let  $C_S = \bigcup_{n \in \mathbb{N}} C_{n,S}$  – these are all configurations of all (finite) sizes. We will drop the  $S$  from the notation whenever it is obvious from context.

Informally, the directed line graph of size  $n$  represents the underlying geometry of our system, and looks like this:

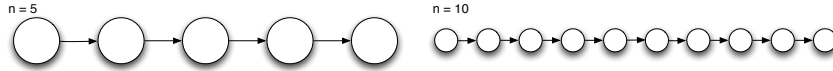


Figure 1: Two examples of finite directed lines,  $L_5$  and  $L_{10}$ .

Nodes of the graph correspond to agents. The nodes are arranged geometrically into a 1-D lattice, with the spatial relations expressed by the directed edges between the nodes.

By definition, a configuration is a version of the directed line graph in which the nodes have been labelled by elements of the set  $S$ . A typical configuration looks like this:

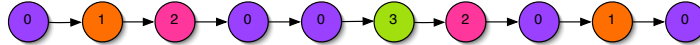


Figure 2: A ten-agent configuration with four internal states: purple (0), orange (1), pink (2), and green (3).

These labels (represented as colors in the picture) correspond intuitively to the agents' internal states. 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 also think of 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  $x_2$ .

**Definition 2 [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

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

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 2, the local ball of radius 2 around agent 5 looks like: . Given any local ball  $B$  let  $\star(B)$  denote the position of the starred agent. 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  $B^i = B(i - r : i + r)$ , in local coordinates.

## 1.2 Local Rule Dynamics

**Definition 3 [Local Rules]** A local rule of information radius  $r$  is any function from local balls of radius  $r$  to the set of states  $S$ :

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

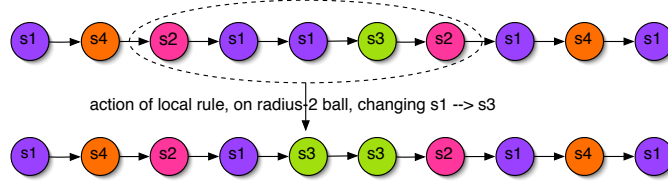
We denote the information radius of  $f$  by  $r(f)$ . Let  $\mathcal{D}_r$  denote the set of all local rules of radius  $r$ , and let  $\mathcal{D} = \bigcup_{r \in \mathbb{N}} \mathcal{D}_r$ .

Intuitively, dynamics are generated by agent-based state-change programs running identically on each of the agents in the 1-D multi-agent system, and drawing information only from other nearby agents. The mapping  $f$  represents a look-up table by which an agent  $a$  takes local information (given by the radius- $r$  ball  $B_r(a, X)$ ) and chooses a new state to adopt as a function of what it sees. By definition, whenever local balls  $B_1$  and  $B_2$  have the same graph structure (even if they arise at different positions), 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. Because of the  $\star$ , the graph structure of the end-agents is distinct from center agents, and end-agents therefore can use that information to act differently.

Given an agent  $a$  in configuration  $X$ , the *action of  $f$  on  $X$  at  $a$* , denoted  $f(a, X)$ , is the new configuration obtained from  $X$  by changing the label of  $a$  from whatever it is to  $f(B_{r(f)}(a, X))$ . For example, suppose we chose a rule defined by

$$f((\text{pink}, \text{purple}, \text{purple}^\star, \text{green}, \text{pink})) = \text{green}$$

on agent 5 of the configuration depicted in figure 2. The action of  $f$  on that configuration by agent 5 would look like:



By virtue of the definitions above, this model of dynamics does not allow for cell birth and death, or mobile agents. We can allow  $f$  to act on more than one agent simultaneously: in analogy to the above, let  $c$  be a *subset* of agents in  $X$ , and define  $f(c, X)$  to be the configuration obtained by replacing the labels of each  $a \in c$  with  $f(B_{r(f)}(a, X))$ .

**Definition 4 [Trajectories]** A size- $n$  call sequence is a 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 configuration  $X$  and a size- $|X|$  call sequence  $c$ , the trajectory of  $X$  generated by  $(f, c)$  is the set

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

where

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

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

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  $X$ . The omega-limit is the state (or states) to which the system eventually converges. Note that since  $\mathcal{C}_{n,S}$  is a finite set of size  $|S|^n$ , and since  $f(a, X) \in \mathcal{C}_{n,S}$  if  $X \in \mathcal{C}_{n,S}$ , that the omega-limit of all trajectories is non-empty. If  $\Omega(f, c, X)$  contains a single configuration, then we say that the trajectory *has a well-defined limit* denoted  $\lim(f, c, X)$ .

### 1.3 Timing Models

**Definition 5 [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 size- $n$  call sequences. 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: A) 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. B) 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. C) 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, but in which 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 it can be shown that  $R_{\mathcal{S}^a}(n) \sim C'n \log(n)$  for a constant  $C'$ .

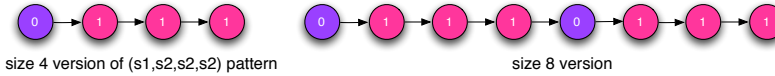
## 1.4 Patterns

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  $T$  of configurations:

**Definition 6 [Patterns]** A pattern is any subset  $T$  of configurations, i.e.  $T \subset \mathcal{C}_S$ . Let  $\mathcal{T} = 2^{\mathcal{C}_S}$  denote the set of patterns.

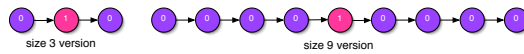
In this case, the elements  $X \in T$  are *instances* of the pattern, and the pattern itself is the totality of its instances. More abstracted notions of pattern will be later on.

**Example 1** For example, a pattern might have one agent in state 0, followed by three agents in state 1, like this: . By repeatedly concatenating this segment with itself we obtain a version of the 0111 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  $\{q^m \mid m \in \mathbb{N}\}$ . For example, if  $q = 1$  then  $T_q = \{1, 11, 111, 1111, \dots\}$ . If  $q = 12$  then  $T_q = \{12, 1212, 121212, \dots\}$ . Evidently  $T_q \cap \mathcal{C}_n \neq \emptyset$  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  $n$ .

In contrast, now define the pattern  $T_{1/2}$  of odd-size configurations, in which all states are 1, except for the center agent:



$T_{1/2}$  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.

## 1.5 Solutions

The central goal of this discussion 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  $\mathcal{C}$  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:

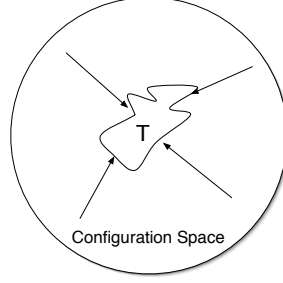


Figure 3: A local rule  $f$  solving  $T$  drives arbitrary configurations in  $\mathcal{C}$  to  $T$ .

**Definition 7 [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  $s \in \mathcal{S}_n$ , 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,

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

Let  $\mathcal{F}_{\mathcal{S}}(T)$  denote the set of solutions  $F$  to  $T$  in timing model  $\mathcal{S}$ .

We impose the condition that  $T \cap \mathcal{C}_n \neq \emptyset$  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  $\mathcal{C}$  to the prescribed pattern  $T$  (fig. 3).

### 1.5.1 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  $M \geq N$ . This measure depends on initial condition and call sequence, and we want to remove these dependencies. Also, because  $T(f, X, s)$  will typically scale with  $|X|$ , what we really want to capture is the scaling dependence.

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:

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

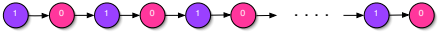
taking the uniform distribution over call sequences. Because  $T(f, X)$  will typically scale with  $|X|$ , what we really want to capture is the scaling dependence. We can either average over all  $|X|$  of a given size, producing the *average-case runtime* scaling function:

$$T_{\mathcal{S}}^{avg}(f)(n) = \frac{1}{|T \cap \mathcal{C}_{\leq n}|} \sum_{X \in T \cap \mathcal{C}_{\leq n}} T_{\mathcal{S}}(f, X)$$

or for each size find that  $X$  which has the longest time, producing the *worst-case runtime*:

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

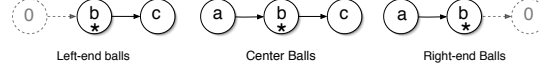
## 1.6 A Very Simple Example

To make this concept concrete, let's look at a simple example. 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

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

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.

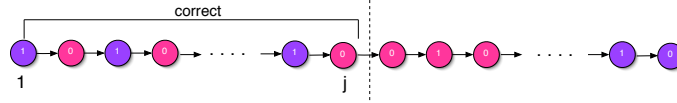
Now, define the local rule  $F_{10}$  according to the formula:

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

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  $\text{odd}(j)$  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  $m > 0$ ,

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

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. In regard to run-time complexity, it is not hard to see that in the above example

**Proposition 2** In all live uniform timing models,  $F_{10}$  has an average and worst-case runtime that scales linearly with the number of agents.

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  $f((1^*, 0) = f((1, 0^*, 0) = f((0, 0^*, 0) = f((0, 0^*) = 1$ . But then 10000000 must also be a fixed point of  $f$ ; since this is not in  $T_{1000}$ ,  $f$  cannot be a solution. ■

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.

These results for two specific patterns can be generalized. In section 2, I describe an effective simple necessary criteria, called “local checkability”, for discriminating patterns that have local solutions from those that don’t. I then show that local checkability is also sufficient by constructing a simple and generic (but potentially slow) solution for each locally checkable pattern. In section 3 I describe a graph-theoretic characterization of locally checkable patterns, and use the technique to compute a detailed runtime performance analysis of the solutions construction in the previous section. In section 4, I then use the graph characterization to construct significantly faster (and in fact, optimal) local solutions for each locally checkable pattern. In section 5, I explore a resource tradeoff between radius and state in local rules. In section 6, I integrate the results of the previous sections to construct a “global-to-local” compiler, a procedure whose input is a finite global pattern description and whose output is an optimally fast local rule solving that pattern.

## 2 Local Checkability

### 2.1 A Necessary Condition for Solvability: Local Checkability

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

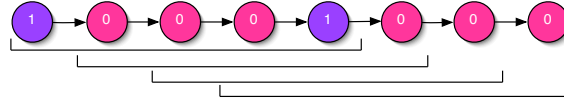
$$\lim_{n \rightarrow \infty} f_s^n(X) = Y$$

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

$$f(B_r(i, Y)) = B_r(i, Y)(0).$$

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  $T \cap \mathcal{C}_n \neq \emptyset$ , 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 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  $\Theta$  on  $\mathcal{B}_r$ , i.e.

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

$\Theta$  should be thought of as a “recognition function” –  $\Theta(z) = 1$  means that the local ball  $z$  is “recognized” by the central 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 8 [Local Check Schemes]** Let  $T$  be a pattern.  $\Theta$  is a local check scheme for  $T$  of radius  $r$  if

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

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  $r(f) \geq LCR(T)$ .

*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  $X \in \mathcal{C}_{n,S}$ , let  $Y = \lim(f, s, X)$ ; then  $Y \in \mathcal{C}_{n,S}$  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  $f(B_{r(f)}(a, X)) = l(a)$ . 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  $Y \in \mathcal{C}_{n,S}$  such that  $\Theta(Y) = 1$ . Thus  $\Theta$  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  $\Theta$  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 2** The pattern  $T_{1000}$  is 2-locally checkable, with local check  $\Theta(b)$  given by  $\Theta(b) = 1$  if  $b = 010^*00, 001^*00, 100^*01, 000^*10, 1^*000, 10^*00, 100^*0$ , or  $1000^*$ , and  $\Theta(b) = 0$  in all other cases.

A slight generalization of the above construction shows that

**Proposition 6** The repeat patterns  $T_q$  are locally checkable, with  $r(T_q) \leq |q|/2$ . That is,  $LCR(T_q) \leq |q|/2$ .

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 §1.6. Suppose  $\Theta$  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 §1.6, 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

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

The local checkability criteria therefore subsumes the results of section 1.6.

### 2.1.1 Local Check Schemes as Parts List

Since a local check scheme  $\Theta$  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  $\Theta$  “accepts.” Let

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

be the set of full configurations accepted by  $\Theta$ , which we’ll call the  $\Theta$ -admissible configurations. As a set of configurations,  $\Theta(\mathcal{C})$  is by definition a pattern. If  $T$  is any pattern and  $\Theta$  is a local check scheme for it, then by definition  $\Theta(\mathcal{C}) \subset T$  and  $B_r(X) \subset \Theta^{-1}(1)$ . Hence,  $\Theta^{-1}(1)$  is like a list of valid local parts, and  $\Theta(\mathcal{C})$  is the set of one-dimensional structures that can be built from putting those parts together.

**Example 3** For example, suppose  $\Theta$  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  $\Theta(\mathcal{C}) = \{(100)^n(1000)^m \mid n, m \in \mathbb{N}\}$ .



Some parts  $z \in \Theta^{-1}(1)$  may be useable with more than one other part. Define  $R(b)$  as the set of parts that are consistent with have  $b$  to their left, i.e. (roughly) the set of states  $i$  such that the  $r$ -ball  $b[2 : 2r + 1] \circ i$  satisfies  $\Theta$ .  $|R(b)|$  is the *right compatibility index* of  $b$ . Analogously, define  $L[b]$  as the set of parts consistent with having  $b$  to their left, i.e. (essentially) those  $i$  such that the  $r$ -ball  $i \circ b[1 : 2r]$  satisfies  $\Theta$ .  $|L(b)|$  is the *left compatibility index* of  $b$ .

**Example 4** In the  $\Theta$  from example 3 above: the part  $b = 1^*000$  has  $|L(b)| = 0$  and  $|R(b)| = 1$ ; the part  $b = 0100^*$  has  $|L(b)| = 1$  and  $|R(b)| = 0$ ; the parts  $b = 1001^*001$  and  $b = 0001^*000$  have  $|L(b)| = |R(b)| = 1$ ; and  $b = 0100^*100$  has  $|L(b)| = 1$  and  $|R(b)| = 2$ .

## 2.2 Local Checkability is Sufficient: A General Construction

Proposition 5 says that local checkability is a necessary condition for robust static solvability, i.e. only for patterns  $T$  that have local check schemes  $\Theta$  can there be finite-radius rules  $f$  that can solve  $T$  on all initial conditions. Here, we will prove the converse to proposition 5.

That is, we will make a generic construction that associates to each local check scheme  $\Theta$  a local rule  $F_\Theta$  with finite radius that is a solution to the pattern  $T = \Theta(\mathcal{C})$ , the pattern generated by  $\Theta$ . As a local rule – an agent-based action lookup table –  $F_\Theta$  will have to take as input local balls  $b$  with some finite radius, and output a state  $s$  to which the agent will change its state when its local view is  $b$ . When iterated under any live call sequence (i.e one that always eventually calls every agent again), the configuration will have to converge in finite time to an element of  $\Theta(\mathcal{C})$ . As long as it is finite, the radius of the local rule  $F_\Theta$  need not be the same as the radius  $r(\Theta)$ , but by virtue of prop. 5, of course  $r(F_\Theta) \geq r(\Theta)$ . In the construction we make, the local rule  $F_\Theta$  will have radius  $2r(\Theta) + 1$ , that is, about twice the radius of the local check scheme  $\Theta$ .

Intuitively, what our construction does is produce a brute-force “lexicographic” search through the set of possible locally correct structures until one is found that is correct everywhere. I describe the construction of  $F_\Theta$  in two overall steps, broken down into 10 constituent rules. In the following construction, we will suppose for the sake of ease that we have several extra states to work with, states which do not appear in the pattern  $\Theta(\mathcal{C})$ . Later on in section 5.2, we’ll describe how to remove the extra states. We’ll end up using no more than  $m + 2$  extra states, where  $m$  is the number of original states.

### 2.2.1 Step 1: Propagating Correct Structure

Step 1 initiates and propagates a “wave of structure creation.” To do this, we will use one additional state, which we’ll denote  $\triangleright$ . Wave structure propagation is achieved by four rules.

- **Rule 1:** Suppose an agent considers the  $2r(\Theta) + 1$  agents to its left and sees that when considered as an  $r$ -ball, their states satisfy  $\Theta$  – that is,  $\Theta[B(-2r - 1 : -1)]$  holds true, where  $B$  is a local ball around  $i$  of radius at least  $2r + 1$ . Moreover, suppose that when it includes consideration of its own state, the  $r$ -ball  $B(-2r : 0)$  does NOT satisfy  $\Theta$ . In words, the agent is on a “border of correctness.” This situation applies to the agent targeted by the arrow in figure 4A, step 1.

When this situation holds, the agent sets its state to  $\triangleright$ .

- **Rule 2:** Now suppose the agent has state  $\triangleright$ , and it sees when it looks to its right that so does the agent to its right; that is,  $B(0) = \triangleright$  and  $B(1) = \triangleright$ . Suppose also that  $B[i - 2r - 1 : -1]$ , considered as an  $r$ -ball, satisfies  $\Theta$ . In this case, the agent finds the minimum state  $j \in \{0, \dots, m - 1\}$  for which  $X[i - 2r : i - 1] \circ j$ , considered as an  $r$ -ball, satisfies  $\Theta$ , and sets  $F(b) = j$  (see fig. 4A, steps 3 and 5). This assumes there is at least one such  $j$ , and we’ll get to what happens when there isn’t one later on in **Rule 5**.
- **Rule 3:** Suppose that an agent looks to  $2r + 2$  agents to its left, and sees that the agent directly on its left has state  $\triangleright$  (i.e.  $B(-1) = \triangleright$ ), while the agents  $B[-2r - 2 : -2]$ , considered as an  $r$ -ball, satisfy  $\Theta$  (see fig. 4A, steps 2, 4 and 6). In this case, we set  $F(B) = \triangleright$ .
- **Rule 4:** Finally, suppose the agent is at the right end of configuration, i.e.  $i = |X|$ . Suppose the agent finds itself in state  $\triangleright$ , and, like in **Rule 2**, sees that  $B[-2r - 1 : -1]$  satisfies  $\Theta$ . In this case, the agent finds the minimum state  $j \in \{0, \dots, m - 1\}$  such that  $\Theta$  holds for all  $B[-2r + j : 0]$ , considered as right-end  $2r + 1$  balls, where  $j$  runs from 0 to  $r$ . and sets  $F(B) = j$  (fig. 4A step 7).

The effect of these four rules is very simple: to initiate and propagate forward a wave of  $\Theta$ -correctness, and if possible, when it gets to the end, complete it. Thought of in traditional computing terms, the  $\triangleright$ -state is a “self-organized Turing machine head”; a head appears (as per rule 1) when a state is determined to have a local error; the head moves across the

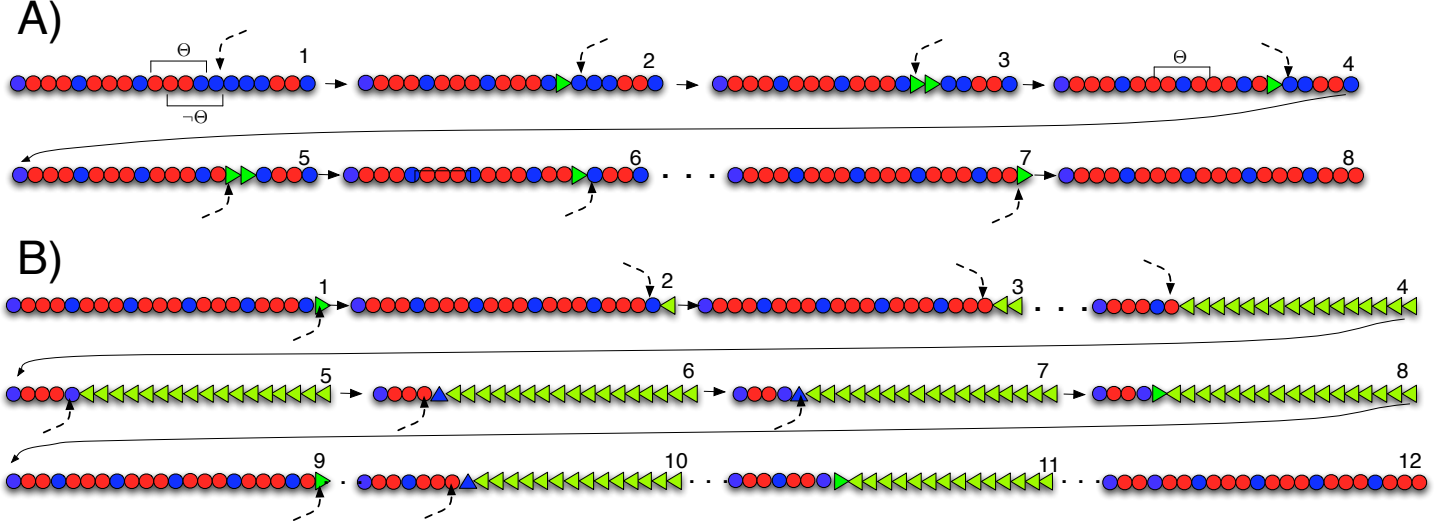


Figure 4: The action of the local rule  $F_\Theta$  on two initial conditions. (Agent taking the actions are indicated by arrows.) A) From step 1 to 2, a right-moving wavefront emerges at a local error. In steps 2-7 it propagates to the right end of the system, leaving behind a trail of  $\Theta$ -correctness. In step 8, the correctness wave disappears off the right end, leaving a complete correct configuration. B) In step 1, the correctness wave cannot resolve a consistent structure at the right end, it reverses direction. In steps 2-5 it propagates to the first position where a “next choice” can be made, whereupon in steps 6-8 it updates the choice reverses, and propagates (multiple steps) according to the new choice. Upon reaching the end (step 9) again without being able to resolve, the process repeats (10 and 11, representing multiple steps) until an acceptable series of choices is found.

system in a wave (as per rules 2 and 3), leaving a trail of local correctness behind it; and when possible, it halts (as per rule 4), disappearing off the boundary of the system. At any given time, multiple such Turing heads can appear; the left-most head will eventually overwrite all the others.

### 2.2.2 Step 2: Recovering from Error

There are two ways in which the wave above can fail to solve the problem: the turing head  $\triangleright$  gets to the end without finding a state  $i$  satisfying the requirements in rule 4 above; or it fails to find a state  $i$  from which it can continue somewhere in the middle of the space, as in rule 2. The basic way that the system recovers from such a situation is to turn around the way, backtrack in the opposite direction until it is possible to make different choice other than the simple first choice made in rule 3 above, implement the new choice, reverse direction, and then try again.

We'll show how to implement this idea in six rules. Again, in each step let  $B$  represent the  $2r + 2$ -radius ball around an agent. It turns that we'll need  $m + 1$  additional extra states, which we'll call  $\triangleleft$  and  $\triangle_1, \triangle_2, \dots, \triangle_m$ .

- **Rule 5:** Suppose agent  $i$  is in the situation of **Rule 2**, that is,  $i < |X|$  and  $\Theta[B[-2r - 1 : -1]]$ . But suppose there is no  $j$  for which  $\Theta[B[-2r : -1] \circ j]$ . Similarly, the agent is in the situation of **Rule 4**, i.e.  $i = |X|$  and  $\Theta[B[-2r - 1 : -1]]$ ; but there is no  $j$  such that  $B[-2r + l : -1] \circ j$  for all  $0 \leq l \leq r$  (fig. 4B steps 1 and 9). In this case, we introduce a new additional state, call it  $\triangleleft$ , and let  $F(B) = \triangleleft$ .
- **Rule 6:** Suppose that the agent to the right of agent  $i$  is in state  $\triangleleft$ , i.e.  $B(1) = \triangleleft$ ; that the  $r$ -ball to the left of agent  $i$  satisfies  $\Theta$ , i.e.  $\Theta[B[-2r - 1 : -1]]$ ; and the  $r$ -ball left of, but including  $i$ , also satisfies  $\Theta$ , i.e.  $\Theta[B[-2r : 0]]$ . Now, suppose further the state of the agent  $i$  is the largest  $j \in \{0, \dots, m - 1\}$  such that  $\Theta[B[-2r : 0]]$  will hold. Such a situation holds in figure 4B, steps 2, 3, and 4. Whenever this situation arises holds, again let  $F(B) = \triangleleft$ .
- **Rule 7:** Suppose that the agent  $i$  is in state  $\triangleleft$ , i.e.  $B(0) = \triangleleft$ , and that  $\Theta[B[-2r - 1 : -1]]$  and  $\Theta[B[-2r - 2 : -2]]$ , i.e. when the agent looks to its left  $2r - 2$  agents down, it sees that  $\Theta$  is satisfied. Now suppose further that the state of the agent to the left of  $i$ , i.e.  $B(-1)$ , is NOT the maximum state for which this holds i.e. there is a state  $j > B(-1)$  such that  $B[-2r - 1 : -2] \circ j$  satisfies  $\Theta$  (see figure 4B, step 5). Whenever this situation holds, we resort to one of the  $m$  extra states  $\triangle_1, \triangle_2, \dots, \triangle_m$ , defining  $F(B) = \triangle_{B(-1)}$ .

- **Rule 8:** Suppose that the agent  $i$  sees that  $\Theta$  is satisfied to its left, so that  $\Theta[B[-2r : 0]]$ , and suppose that  $B(1) = \triangle_{B(0)}$ . Furthermore, suppose that  $B(0)$  is NOT the maximum for which this is the case, i.e. there is  $j > B(0)$  such that  $\Theta[B[-2r : -1] \circ j]$ . Then we define  $F(B) = j(B)$ , the smallest such  $j$  (see fig. 4B, steps 6 and 10).
- **Rule 9:** Suppose that the agent  $i$  is in state  $\triangle_j$  for some  $j$ , and either  $j \neq B(-1)$  or  $j$  is the maximum state such that  $\Theta[B[-2r - 1 : -2] \circ j]$  holds (fig. 4B, step 7). In this case,  $F(B) = \triangleright$ .
- **Rule 10:** Suppose the agent  $i$  is the left-most agent, i.e.  $i = 1$ , and  $B(0) = \triangleleft$ . Then  $F(B) = \triangleright$ .

The effect of these rules is very simple: when a mistaken choice has been made, leading to the inability to complete the configuration, the structure propagation wave created in the first step is reversed, and propagates until a point where the first chance to try a can be made; it then makes the change, reverses direction again, and propagates to the end; upon reaching the end, either a completion is possible in which case the completion is made, or no completion is made, in which case the wave reverses again and repeats the next-option, &c.

In terms of the Turing-head analogy, the  $\triangleleft$  state represents the turing head moving in the opposite direction, and the  $\triangle_i$  states some internal memory of the head that allow it to correctly re-write and know when to reverse direction to the right-moving  $\triangleright$ -state.

### 2.2.3 Putting It Together

Define the algorithm  $F_\Theta$  by:

---

**Algorithm 1:** The Naive Generic One-Dimensional Local Rule

---

```

if Rule 1 or Rule 3 or Rule 9 or Rule 10 applies then
  |  $F(B) = \triangleright$ 
else if Rule 5 or Rule 6 applies then
  |  $F(B) = \triangleleft$ 
else if Rule 2 applies then
  |  $F(B) = \min\{j \in [0, m-1] \mid \Theta[B[-2r : -1] \circ j]\}$ 
else if Rule 4 applies then
  |  $F(B) = \min\{j \in [0, m-1] \mid b_j[B]\}$ 
else if Rule 8 applies then
  |  $F(B) = \min\{j > B(0) \mid \Theta[B[-2r : -1] \circ j]\}$ 
else if Rule 7 applies then
  |  $F(B) = \triangle_{B(-1)}$ 
else
  |  $F(B) = B(0)$ .
end

```

---

A formal description of this algorithm is given in appendix A. We will describe in section 5.2 how to remove the requirement for extra states, at the cost of adding another  $4r(\Theta) + 2$  to the radius of information. The result is that:

**Theorem 1** *A pattern  $T$  is locally solvable if and only if the local check radius  $LCR(T)$  is finite, i.e. local checkability is a necessary and sufficient condition for solvability. Moreover, every solvable pattern  $T$  has a solution  $F$  whose radius of information is at most  $6 \cdot LCR(T) + 3$ .*

The lexicographic search self-organized by  $F_\Theta$  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.

**Definition 9 [Choice Ordering Function]** *A choice ordering function for  $\Theta$  is a mapping*

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

*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  $b$ , the ordering in by which different options for completing a structure from  $b$  will be taken:  $O(b, 1)$  is the first choice,  $O(b, 2)$  the second choice, etc... and  $O(b, |R(b)|)$ , the last choice. For each choice function  $O$ , we can define a corresponding  $F_\Theta^O$  by replacing all references to the numerical order with appropriate references to  $O$  (a formal specification is given in appendix 1). The rule  $F_\Theta^O$  is a solution to  $\Theta$  for any ordering function  $O$ .

### 3 Graph-Based Analysis

By closer inspection of the concept of local check schemes, we can produce deeper analysis of solvable patterns and compute run-time performance of the naive algorithm. We can also produce more sophisticated faster algorithms.

#### 3.1 Local Check Schemes as Graphs

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  $X \in \mathcal{C}_S$  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,

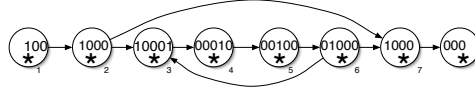
$$\mathbb{D}(r, m) = (\mathcal{B}_r, \{(b_1, b_2) \in \mathcal{B}_r^2 \mid b_1(1 + \max(0, \star(b) - r) : |b_1|) = b_2(1 : |b_2| - \max(0, \star(b_2) - r))\}).$$

A local check scheme  $\Theta$  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  $\Theta$ . On the other hand,  $\Theta^{-1}(1) \subset \mathcal{B}_r$ , so now simply assign to  $\Theta$  the induced subgraph of  $\mathbb{D}(r, m)$  whose nodes are  $\Theta^{-1}(1)$ . We thus have:

**Proposition 8** *Local check schemes of radius  $r$  are in 1-1 correspondence with subgraphs of  $\mathbb{D}(r, m)$ , via the assignment*

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

**Example 5** Take the repeat pattern  $T_{1000}$  and the local check for it described in example 2. The associated graph  $G(\Theta)$  is:



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 instance 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  $4(n + 1)$  instance of the pattern.

#### 3.2 Detailed Analysis

The graph representation allows us to classify the various kinds of locally checkable patterns, as well as perform a detailed analysis the run-time performance of the naive glocal compiler. We will describe a few illustrative examples of increasing complexity, and the general case.

##### 3.2.1 Line Graph

The simplest case is when  $G(\Theta)$  is a line graph:



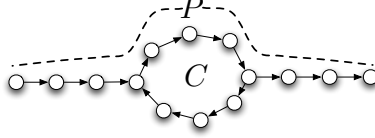
All nodes in  $G(\Theta)$  have indegree and outdegree 1, except the unique initial node  $N_i$  (with indegree 0) and the unique terminal node  $N_t$  (with outdegree 0). The pattern generated by  $\mathcal{C}(\Theta)$  consists of a single configuration  $X$ , corresponding to the unique maximum-length path through  $G(\Theta)$ . Of course, the size of  $X$  is  $|X| = |G(\Theta)|$ , the number of nodes in  $G(\Theta)$ . Each agent has one and only one possible correct local view;  $G(\Theta)$  being a line-graph corresponds to the situation of a “self-organized coordinate gradient.” (More on this in section 5.1.)

Since there is only one solving configuration  $X$ , only initial configurations of size  $|X|$  can be solved at all. Whenever  $|X_0| = |X|$ , the rule  $F_\Theta$  will on average produce the solution on  $X_0$  in about  $2 \cdot |X_0|$  timesteps. To see why this is, suppose

that a configuration of the correct size has its left-most error at position  $i$ . Shortly therefore, a local turing head  $\triangleright$ -state will appear at the position  $i$  as per rule 1 in Step 1; It will travel to the right at the constant rate of one agent per two timesteps as per rules 2 and 3; when  $\triangleright$  reaches the right end, given that  $X_0$  has the right size, the  $\triangleright$  will disappear, leaving a correct finishing state, as per rule 4. The solution has appeared in about  $2(|X_0| - i)$  steps, and averaged over all configuration,  $i$  will be distributed uniformly. If  $|X_0| \neq |X|$ , the trajectory of  $X_0$  under  $F_\Theta$  will be quasi-periodic (depending on the aperiodicity of the call sequence  $s$ ), with period at least  $4|X_0|$ .

### 3.2.2 Single Cycle Graph

The next simplest case is that when  $G(\Theta)$  contains one cycle and maximal acyclic path:

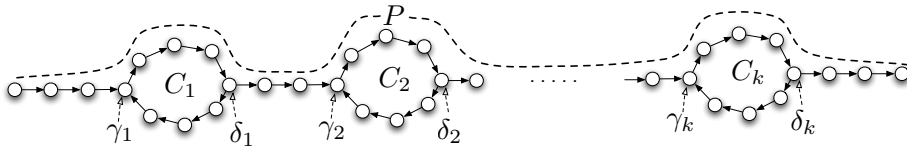


Every repeat pattern corresponds to a graph of this kind. Denote the cycle  $C$ , and the maximal acyclic path  $P$ . In addition to the unique initial and terminal nodes, there is a unique node  $\alpha$  with  $InDeg(\alpha, G(\Theta)) = 1$  and  $OutDeg(\alpha, G(\Theta)) = 2$ ; and unique  $\beta$  with the reverse. Otherwise, all nodes have indegree and outdegree 1. The pattern generated by such a  $\Theta$ ,  $\mathcal{C}(\Theta)$ , consists of infinitely many configurations  $X_n$ , where  $|X_n| = |P| + n|C|$ . There is only one configuration  $X_n$  at each size of this form, corresponding to the path which repeats the cycle  $C$   $n$  times. Only configurations larger than  $|P|$  and of length  $n|C| + \text{mod}(|P|, |C|)$  can be solved.

Whenever  $|X_0|$  satisfies this condition, the rule  $F_\Theta$  will typically produce the solution on  $X_0$  in about  $2(|X_0| - i + C(r))$  timesteps (on expectation), where  $C(r)$  is a constant independent of  $|X_0|$ , no larger than  $m^{2r}(m^{2r} + 1)$ . To see why this is, suppose wlog that the state that in  $R(\beta)$ , the state that continues the cycle is less than the state that moves on the rest of  $P$ . If a configuration of the correct size has its left-most error at position  $i$ , a local turing head  $\triangleright$ -state will appear at the position  $i$  as per rule 1 in Step 1; it will travel to the right at the constant rate of one agent per two timesteps as per rules 2 and 3, leaving behind copies of  $C$  until it reaches the end after  $2|X_0|$  timesteps. At this point, the head will reverse and back up to the last instance of  $\beta$ , reverse again, and choose to write  $P$  instead. The same will repeat until the proper number of copies of  $C$  have been overwritten to the right by the remainder of  $P$ ; this portion of  $P$  can be at most  $m^{2r}$  in length, and the amount of time it takes is therefore on expectation  $\sum_{i=0}^{m^{2r}} 2i = m^{2r}(m^{2r} + 1)$ . Moreover, the average case time will scale with  $|X_0|$ , since  $|X_0| - i$  will be roughly uniformly distributed between 1 and  $|X_0|$ .

### 3.2.3 Multi-Cycle Linear Graph

Now suppose  $G(\Theta)$  has several simple cycles in a row:



Let  $P$  denote the unique maximal acyclic path in this graph. Let  $C_i$  denote the cycles as shown in the figure. All nodes in  $G(\Theta)$  have indegree 1 except for the  $k$  nodes  $\gamma_i$ , where the indegree is 2, one incident edge coming from the cycle  $C_i$  and one from  $P$ . Likewise, all nodes have outdegree 1 except for the  $k$  nodes  $\delta_i$ , where the outdegree is 2, once edge going to  $P$  and one edge remaining in  $C_i$ .

The pattern generated by such a  $\Theta$ ,  $\Theta(\mathcal{C})$ , consists of the configurations

$$\{X_{n_1, \dots, n_k} | n_i \in \mathbb{N}\}$$

in which  $X_{n_1, \dots, n_k}$  corresponds to the path through  $G(\Theta)$  containing  $n_i$  repeats of the unit  $C_i$  for each  $i$ . It has size

$$|X_{n_1, \dots, n_k}| = |P| + \sum_{i=1}^k n_i |C_i|. \quad (1)$$

The  $k$  cycles in  $G(\Theta)$  correspond to  $k$  “irreducible modules” that can be combined in varying multiplicities to produce structures of many sizes. The fact that the cycle  $C_i$  is connected to  $C_j$  for  $j > i$ , but not vice-versa, means however that the modules may only be combined in a specific ordering.

Applying the Chinese Remainder Theorem, the set of sizes of the form in expression 1 consists a finite set of integers less than  $|P| + lcm(|C_1|, \dots, |C_k|)$ , together with all integers larger than  $|P| + lcm(|C_1|, \dots, |C_k|)$  of the form  $n \cdot gcd(|C_1|, \dots, |C_k|) + mod(|P|, |C_1|, \dots, |C_k|)$ . At each such size  $n$  for which there are configurations in  $\Theta(\mathcal{C})$  of size  $n$ , the number of possible configurations scales as  $n^{k-1}$ ; that is, there are polynomially many possible correct configurations, where the polynomial exponent is the number of cycles in  $G(\Theta)$ , less 1.

**Example 6** Suppose that  $k = 3$ ,  $|P| = 37$ ,  $|C_1| = 6$ ,  $|C_2| = 10$ , and  $|C_3| = 15$ . Then  $lcm(|C_1|, |C_2|, |C_3|) = 30$  and  $gcd(|C_1|, |C_2|, |C_3|) = 1$ , so  $F_\Theta$  must be able to construct configurations of all sizes greater than  $37 + 30 = 67$  (as well as some smaller ones). For each  $n > 67$ , there are asymptotically  $\sim n^{3-1} = n^2$  solving configurations. To construct even-sized configurations it may be possible only to use instances of  $C_1$  and  $C_2$ ; to construct odd-sized configurations, at least one instance of  $C_3$  must be used.

A solution  $F$  to  $G(\Theta)$  must ensure that for configurations of every admissible size (such sizes are integers of the form in expression 1),  $F$  drives such configuration to a solved state. Unlike in the case of the previous section, however, there are many different possible solved states at each admissible size, and a variety of different unit combinations (taken with various multiplicity) will have to be used to achieve all the solved states. Hence, the problem of solving  $\Theta$  when  $G(\Theta)$  is a multiple-cycle graph is “resource-limited pattern choice,” which involves not just forming the pattern, but also the problem of “distributedly” figuring out which pattern to form as a function of size constraints.

Now, let  $j$  be the minimal  $l$  for which  $gcd(|C_l|, \dots, |C_k|) > gcd(|C_1|, \dots, |C_k|)$ . (For instance, in example 6,  $j = 2$ .) Let  $i$  be any node in  $C_j$  or the portion of  $P$  between  $C_{j-1}$  and  $C_j$ . Let  $Y$  be any configuration such that  $|Y| + i + 1$  is of the form  $|P| + gcd(|C_1|, \dots, |C_k|) \cdot n$  for some  $n$ , but NOT of the form  $|P| + gcd(|C_j|, \dots, |C_k|) \cdot m$  for any  $m$ . There is an infinite arithmetic progression of such  $Y$ 's. Now, consider the initial condition  $X_0$ :

$$X_0 = P[1 : i] \circ \triangleright \circ Y.$$

In any live uniform timing model, the algorithm  $F_\Theta$  will require on the order of  $n^{max(1, k-j+1)}$  timesteps to solve  $X_0$ .

The reason this is so is that  $F_\Theta$  will have to “try out” all possible configurations that can complete  $P[1 : i]$  before moving on to using at least one copy of the cycle  $C_{j-1}$ . No number that is not 0 modulo  $gcd(|C_j|, \dots, |C_k|)$  can be written as sum of multiples of  $|C_j|, \dots, |C_k|$ , so none of the configurations not containing an instance of  $C_{j-1}$  can work. Yet they all are explored. This is because the “self-organized turing head” represented by the coherent moving  $\triangleright \circ \triangleright$  and  $\triangleleft \circ \triangleleft$  states, makes many reversals. Rules 7 and 8 are applied, forcing a head reversal, every time the  $\triangleleft \circ \triangleleft$  head encounters a new possible decision point. All these head reversals are *futile*. Now, there are  $k - j + 1$  cycles from  $C_j$  to  $C_k$ , so the number of completions of  $X_0[1 : i]$  configurations scales with  $|X_0|$  as  $(|X_0| - i)^{k-j+1}$ . Thus, there is a constant  $C(r)$  such that the run-time of  $F$  on  $X_0$  will be at least  $C(r) \cdot (|X_0| - i) + 2(|X_0| - i)^{k-j+1}$ ; it can also be shown that the run-time is bounded in all cases by  $D(r) \cdot (|X_0| - i) + 2(|X_0| - i)^{k-j+1}$  for another constant. The average and worst-case run-time (in live, uniform timing models) of  $F_\Theta$  is therefore asymptotically  $O(n^L)$  for some integer  $L$  such that  $1 \leq L \leq k - 1$ .

### 3.2.4 One Non-Trivial SCC

Given a directed graph  $G$  and two nodes  $x, y \in G$ ,  $x \in G$  is connected to  $y \in G$  if there is a path  $P = (P_1, \dots, P_k)$  in  $G$  such that  $P_1 = x$  and  $P_k = y$ . A *strongly connected component* (SCC) of  $G$  is a maximal subgraph  $S \subset G$  such that for all  $x, y \in S$ ,  $x$  is connected to  $y$ , AND vice versa. An SCC is *trivial* if it contains a single node. The strongly connected components of a graph form a partition of  $G$ , and non-trivial SCCs consist of unions of connected cycles. The *adjacency matrix*  $A(G)$  is associated to  $G$  by letting  $A_{ij}$  equal 1 if node  $i$  is connected to node  $j$ , and 0 zero otherwise.  $A(G)$  is called the *adjacency matrix* of  $G$ . Like any other matrix,  $A(G)$  has eigenvalues, that are *a priori* complex numbers. When  $G$  consists of a single strongly connected component, the largest-norm eigenvalue of  $A(G)$ , denoted  $\lambda_{max}(G)$ , is a real number no less than 1, with equality iff  $G$  contains a single cycle.

Suppose  $G(\Theta)$  has a single nontrivial strongly connected component consisting of several cycles:



Denote the strongly connected component  $\mathcal{C}$ . This graph may contain multiple maximal acyclic paths. The indegrees and out-degrees of the nodes can range between 1 and  $m$ , where  $m = |S|$ . Let  $A(\Theta)$  be the set of all maximal acyclic paths in  $G(\Theta)$  and let  $C_1, \dots, C_k$  enumerate the induced cycles in  $\mathcal{C}$ . Each of these induced cycles is like an “irreducible module” of

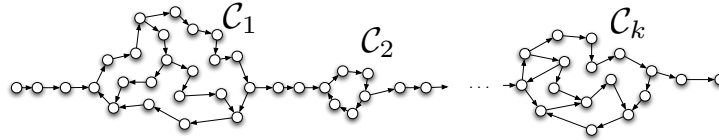
the pattern, and the overall pattern generated by such a  $G(\Theta)$  consists combining these modules freely in arbitrary orders and multiplicities.

The sizes of configurations that can thereby be obtained includes a finite set smaller than  $lcm(|C_1|, \dots, |C_k|)$ , together with the union over  $P \in A(\Theta)$  of all integers of the form  $gcd(|C_1|, \dots, |C_k|) \cdot n + mod(|P|, |C_1|, \dots, |C_k|)$  larger than  $|P| + lcm(|C_1|, \dots, |C_k|)$ . Using standard techniques, it can be shown that at each such size  $n$  for which there are configurations in  $\Theta(\mathcal{C})$  of size  $n$ , the number of possible configurations scales as  $(\lambda_{max}(\mathcal{C}))^n$ ; that is, there are exponentially many possible correct configurations, where the exponential growth rate is the top eigenvalue of the adjacency matrix of the connected component  $\mathcal{C}$ .

The problem of solving this type of  $\Theta$  is, like the previous case, a “resource-limited pattern choice,” problem. However, if  $X_0$  is of the proper size, for reasons similar to the arguments in the first two cases, the average and worst-case runtime scales like  $C(r) \cdot |X_0|$  for some constant  $C(r)$ .

### 3.2.5 Multi-SCC Linear Graphs

Now suppose  $G(\Theta)$  has several nontrivial strongly connected components in a linear arrangement, combining and generalizing the structure of the past several sections:



If each of the SCCs contains a single cycle, we’re in the case of §3.2.3. Let  $\mathcal{C}_i$  denote the  $i$ -th nontrivial strongly connected component. Let  $\lambda_\Theta = \max_i \{\lambda_{max}(\mathcal{C}_i)\}$ . Let  $g_i = g(\mathcal{C}_i)$ , the greatest common divisor of the lengths of the induced cycles in  $\mathcal{C}_i$ , and  $l_i$  their least common multiple. As in the previous example with a single component, the admissible sizes include a finite set smaller than  $lcm(l_1, \dots, l_k)$ , together with the union over  $P \in A(\Theta)$  of all integers of the form  $n \cdot gcd(g_1, \dots, g_k) + mod(|P|, g_1, \dots, g_k)$  larger than  $|P| + lcm(l_1, \dots, l_k)$ .

The run-time analysis of the algorithm  $F_\Theta$  on such a pattern is slightly subtle. Let  $j$  be the minimal  $l \leq k$  for which

$$\bigcup_{P \in A(G(\Theta))} \{gcd(g_l, \dots, g_k)n + mod(|P|, gcd(g_l, \dots, g_k)) | n \in \mathbb{N}\}$$

is a proper subset of

$$\bigcup_{P \in A(G(\Theta))} \{gcd(g_1, \dots, g_k)n + mod(|P|, gcd(g_1, \dots, g_k)) | n \in \mathbb{N}\}.$$

Let  $P$  be a maximal acyclic path and  $X_0$  an initial condition such that  $|X_0| - |P|$  is of size divisible by  $gcd(|C_1|, \dots, |C_k|)$  but not  $gcd(|C_j|, \dots, |C_k|)$ ; suppose moreover that the left-most error in  $X_0$  is at position  $i$ , and that  $X_0[1 : i] = P[1 : i]$ . Then the head that appears near  $i$  will have to cycle through all possible completions of  $X_0[1 : i]$  before being able to come to a correct solution. There are now two cases. The number of such configurations can scale: (A) polynomially, if none of the  $\mathcal{C}_l$  for  $l \geq j$  contain multi-cycle SCCs; or (B), exponentially, if at least one SCC  $\mathcal{C}_l$  for  $l \geq j$  contains two cycles. Hence, the average and worst-case runtime of  $F_\Theta$  will scale either as  $O(n^L)$  for integer  $L$  such that  $1 \leq L \leq j$ ; or as  $O(\lambda^n)$  where  $\lambda$  is a real number such that  $1 \leq L \leq \lambda_\Theta$ .

### 3.3 The General Case

Let  $\mathbf{G} = \{S_i\}$  be the set of strongly connected components of  $G$ . 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  $\mathcal{G}$ . In terms of  $G(\Theta)$ , the general picture is this:

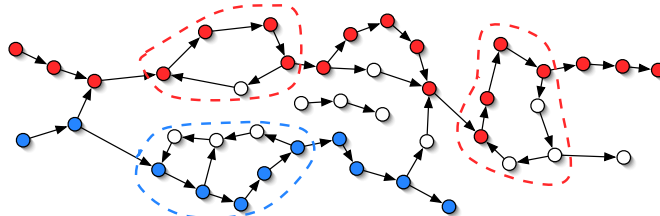


Figure 5: Maximal acyclic paths  $P_{red}$  and  $P_{blue}$  with  $SCC(P_{red})$  and  $SCC(P_{blue})$  outlined.

in which the SCCs correspond to islands of configurations composed of alternatable irreducible repeatable modules bridged by unrepeatable configurations; and every SCC is in one or more WCC, each of which corresponds to a linear ordering of non-alternatable blocks.

The general form of  $G(\Theta)$ , therefore corresponds to the union of several (possible non-disjoint) WCCs, each of which is of the form described in §3.2.5, and the results respect this structure. For any maximal acyclic path  $P \in A(G(\Theta))$ , define  $SCC(P) = \bigcup_{x \in P} SCC(x, G)$ , the set of all strongly connected components that  $P$  passes through. Let  $InCyc(P)$  be the set of induced cycles in all these SCCs. Let  $E(P) = \{|c|, c \in InCyc(SCC(P))\}$ , the set of sizes of induced cycles of  $SCC(P)$ . For the red path,  $E(P_{red}) = \{5, 7\}$ , 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,  $L(P_{red}) = 35$  and  $T(P_{red}) = 1$ , while  $L(P_{blue}) = 8$ , and  $T(P_{blue}) = 4$ .

The arithmetic progression  $A(a_0, d)$  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 9** *Let  $K(P) = \max_{P \in A(G)} L(P)$ . Then*

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

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

**Example 7** In the example in figure 5, the maximal acyclic path highlighted in red is of length 19 and intersects with two strongly connected components, each with a single cycle, of size 5 and 7 respectively, and hence contributes the arithmetic series  $A(19, 1)$ . The blue maximal acyclic path is of length 11 and intersects with one strongly connected component containing two cycles of length 4 and 8, so contributes  $A(11, 4)$ .

Again, given a maximal acyclic path  $P$ , denote by  $N(P)$  the number of strongly connected components that  $p$  intersects. Let

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

Intuitively  $N(\Theta)$  measures the largest number of non-interchangeable modules that can be connected in  $\Theta$ . For the pattern in figure 5,  $N(P_{blue}) = 1$ ,  $N(P_{red}) = 2$ , and thus  $N(\Theta) = 2$ . Let

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

so that for the example in figure 5,  $\lambda_\Theta = 1.1148$ .

With these definitions at hand, we can state:

**Proposition 10** *There is a constant  $C(r)$  such that for all  $n$  of admissible size, if  $\Theta$  contains no alternatable segments,*

$$|\Theta(\mathcal{C}) \cap \mathcal{C}_n| \sim C(r)n^{N(\Theta)-1},$$

*while if  $\Theta$  contains two or more alternatable segments, then*

$$|\Theta(\mathcal{C}) \cap \mathcal{C}_n| \sim C(r)\lambda_\Theta^n.$$

In words, a lack of alternability leads to polynomial growth, while alternatable of modules generates exponential growth. The growth rates are dependent on the number and relationships between the modules. For the example in figure 5,  $|\Theta(\mathcal{C}) \cap \mathcal{C}_n| \sim C \cdot 1.1148^n$ .

In the polynomial (non-alternatable) case, the interpretation is easy: the polynomial growth order is 1 more than the maximum number of modules that can be connected. In the exponential case, the internal structure of the module interrelationships becomes more important than the number of indepdent 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  $\lambda_{max} = C^{1/L}$ .

The same analysis of run-time from the above cases generalizes to:



**Proposition 11** *For any live uniform synchrony model, there are positive constants  $K_1(\Theta), K_2(\Theta)$  such that either*

$$K_1(\Theta)n^{C(\Theta)} \leq TTS_S^{avg}(F_\Theta)(n) \leq TTS_S^{worst}(F_\Theta)(n) \leq K_2(\Theta)n^{C(\Theta)}$$

or

$$K_1(\Theta)\lambda'(\Theta)^n \leq TTS_S^{avg}(F_\Theta)(n) \leq TTS_S^{worst}(F_\Theta)(n) \leq K_2(\Theta)\lambda'(\Theta)^n$$

where in the former case  $C(\Theta)$  is a positive integer satisfying  $1 \leq C(\Theta) \leq \max 1, N(\Theta)$ , and in the latter  $\lambda(\Theta)$  is a real number greater than 1, satisfying  $1 \leq \lambda'(\Theta) \leq \lambda_\Theta$ .

In words, the average and worst-case run-time of  $F_\Theta$  are the same, are either polynomial or exponential, and whose asymptotic rates are completely determined by the graph structure of  $G(\Theta)$ .

## 4 Faster Algorithms

We can use the graph interpretation of local check schemes introduced in the previous section to construct local rules with much shorter run-times.

### 4.1 A Linear-Time Algorithm

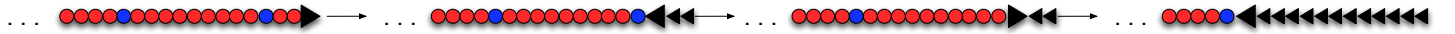
As we have just seen, the initial configurations that cause the slowest running time of the algorithm  $F_\Theta$  are those which start the system “past” a necessary choice point, and condemn it to make many futile backtrackings before accessing a module of the right size. It turns out that there is a simple modification of  $F_\Theta$  that will enable an agent at a decision point to decide, with a purely local computation, when a given reversal would be necessarily be futile, and therefore to choose not to reverse. The ability to avoid futile backtrackings reduces the runtime dramatically.

Suppose  $\Theta$  is a local check scheme of radius  $r$  such that  $G(\Theta)$  is a multi-cycle linear graph as in §3.2.3, with cycles  $C_i$  and maximal acyclic path  $P$ . We will make two modifications to the rules of  $F_\Theta$ , turning it from a polynomial time algorithm into a new algorithm  $F'_\Theta$  whose worst-case time is always linear in system size, independent of the number of cycles in  $G(\Theta)$ .

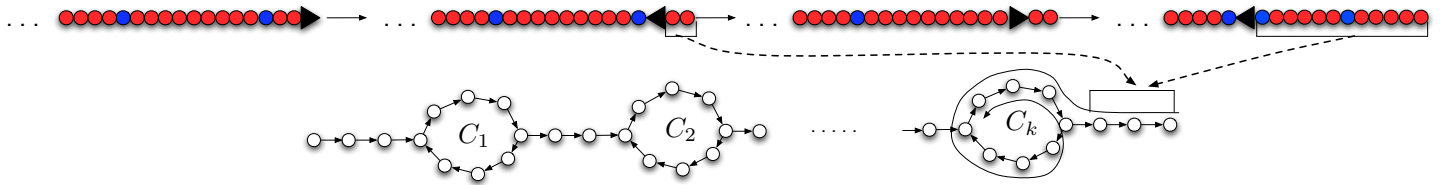
For the first modification: recall that as per **Rule 6** in  $F_\Theta$ , given an initial condition  $X_0$  of the form

$$X_0 = Y \circ \triangleleft,$$

where  $Y$  is  $\Theta$ -consistent, the left-moving turing head  $\triangleleft$  will move across the configuration, leaving a string of  $\triangleleft$  states behind it, until it arrives at a new decision point:

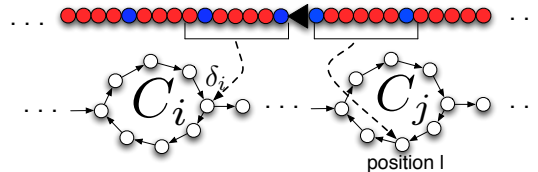


Modify **Rule 6** to a new **Rule 6'**, so that when the head  $\triangleleft \circ \triangleleft$  progresses to the left, instead of leaving behind a string of  $\triangleleft$ 's, it writes a string of states consistent with the local check scheme  $\Theta$  from the right-end:



As the  $\triangleleft$ -head moves, it leaves behind copies of  $C_k$  flanked on the right by a  $\Theta$ -consistent right end, i.e. subconfigurations of the form  $C_k^m \circ P_{\geq k}$  where  $P_{\geq j}$  refers to the portion of the maximal path after  $\delta_j$ .

We now make a second modification to the rules of  $F_\Theta$ . Suppose that an agent sees in its local radius  $2r + 1$  ball, the situation in which: i) its own state is that of the left moving turing head, i.e.  $B(0) = \triangleleft$ ; ii) to the left of the  $\triangleleft$ -head, that the ball  $B(-2r - 1 : -1)$  is  $\Theta$ -consistent, and at the decision point  $\delta_i$  in some cycle  $C_i$ ; while iii) to the right of  $\triangleleft$ , the agent sees a ball  $B(1 : 2r + 1)$  that is  $\Theta$ -consistent, in the cycle  $C_j$  with  $j \geq i$ , and is at some position  $l$  within that cycle:



In this case, by definition of **Rule 7** and **Rule 8** in  $F_\Theta$ , the system would reverse direction of the head to  $\triangleright$ , if a “next state” is available at  $B(-1)$ . This reversal might be futile. In fact, let’s assume that all the states of the agents down to the right end have been constructed by the left-moving head according to the modified rule 6. In this case, the agent knows that the size of the remainder of the configuration to its right is of the form

$$\gcd(|C_j|, \dots, |C_k|) \cdot n + l + |P_{\geq j}|$$

for some  $n$ . If the head reverses, it can only form a completion if there is some  $m$  for which

$$m \cdot \gcd(|C_{i+1}|, \dots, |C_k|) + |P_{\geq i}| = \gcd(|C_j|, \dots, |C_k|) \cdot n + l + |P_{\geq j}| + r + 2.$$

By the chinese remainder theorem this is in turn only possible if

$$A = l + |P_{\geq j}| + r + 2 - |P_{\geq i+1}|$$

is divisible by

$$B = \gcd(|C_{i+1}|, \dots, |C_k|).$$

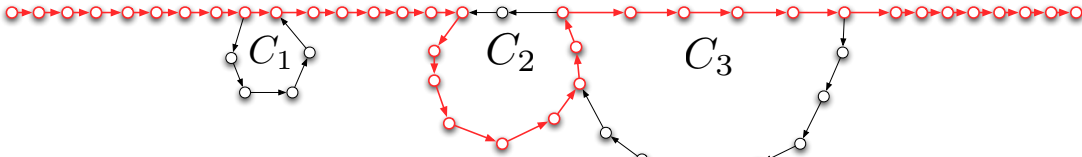
Moreover, if  $A$  is divisible by  $B$  and the size  $J$  is larger than  $L = \text{lcm}(|C_{i+1}|, \dots, |C_k|)$ , a completion will necessarily exist. Computing  $A$  and  $B$  is a purely local computation, since the value of  $l$ ,  $i$ , and  $j$ , can locally be determined by the agent from the  $2r + 1$  ball as illustrated in the last figure. Hence, let’s replace Rule 7 and Rule 8 with **Rule 7’** and Rule 8’ so that the head reverses *only if*  $A$  is divisible by  $B$ .




















































Denote the resulting local rule with these modifications to rules 6,7,8 by  $F'_\Theta$ . Because it bypasses futile reverses whenever  $|X| > L$  (and  $L$  is a constant depending only radius  $r$ ), the worst-case run-time of  $F'_\Theta$ , (on expectations over call sequences) is *linear* in  $|X_0|$ , regardless of the number of cycles in  $G(\Theta)$ .

To see how this works in practice, let’s work through an example. Suppose  $\Theta$  is the radius-8 local check scheme on 2 states generated by accepting all the 8-balls in the configurations

$$\{(100000)^n (1000000000)^m (1000000000000000)^o | m, n, o \geq 2\}.$$

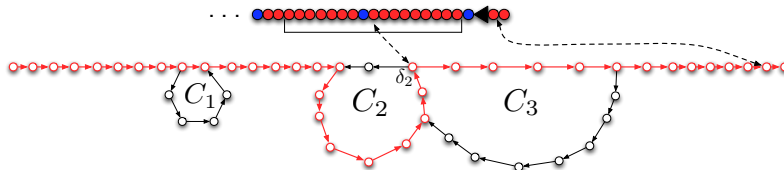
The graph structure of  $G(\Theta)$  is:



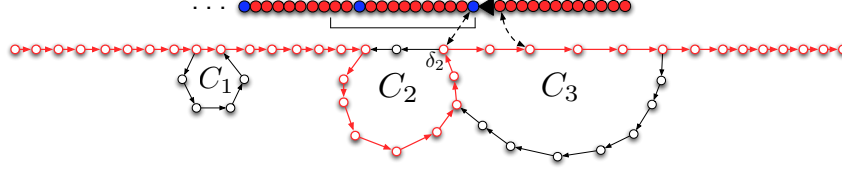
The graph has a single maximal acyclic path  $P$  (highlighted in red) of length 37 and three cycles, of length 6, 10, and 15 respectively. (This  $G(\Theta)$  has the structure of the check scheme mentioned in example 6.) The cycles  $C_1$ ,  $C_2$ , and  $C_3$  correspond to repeatable sequences 100000, 1000000000, and 1000000000000000, respectively. For the sake of convenience, let’s represent them by color sequences       ,           , and                                    , respectively.

Suppose we start from the initial condition  $X = (100000)^2 (1000000000)^n 100 \circ \triangleright$  for  $n > 6$ .  $X$  contains no instances of the  $C_1$  loop. It has size  $10n + 16$ , and since  $|X| - |P| = 10n - 21$  is congruent to 4 modulo  $\gcd(|C_2|, |C_3|) = 5$ , it cannot be solved unless an instance of  $C_1$  is used.

Because  $X$  is not locally completable to be consistent with  $\Theta$  at the right end, under **Rule 5** in  $F_\Theta$ , a left-moving turing head will form. Under the new **Rule 6’**, when it reaches the first decision point  $\delta_2$  at position  $10n + 12$ , the left-moving head will see a local situation like this:



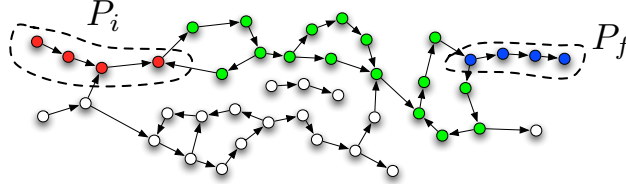
In the notation from above,  $i = 2$ ,  $j = 3$ , left ball  $B(-2r - 1 : -1)$  is at the decision point  $\delta_2$  in cycle  $C_2$  and the right ball  $B(1 : 2r + 1)$  is at position  $l = -6$  (in the right-terminal segment). In this case,  $A = -6 + 8 + 8 + 2 - 8 = 4$  and  $B = |C_3| = 15$ ;  $A$  is not divisible by  $B$  so the head does not reverse. At the next decision point, the agent carrying the  $\triangleleft$  head sees:



In this case, again  $i = 2$  and  $j = 3$ . Now  $l = 4$ , so  $A = 4 + 8 + 8 + 2 - 8 = 14$  and  $B = |C_3| = 15$ ; again, no reversal. At the next decision point, again  $i = 2$ ,  $j = 3$ , and now  $l = 14$ , so again no reversal. At the  $i$ -th decision point,  $l$  is such that  $10i + 12 = 15m + 8 + l$ , so that either  $l \equiv 4(15)$  or  $l \equiv 14(15)$ . Thus, no reversal takes place before the  $\triangleleft$  reaches the left end of the configuration. The configuration is now  $X' = \triangleleft \circ Y$ , so under **Rule 10**, this first becomes  $\triangleright \circ Y$ . Then under **Rule 2** and **Rule 3**, the head reverses and travels all the way to the right, eventually generating the configuration  $C_1^{m_1} \circ C_1(1 : r) \circ \triangleright$ , where  $C_1 = 100000$ ,  $m_1 = \lfloor (10n + 16)/6 \rfloor$  and  $r = \text{mod}(10n + 16, 6) - 1$ . Now, there is a solving configuration  $X' = (10000)^{m'_1} \circ (10000000000)^{m_2} \circ (10000000000000000)^{m_3}$  where  $m'_1 \geq \text{lcm}(|C_1|, |C_2|, |C_3|)/|C_1| = 5$ . This solved configuration is achieved from  $X'$  by  $F'_\Theta$  within  $2\text{lcm}(|C_1|, |C_2|, |C_3|)$  timesteps (on expectation over call sequences).

Putting it all together, it took about  $2|X| = 2(10n + 16)$  timesteps for the head to travel once from the right end to the left end; and another  $2(10n + 16)$  for it to travel back to the right; and  $2\text{lcm}(|C_1|, |C_2|, |C_3|)$  to finish from there. Hence,  $\text{TTS}(F'_\Theta, X)$  is linear in  $n$ ; whereas under the original rule  $F_\Theta$ ,  $\text{TTS}(F_\Theta, X)$  would have been quadratic in  $n$ .

This construction can be extended to work for all local check schemes. Given a generic  $G(\Theta)$ , suppose we've chosen a maximal initial path  $P_i$  and a maximal terminal path  $P_f$  in  $G(\Theta)$ :



In the above figure, the red path is a maximal initial path, and the blue a maximal terminal path. The set of nodes on paths  $P$  such that  $P[1 : |P_i|] = P_i$  and  $P[|P| - |P_f| + 1 : |P|]$  (i.e. the colored nodes in the figure above) comprises a subgraph  $G_{P_i, P_f}$  on which the construction of  $F'_\Theta$  already works. We now order the  $(P_i, P_f)$  lexicographically, and construct an algorithm that takes each pair  $(P_i, P_f)$  one at a time. First, it constructs  $P_i$  on the first  $|P_i|$  agents, and  $P_f$  on the last  $|P_f|$  agents, and then applies the construction  $F'_{G_{P_i, P_f}}$  relative to that choice; if no solution is found for a given choice of  $(P_i, P_f)$ , the algorithm moves on the next pair in the lexicographic ordering and tries again. Once the last pair is exhausted, the algorithm returns to the first and begins the process again.

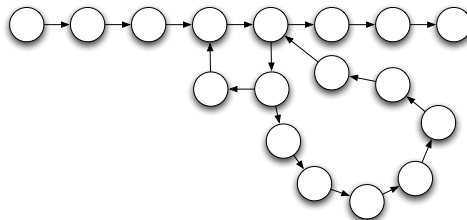
A complete precise description of this algorithm is found in appendix 2. A number of extra states – no more than  $|G(\Theta)|/2$  – are used to implement this algorithm, and we will see in section 5.2 how to remove these states, and the cost of an addition of  $4r$  to the radius of the rule. This results in:

**Proposition 12** *All locally solvable patterns  $T$  are solvable in linear time, by a local rule whose radius of information is at most  $10 \cdot \text{LCR}(T) + 3$ .*

## 4.2 Patterns that Admit Local Patching

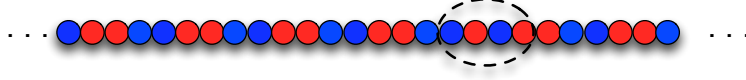
Passing from  $F_\Theta$  to  $F'_\Theta$  reduces the asymptotic runtime from exponential or polynomial to linear. For certain patterns – patterns that admit “local patching” – an even faster solution is possible.

Suppose we are given the radius 3 local check scheme  $\Theta$  on 3 states  $\{0, 1, 2\}$  consisting of the 3-balls arising in the configurations generated by freely alternating combinations of the sequences 1000 and 100112001. The graph  $G(\Theta)$  associated with  $\Theta$  has the structure:



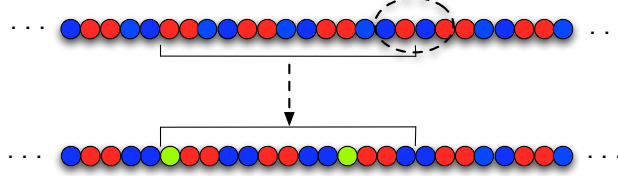
$G(\Theta)$  has 17 nodes, one maximal acyclic path of length 8, and one strongly connected component with two cycles  $C_1$  and  $C_2$  of lengths 4 and 9 respectively.

Now, consider the subconfiguration  $Z_2 = \dots (1001)^4 10(1001)^2 \dots$ ;  $Z_2$  looks like:



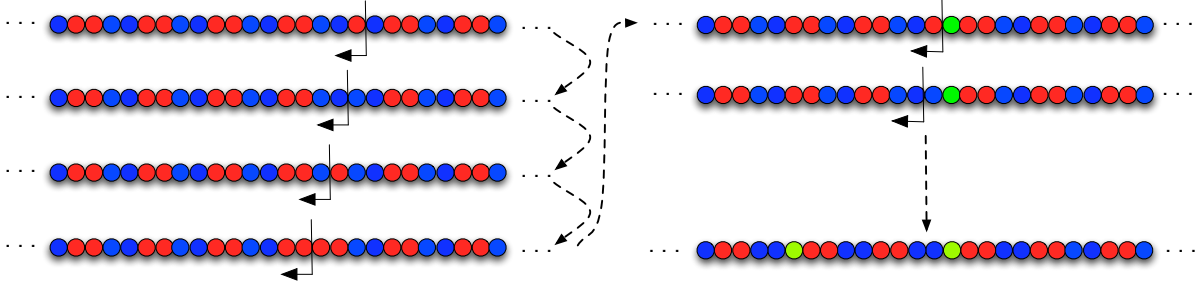
in which blue represents state 1 and red represents 0. This subconfiguration is a repetition of the 4-cycle  $C_1$  everywhere – except for one “bad spot” at the location indicated by the dotted oval, where  $\Theta$  is not satisfied.

It turns out that there is a “local patch” to “correct” the bad spot, achieved 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:

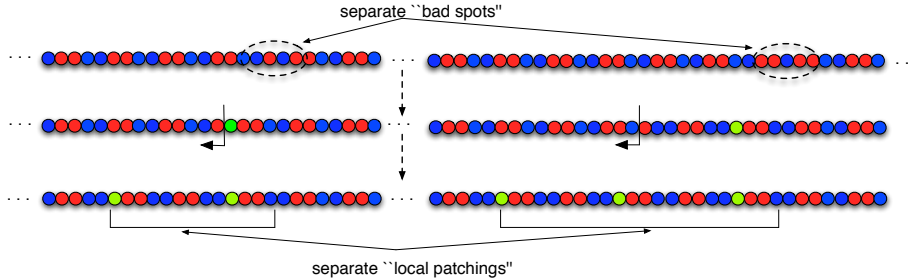


in which green color represents state 2. Such a local patch exists for *all* subconfigurations of the form  $Z_i = \dots (1001)^n \circ 10^i \circ (1001)^m \dots$  for  $i = 0, 1, 2$ , as long as  $n$  is larger than 4. This can be verified case by case, but the generic reason is that the greatest common divisor of the lengths of the two cycles  $C_1$  and  $C_2$  is 1, so by the Chinese Remainder theorem, every integer larger than  $\text{lcm}(|C_1|, |C_2|) = 36$  can be formed by some non-negative combination of  $|C_1|$  and  $|C_2|$ . Also as a result of the Chinese remainder theorem, the size of local patch can always be chosen to be no larger than  $\text{lcm}(|C_1|, |C_2|)$ .

Consider all the configurations that are concatenations of such  $Z_i$ 's, and denote this set  $\mathcal{Z}$ . It is simple to define a local rule  $F_1$  that drives every  $X \in \mathcal{Z}$  into a solved state. Details of this construction are given in appendix 3, but intuitively what  $F_1$  does is to “apply the local patch” at every bad spot, effecting the replacement shown in the above figure one step at a time:

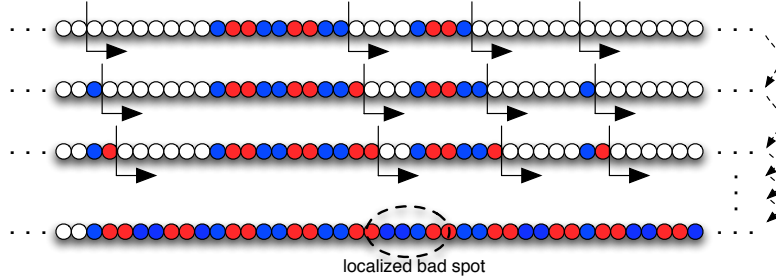


Because each of the bad spots and local patches are independent, these replacements happen independently for each bad spot in the concatenation.

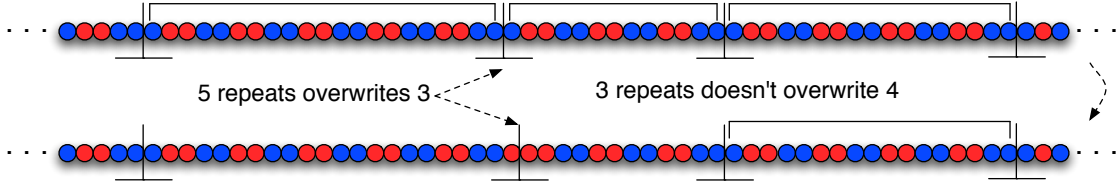


Now, let's see how long it takes for  $F_1$  to finish, for an initial condition  $X \in \mathcal{Z}$ . In any timing model, during every round (i.e. period in which each agent is called at least once), at least one step of each independent patching is made; and each patching is at most of size  $\text{lcm}(|C_1|, |C_2|)$ . Hence,  $\text{TTS}_{\mathcal{S}}(F, X) \leq R_{\mathcal{S}}(|X|) \cdot \text{lcm}(|C_1|, |C_2|)$ , where  $R_{\mathcal{S}}(n)$  is (as defined in 1.3), the average length of a round on a configuration of size  $n$  in the timing model  $\mathcal{S}$ . In the synchronous timing model, the length of any round is 1 so  $\text{TTS}_{\mathcal{S}^s}(F_1, X) \leq \text{lcm}(|C_1|, |C_2|)$  for all  $X \in \mathcal{Z}$ . In the  $k$ -bounded asynchronous model,  $R_{\mathcal{S}^{(k)}}(n) \leq kn$  so  $\text{TTS}_{\mathcal{S}^{(k)}}(F_1, X) \leq k \cdot \text{lcm}(|C_1|, |C_2|)$ , while in the totally asynchronous timing model  $R_{\mathcal{S}^a}(n) \leq K n \log(n)$ , for some constant  $K$ , so  $\text{TTS}_{\mathcal{S}^a}(F_1, X) \leq \text{lcm}(|C_1|, |C_2|) \cdot K \cdot \log(|X|)$ .

It is important to note that rule  $F_1$  is defined *only* on local balls that arise in the trajectories connecting the configurations in  $\mathcal{Z}$  with their corresponding solved states. Denote this set of local balls  $\mathcal{B}_1$ . We have to extend the definition of  $F_1$  to the remaining local balls, so that arbitrary initial conditions are driven to  $\mathcal{Z}$ . The key point now is that we can do this with an algorithm  $F_2$  that has runtime comparable to that of  $F_1$ . The basic principle behind this very simple algorithm is to have any agent that isn't already part of a subconfiguration that repeats  $C_1 = 1000$  start producing  $C_1$ , and to have those agents that are part of such a subconfiguration extend such configurations:

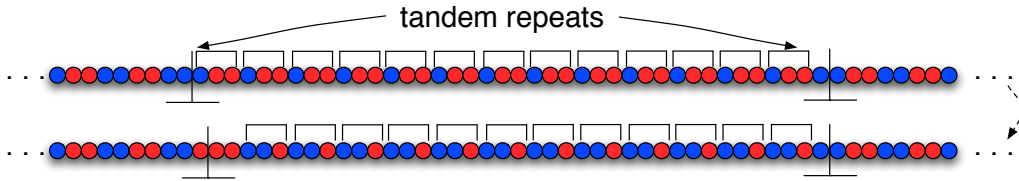


The algorithm  $F_2$  eventually drives any initial configuration to one which is consistent with  $C_1$  at every point except at localized “bad parts”, i.e. a configuration in  $\mathcal{Z}$ . To ensure that regions of correctness always grow, the rule allows a region of  $k$  to overwrite (from the left) a region of size  $l$  only when  $k \leq l$ :



Once any such region is larger than  $lcm(|C_1|, |C_2|)/|C_1|$ , it needn't grow any longer. Details of this construction are also given in appendix 3.

Let's compute, heuristically, how long it takes for  $F_2$  to drive an arbitrary configuration to  $\mathcal{Z}$ . For the synchronous model, it's not hard to see that the configurations which are worst for  $F_2$  are those which have many small, equally-sized repeats of  $C_1$ -incorrect subconfigurations concatenated together (so-called “tandem repeats”):



Because of the synchronous update, each small unit will attempt to overwrite the one to its right simultaneously; assuming that  $|C_1| > 1$ , no progress is made, except on the boundaries of the tandem repeat region. In the worst case, such tandem repeats cover nearly the entire initial configuration  $X$ ; for these  $X$ ,  $TTS_{\mathcal{S}^s}(F_2, X)$  scales with  $|X|$ . On average, however, the length of the largest tandem repeat in a configuration can be shown to scale with  $\log(|X|)$ . Hence, when  $|C_1| > 1$ ,

$$TTS_{\mathcal{S}^s}^{worst}(F_2)(n) \sim A \cdot n \quad \text{and} \quad TTS_{\mathcal{S}^s}^{avg}(F_2)(n) \sim B \cdot \log(n)$$

for some constants  $A$  and  $B$ .

In any asynchronous timing model, or when  $|C_1| = 1$  in the synchronous model, the problem posed by tandem repeats is irrelevant since in the former case all local symmetries are broken by the asynchronous update and in the latter, there are no symmetries to be broken. Hence, the runtime of  $F_2$  is simply bounded by the average round time of the timing model. Thus, in the  $k$ -bounded asynchronous model

$$TTS_{\mathcal{S}(k)}^{worst}(F_2)(n) \sim TTS_{\mathcal{S}(k)}^{avg}(F_2)(n) \sim O(1),$$

while in the totally asynchronous timing model

$$TTS_{\mathcal{S}^a}^{worst}(F_2)(n) \sim TTS_{\mathcal{S}^a}^{avg}(F_2)(n) \sim O(\log(n)).$$

Having defined  $F_1$  and  $F_2$  separately, we combine them

$$\tilde{F}(B) = \begin{cases} F_1(B), & \text{if } B \in \mathcal{B}_1 \\ F_2(B), & \text{otherwise} \end{cases}$$

where  $\mathcal{B}_1$  is, as defined above, the set of local balls on the trajectories between configurations with isolated bad spots and solved states. In words,  $F_1$  acts whenever a local configuration is already a locally isolated bad spot; and  $F_2$  acts otherwise.

This whole set of constructions can be repeated whenever  $G(\Theta)$  contains a strongly connected component  $\mathcal{C}$  such that the greatest common divisor of the lengths of the induced cycles in  $\mathcal{C}$  is 1 – that is, when  $\Theta$  “admits local patching.” (See appendix 3 for details.) As a result,

**Proposition 13** *Suppose  $\Theta$  is a local check scheme such that*

$$g(\Theta) \triangleq \min\{g(\mathcal{C}) \mid \mathcal{C} \in \text{SCC}(G(\Theta))\} = 1.$$

*Then there is a solution  $\tilde{F}_\Theta$  to  $\Theta$  whose run-time 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  $G(\Theta)$ .*

#### 4.2.1 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  $\Theta$  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 Optimality

Having been able to reduce the run-time from polynomial or exponential to linear, and in some specific cases, to constant or logarithmic, we naturally want to know if we’ve done the best we can. In other words, are the fast algorithms we described in the previous two sections optimal? Indeed they are.

Let’s first define what we mean by optimality.

**Definition 10 [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.  $TTS_{\mathcal{S}}^{\text{avg}}(f)(n) \leq K(g)TTS_{\mathcal{S}}^{\text{avg}}(g)(n)$ ).*

**Proposition 14** • *For any locally checkable pattern  $T$  such that all local check schemes  $\Theta$  for  $T$ ,  $g(\Theta) > 1$ , the local rule  $F'_\Theta$  defined in §4.1 is worst- and average-case optimal in all live uniform timing models.*

- *For any nontrivial locally checkable pattern  $T$  for which there is a local check scheme  $\Theta$  such that  $g(\Theta) = 1$ , the local rule  $\tilde{F}_\Theta$  defined in §4.2 is worst- and average-case optimal in the synchronous,  $k$ -bounded asynchronous, and totally asynchronous timing models.*



To see why this holds, let's first consider the claim that in the  $k$ -bounded asynchronous timing model, for any nontrivial local check schemes  $\Theta$  that admits local patching,  $\tilde{F}_\Theta$  is average-case optimal. To establish this we need to show that when  $T$  is nontrivial, for any solution  $g$  to any local check scheme  $\Theta$  for  $T$ , there is a constant  $K(g, k)$  such that  $TTS_{S(k)}^{avg}(g)(n) > K(g, k)$  for all  $n$ . Now, let  $B$  be any ball in  $\mathcal{B}_r(\Theta)$ , and let  $\mathcal{X}_B = \{X \in \mathcal{C} \mid B \in B_r(X)\}$ . It is clear that  $\mathcal{X}_B$  in  $\mathcal{C}$  is a full-measure subset of  $\mathcal{C}$ . On the other hand,  $T$  being nontrivial 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,  $TTS(g)(n) \geq k/(8r(g) + 4)$ .

Now, let's look at the claim that  $\tilde{F}_\Theta$  is optimal for the synchronous model, for nontrivial  $\Theta$  that admit patching. 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), notice that the tandem repeat argument from the previous section works for *any* algorithm, not just  $\tilde{F}$ . 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.

Finally, let's turn to the case of showing that  $F'_\Theta$  is optimal for  $\Theta$  that do not admit patching, i.e. for which for every strongly connected component  $\mathcal{C}$  in  $G(\Theta)$ , the greatest common divisor of the lengths of the induced cycles is greater than 1. Showing this result in for worst-case optimality is simple (the result for the average-case requires several more advanced techniques and is outside the scope of these notes). Let  $f$  be any solution to  $\Theta$ , and let  $C_1$  be a cycle in  $G(\Theta(fix(f)))$ . Choose  $y$  such that

$$\dots C_1^{m(r)} y C_1^{m(r)} \dots$$

is not  $\Theta$ -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  $\Theta$ -consistent. Such a  $y$  can be chosen because  $\Theta$  does not admit patching. Now, let  $Y$  and  $Z$  be such that  $Y \circ C_1^{m(r)} \dots$  is  $\Theta$  consistent and  $\dots C_1^{m(r)} \circ C_1^{m(r)}$  is  $\Theta$ -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  $\Theta$ -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  $C_1$ . Now, in all live timing models evidently  $TTS(f, X_n) > n|C_1|/(2r(f) + 1)$ . Hence

$$TTS^{worst}(f)(n) > ((n - |Y| - |Z|)/a_n - |y| - 2m(r)|C_1|)/2 \geq k(r)n$$

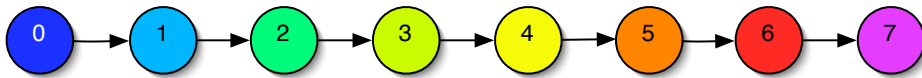
for some constant  $k(r)$ .

## 5 The Radius-State Tradeoff

### 5.1 Coordinates and The Static Tradeoff

Two of the most important properties of local check schemes are their ability to model coordinate systems, and their ability to tradeoff radius for state.

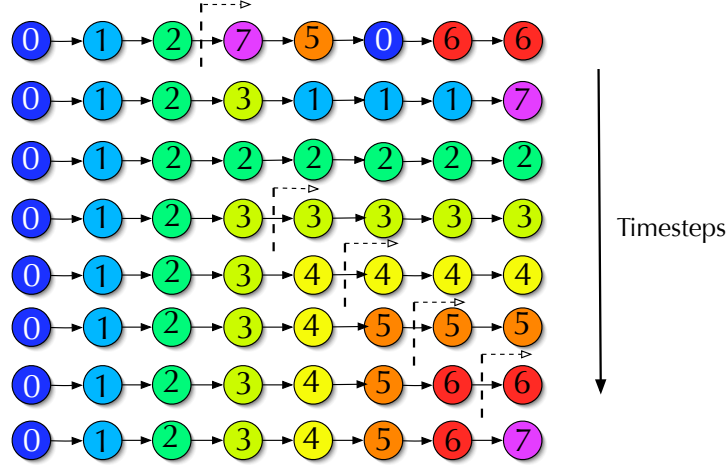
Suppose our task is to self-organize a coordinate system within our one-dimensional model system. Let's make the assumption that the number of possible agent internal states,  $|S| = m$ , is large compared to the size of a configuration  $X$ . A "coordinatization" of a configuration  $X$  with of size 8, with states, would look something like



One simple way to achieve this state is via the radius 1 local rule  $F$  defined on the local ball  $b$  by

$$F(b) = \begin{cases} 0, & \text{if } b \text{ is the left-most agent} \\ 1 + \min(b), & \text{otherwise} \end{cases}$$

in which  $\min(b)$  denotes the minimum state value seen in the local ball  $b$ . As long as  $|X| < m$ , this rule will generate a state in which 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:



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 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*. Because coordinate systems are basic to very many tasks, the gradient is a tool of great importance in local-to-global programming.

The description above suggests that the gradient algorithm is implementing a local check scheme of radius  $1/2$ : namely,

$$\Theta(b) = \begin{cases} 1 & , \text{ if } b(-1) = b(0) - 1 \\ 0 & , \text{ otherwise} \end{cases}.$$

(By a radius of  $1/2$  I mean that the local check scheme only makes use of information from the agent to the left.) The problem with this formulation is that the local check scheme requires 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 (as is increasingly the case for engineering systems) 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:

$$F(b) = 1 + b(-1) \mod k$$

produces the correct result. For each  $k$ , the  $k$ -coordinatization problem is locally checkable with  $k$  states, again 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  $\Theta$  for a  $k$ -coordinatization corresponds to  $G(\Theta)$  being an induced cycle of length  $k$  in  $\mathbb{D}(2r+1, k)$ ; and the cycles of  $\mathbb{D}(1/2, m)$  are those of  $DB(2, m)$ , whose largest induced cycle is of size  $m$ .

However, if we allow a somewhat larger radius, can we lower the required amount of state? Yes. Recall from the literature the idea of a *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  $n$  (they correspond to hamiltonian cycles in the strongly connected component of the graph  $\mathbb{D}((n-1)/2, m)$ ). 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 8** A coordinate gradient of length 8 can be encoded by a DeBruijn sequence with  $n = 3$  and  $m = 2$  (figure 6). 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  $110 \rightarrow 7$ .

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  $B(n, m)$ .



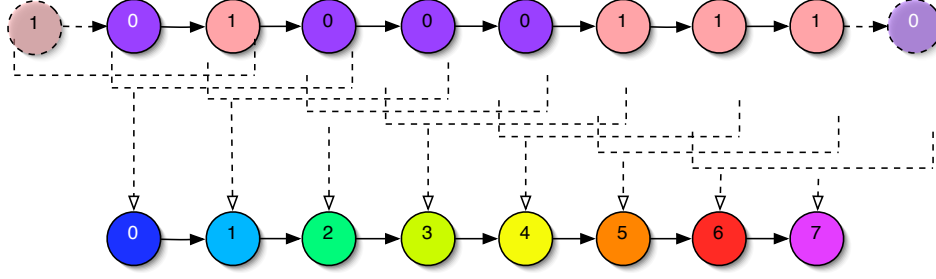


Figure 6: DeBruijn sequence (low state, high radius) encoding of gradient (low radius, high state).

**Example 9** A local check scheme of radius  $3/2$  for  $B(3, 2)$  is given by:

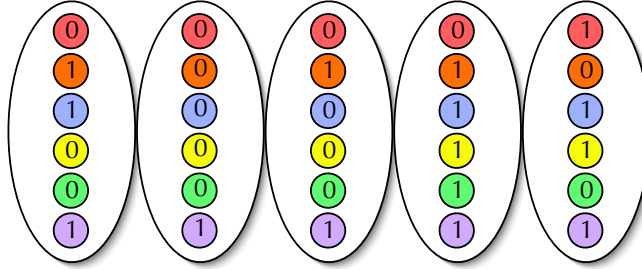
$$\Theta(b) = \begin{cases} 1 & , \text{ for } b = 1010, 0100, 1000, 0001, 0011, 0111, 1110, \text{ and } 1101 \\ 0 & , \text{ otherwise} \end{cases}.$$

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.

Formally, we say that a pattern  $T$  over  $m$  states is the  $k$ -encoding of a pattern  $T'$  over  $m'$  states if there is a function  $\phi : \mathcal{B}_{k, m'} \rightarrow [m]$  such that  $T = \{\phi(X) | X \in T'\}$ . By  $\phi(X)$ , I mean the configuration resulting from applying  $\phi$  to every  $k$ -ball in  $X$ .

**Proposition 15** Any pattern  $T$  over  $m$  states with  $LCR(T) \leq r$  has a local check  $\Theta$  such that  $\Theta(C)$  is the  $r \log_2(m)$ -encoding of a pattern in 2 states that is locally checkable with radius  $\lceil r \log_2(m) \rceil$ , and the 2-encoding of a pattern in  $m^{2^r}$  states that is locally checkable with radius 1.

There are a variety of ways of implementing this tradeoff concretely. One of the most conceptually intuitive arises by adding a little more structure to the model. Instead of modeling the states of the agents as an unstructured set  $S$ , suppose internal state is now specified as a length- $k$  binary sequence. For instance, if  $k = 6$ , then the model will look something like this:



in which the large ovals correspond to agents and each of the slots is assigned a unique color. Since each of the  $k$  “binary register” slots can take on two possible states, the  $k$ -slot model is effectively like choosing  $S$  to have  $2^k$  states. 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  $b = (b_1, b_2, \dots, b_k)$ . 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  $b_6 = (1, 1, 1, 1, 1)$ .

Imagine we’re given a radius  $r$  local check scheme  $\Theta$  with  $k = 1$ , i.e. two states. Suppose our goal is to get a radius  $l$  local check scheme for the pattern  $\Theta(C)$  generated by  $\Theta$ , for any  $l < r$ . To this end, let  $\Theta^l$  be the radius- $l$  local check scheme on

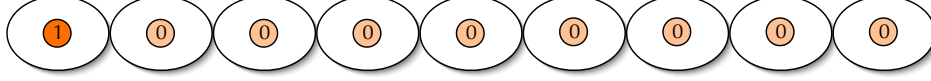
$$M = \lceil (r - l) / (2l + 1) \rceil$$

slots defined by setting  $\Theta^l(b) = 1$  iff there is  $X \in \Theta(C)$  and  $i$  such that

$$b_j = B_l(i + j(2l + 1), X)$$

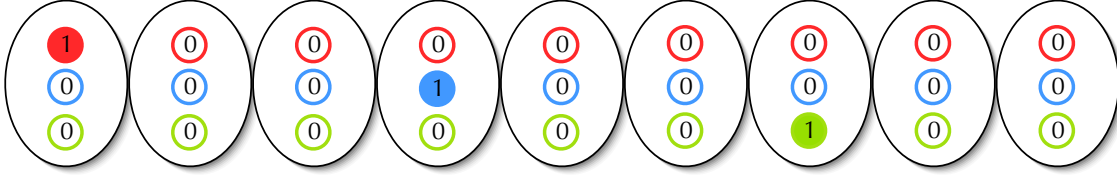
for  $j = -M, \dots, M$ . The pattern encoding can then simply be obtained by reading off the states in  $b_0$ . This construction can be thought of as a “cut and shift” construction, in which the original local check scheme is segmented into a number of separate parts, which are then shifted through the slots.

**Example 10** 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:



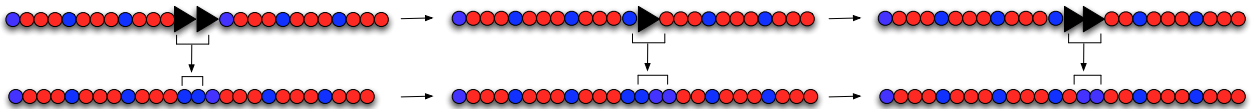
## 5.2 The Dynamic Tradeoff

In the previous section, we discussed a “static” radius/state trade-off, in the sense that the radius/state tradeoff applied to patterns (and their local check schemes), not to the local rules that created those patterns. However, we wish understand the latter as well.

Suppose we’re given a state set  $S_1$ , with size  $m$ . Let  $T \subset \mathcal{C}_{S_1}$  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  $\mathcal{C}_S$ , because whenever  $S \subset S'$ ,  $\mathcal{C}_S \subset \mathcal{C}_{S'}$ . In words,  $T$  is simply a pattern that “doesn’t mention the states in  $S_2$ .” Now, say we’re given a rule  $F$  that solves  $T$  in  $\mathcal{C}_S$ ; that is, for all initial conditions  $X_0$  over the  $m + m'$  states in  $S$ , and all  $s$  in whatever timing model  $\mathcal{S}$  has been chosen,  $\lim_{n \rightarrow \infty} F_s^n(X_0)$  exists and is in  $T$  whenever  $|X|$  is  $T$ -admissible.

The rule  $F$  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  $F_\Theta$  in §2.2;  $\triangleleft$  and  $\triangleright$  and  $\triangle_i$  are just this sort of extra state that is used by the algorithm. The formal description of algorithm  $F'_\Theta$  in appendix 2 uses even more such states. We are thus led to ask if can we always find a new rule  $F'$ , possibly with a larger radius, that is a solution  $T$  over  $\mathcal{C}_{S_1}$ . I.e. can we make a radius/state tradeoff to encode the extra states as larger-radius structures using just the original states, so that the resulting rule still is a solution to  $T$  with the same asymptotic run-time scaling?

Take for example of  $F_\Theta$ , in the case where  $\Theta$  is the radius-2 check scheme for the pattern  $T = \{(100)^n\}$ . In the algorithm  $F_\Theta$ ,  $\triangleright \circ \triangleright$  acts as a “turing head”, a marker for the moving border of  $\Theta$ -correctness. As the  $\triangleright$ -head progresses, there is an alternation of a state with one  $\triangleright$  and a state with two  $\triangleright$ ’s. We can replace this with an alternation of a state with three 1s, i.e. 111, and a state with 1’s i.e. 11:



Because neither 111 nor 11 are states that are present in the final correct pattern, they can be chosen to act as “movable coherent border markers”, just like the  $\triangleright$ -states originally did. It’s not hard to modify the ten rules defining  $F_\Theta$  so that it uses these 111 and 11 subconfigurations in place of  $\triangleright$  and  $\triangleright \circ \triangleright$ .

Now, suppose we take  $\Theta$  to be the radius-3 check scheme for the pattern  $\{(100)^n(1000)^m\}$ . In this case to replicate the states used in  $F_\Theta$ , we have to encode a  $\triangleright/\triangleright \circ \triangleright$  head, as well as a  $\triangleleft/\triangleleft \circ \triangleleft$  state, as well as a  $\triangle_0$  state. Figure 7 shows how this is to be accomplished. Notice that the size of the encoded structures gets larger the more different states need to be encoded. This has two consequences: first, the radius of the local rule must increase commensurately. Second, the dynamic motion of these larger newly encoded movable coherent borders may have to be slower than the dynamics of the original

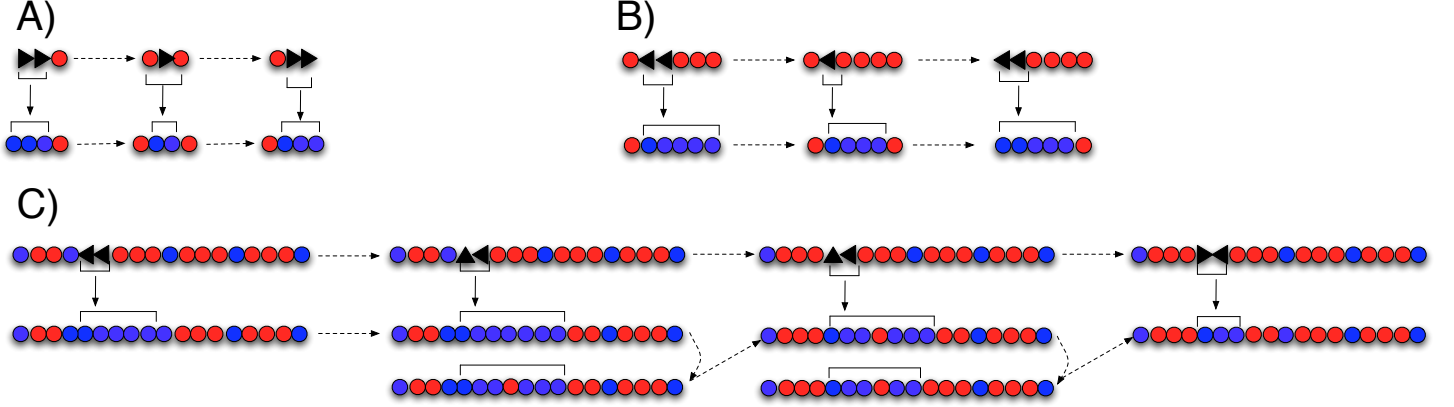


Figure 7: Encoding rules for removing extra states from  $F_\Theta$ , in case of  $\Theta(\mathcal{C}) = \{(100)^n(1000)m\}$ . A) Rules for  $\triangleright$  encoding. B) Rules for  $\triangleleft$  encoding. C) Rules for  $\Delta_0$  encoding.

turing head markers, since several states following each other in time in the encoded version may have to correspond to a single state in the original version.

For a general  $\Theta$ , the procedure for encoding new states to replicate the action of  $F_\Theta$  is to first identify a configuration  $\zeta$  such that no solved state in  $\Theta$  contains  $\zeta$  as a subconfiguration. (This is possible when  $\Theta$  is non-trivial; if  $\Theta$  is trivial then the algorithm  $F^*$  described in §4.2.1 solves  $\Theta$  already with no need for extra states.) Then, suppose we need to encode  $m'$  states  $\{s_1, \dots, s_{m'}\}$ . Let  $k = \lceil \log_m(m') \rceil$ , and assign each  $s_j$  to a unique length- $k$   $m$ -ary sequence  $\sigma_j$ . Finally, we replace the state  $s_j$  with the sequence  $\tau_j = \zeta \circ \sigma_j \circ \zeta$ . The dynamics on the extra states that  $F_\Theta$  generates can be replicated by similar rules operating on the  $\tau_j$  sequences (the details are tedious but straightforward). Because  $F_\Theta$  requires  $m' = m + 2$  extra states, the process of replacing  $s_j$  with  $\tau_j$  requires extending the radius of the rule by  $|\tau_j| = 2|\zeta| + 2\log_m(m + 2)$ . A  $\zeta$  of size no larger than  $2r$  can be found, so we need only extend the radius by a quantity that is at most  $4r + 2$ . For  $F'_\Theta$ , the requirement is for at most  $m^{2r}$  extra states, leading to an extension of the radius by  $8r + 2$ .

## 6 “Glocal” Compilation

What we’re really after is a glocal-to-local (or “glocal”) compiler. What I mean by this is as follows. Recall from 1.4 that we defined  $\mathcal{T}$  to be the set of all patterns. Recall from 1.2 that we defined  $\mathcal{D}$  be the set of all local rules. Recall from 1.5 that we defined  $\mathcal{F}(T)$  be the set of local rules that are solutions to  $T$ . Let  $\emptyset$  be a special symbol. Intuitively, a glocal compiler should be a computable function

$$\mathcal{GC} : \mathcal{T} \rightarrow \mathcal{D} \cup \{\emptyset\}$$

such that

$$\mathcal{GC}(T) = \begin{cases} F \in \mathcal{F}(T), & \text{if } \mathcal{F}(T) \neq \emptyset \\ \emptyset, & \text{if } \mathcal{F}(T) = \emptyset \end{cases}.$$

In words,  $\mathcal{GC}(\cdot)$  is an algorithm that on input  $T$  returns a solution to  $T$  if  $T$  is solvable, and indicates an error (represented by  $\emptyset$ ) otherwise. In an eventual application of this theory, we’d think of the user as programming the amorphous computer (in this case, a linear array of computing agents) by specifying the input  $T$  as a global problem description, and then allowing  $\mathcal{GC}$  to compile down from the global problem to local agent-based rules  $F$ .

Algorithm 2.2 gets us part of the way. The problem, however, is that to make the procedure really automatic and computable, the input  $T$  must be *described finitely*. Just thinking of  $T$  as an unstructured set of final configuration, as we have been doing so far, is not enough. If we want to work with big or infinite pattern sets (which we surely do), we need a description language that efficiently captures the inherent structure of those sets.

Of course, a local check scheme is a finite description of a problem, and so we could ask the user to input problems directly as local check schemes. However, as example 3 shows, making descriptions in the LCS format seems laborious and unnatural. The challenge then is two-fold: A) finding a description language that is appropriate for actual users, in which

input corresponds with natural ways of describing problems, and B) figuring out how to translate from that description language of problems into a local check scheme description.

Here are two simple approaches to meeting this challenge:

## 6.1 Sampling for Local Features

Let  $\mathcal{X} = \{X_1, X_2, \dots, X_K\}$  be a finite set of “sample configurations,” and fix an “feature radius”  $r$ . Let

$$\mathcal{B}(\mathcal{X}, r) = \{B_r(i, X) | X \in \mathcal{X}, i \in \{1, \dots, |X|\}\},$$

that is, the set of all  $r$ -balls present in the samples. Define a local check scheme  $\Theta(\mathcal{X}, r)$  as the indicator function on the set  $\mathcal{B}(\mathcal{X}, r)$ , i.e.  $\Theta(\mathcal{X}, r)$  takes everything in  $\mathcal{B}(\mathcal{X}, r)$  to 1 and everything else to 0. The local rule

$$F(\mathcal{X}, r) = F'_{\Theta(\mathcal{X}, r)}$$

constructs configurations that look identical, up to the feature radius  $r$  like the sample configurations in  $\mathcal{X}$ .

**Example 11** Let

$$\mathcal{X} = \{100, 100100, 100100100, 1000, 10001000, 1001000\},$$

and set  $r = 3$ . Then  $F(\mathcal{X}, r)$  is a solution to the pattern  $\{(100)^n(1000)^m | n, m \in \mathbb{N}\}$ .

## 6.2 Logic

Another approach is to use logic. Let  $S$  be a finite set, and let  $\mathbb{L}$  be the set containing the symbols

$$\mathbb{L} = \{', \neg', \wedge', \vee', \in', ('), ', x_1, x_2, \dots, x_n, \dots\}.$$

**Definition 11** A first-order formula with added constants  $S$  is any symbol string considered well-formed according to the rules:

1.  $\phi(x) = [x = s]$  is well-formed, for each  $s \in S$ , the set of node labels, and any variable  $x$ .
2.  $\phi(x) = [x = y]$  is well-formed for all variables  $x$  and  $y$ .
3.  $\phi(x) = [x \in y]$  is well-formed for all variables  $x$  and  $y$ .
4.  $\phi(\vec{x}_1 \circ \vec{x}_2) = [\phi(\vec{x}_1) \wedge \psi(\vec{y}_2)]$  is well-formed for all well-formed formulas  $\phi(\vec{x}_1)$  and  $\psi(\vec{x}_2)$ , where  $\vec{x}_1 = (x_{i_1}, \dots, x_{i_k})$  and  $\vec{x}_2 = (x_{j_1}, \dots, x_{j_l})$ , respectively. We allow  $i_m = i_n$  or  $j_m = j_n$  only when  $m = n$ , and the notation  $\vec{x}_1 \circ \vec{x}_2$  denotes the amalgamated list of the two vectors of variables.
5.  $\phi(x) = [\neg\phi(\vec{x})]$  is well-formed for all well-formed formulas  $\phi$  of variable vector  $\vec{x} = (x_{i_1}, \dots, x_{i_k})$ .
6.  $\phi(\vec{x}) = [\forall x_{i_0} \phi(x_{i_0}, \vec{x})]$  is well-formed for all well-formed formulas  $\phi$  of variable vector  $(x_{i_0}, x_{i_1}, \dots, x_{i_k})$ .

The notation  $\exists x_0 \phi(x_0, \vec{x})$  is a shorthand for  $\neg[\forall x_0 [\neg\phi(x_0, \vec{x})]]$  and  $\phi(\vec{x}_1) \vee \psi(\vec{x}_2)$  is shorthand for  $\neg[[\neg\phi(\vec{x}_1)] \wedge [\neg\psi(\vec{x}_2)]]$ . The number of free variables in a formula  $\phi(\vec{x})$  is the length of  $\vec{x}$  as a list. A sentence is a formula with zero free variables. The quantifier rank of a formula, denoted  $Q(\phi(\vec{x}))$ , is defined by: setting it as zero for the formulas of the form 1-3 in the above list (the “atomic” formulae);  $Q(\phi \wedge \psi) = \max(Q(\phi), Q(\psi))$ ;  $Q(\neg\phi) = Q(\phi)$ ; and  $Q(\forall x_0 \phi(x_0, \vec{x})) = Q(\phi(x_0, \vec{x})) + 1$ .

Now, we want to apply logic formulae  $\phi$  to configurations  $X$ . To start with, recall that an ordered pair  $(a, b)$  can be represented as a set in a standard way:  $(a, b) = \{a, \{b\}\}$ . A configuration  $X$  is by definition a graph whose nodes are labeled in the set  $S$ , so it too can be described as a set  $Set(X)$ . In particular, the elements of  $Set(X)$  are the representations of the pairs  $(i, s)$ , one for each node in  $X$ , as well as those of the pairs of pairs  $((i, s), (j, s'))$ , one for each edge. For any first-order formula  $\phi$ , the interpretation of  $\phi$  on  $X$  denoted  $\phi[X]$ , is the truth value of  $\phi$ , in which quantifiers  $\forall x_0$  are taken as ranging over the elements of  $Set(X)$ . The pattern defined by  $\phi$  is the set of configurations  $X \in \mathcal{C}$  such that  $\phi[X] = TRUE$ .

The key fact is that:

**Proposition 16** Any pattern defined by a first-order logic formula  $\phi$  is locally checkable by a local check scheme  $\Theta(\phi)$  with radius  $r \leq 3^{Q(\phi)}$ ; and conversely, all patterns of the form  $\Theta(\mathcal{C})$  for some local check scheme  $\Theta$  are first-order definable.

The proof of this result (which is beyond the scope of these notes) indicates a simple algorithmic procedure for constructing  $\Theta(\phi)$  from  $\phi$  and vice-versa. We therefore have an optimal glocal compiler

$$\mathcal{GC} : \text{First-Order Logic Formulae} \longrightarrow \text{Local Rules}$$

defined by

$$\mathcal{GC}(\phi) = \begin{cases} F'_{\Theta(\phi)}, & \text{if } g(G(\Theta(\phi))) > 1 \\ \tilde{F}_{\Theta(\phi)}, & \text{if } \Theta(\phi) \text{ is nontrivial and } g(G(\Theta)) = 1, \\ F^*, & \text{If } \Theta \text{ is trivial} \end{cases}$$

where  $g(\Theta)$  is as defined at the end of section 4.2.

This result is conceptually pleasing, but the practical question of whether logical description is ever useful (i.e. more intuitive than specifying a local check directly) is not clear to me.

## 7 Appendices

### Details of Algorithm 1

Given an  $r$ -ball  $B$ , define  $\Theta[B]$  as the boolean that is 1 when  $\Theta(B) = 1$ . Given an  $r+k$  ball  $B$ , let for  $l \leq k$ ,  $\Theta_l[B] = \Theta[B[l-r : l+r]]$ , and  $\Theta_{l,j} = \bigwedge_{i=l}^j \Theta_k[B^i]$ . Let

$$\eta[B] = (\star(B) = |B|) \wedge \bigvee_{l=-r+1}^0 \Theta^\neg[B[l-r : 0]],$$

where  $B[l-r : 0]$  is considered as a right-end ball. Let

$$L(B) = (B(0) = \max\{j \in \{0, \dots, m-1\} | \Theta[B[-2r : -1] \circ j]\}).$$

Let  $b_j[B] = \bigwedge_{l=-r}^0 \Theta[B[l-r : -1] \circ j]$ , which asks if any such  $j$  exists at all.

Now, define boolean functions of balls  $B$  by:

$$\begin{aligned} \alpha_1[B] &= \Theta_{-r-1}[B] \wedge (\Theta_{-r}^\neg[B]) \vee \eta[B] \wedge (B(0) \in S), \\ \alpha_2[B] &= (\star(B) > 1) \wedge (B(-1) = \triangleright) \wedge \Theta_{-r-2}[B], \\ \alpha_3[B] &= (\exists j (B(0) = \triangle_j)) \wedge (L[B^{-1}] \vee (B(0) \neq j)), \\ \alpha_4[B] &= (B(0) = \triangleleft) \wedge (\star(B) = 1), \\ \beta_1[B] &= (\star(B) < |B|) \wedge \Theta_{-r-1}[B] \wedge (B(0) = \triangleright) \wedge (\bigwedge \Theta^\neg[B[-2r : -1] \circ j]), \\ \beta_2[B] &= (\star(B) = |B|) \wedge (B(0) = \triangleright) \wedge \Theta_{-r-1}[B] \wedge (\bigwedge_{j=0}^{m-1} b_j^\neg[B]), \\ \beta_3[B] &= (\star(B) < |B|) \wedge (B(1) = \triangleleft) \wedge \Theta_{-r-1}[B] \wedge \Theta_{-r}[B] \wedge L(B), \\ \gamma_1[B] &= (\star(B) < |B|) \wedge \Theta_{-r-1}[x] \wedge (B(1) = \triangleright) \wedge (B(0) = \triangleright) \wedge (\exists j | \Theta[B[-2r : -1] \circ j]), \\ \gamma_2[B] &= (\star(B) = |B|) \wedge (B(0) = \triangleright) \wedge \Theta_{-r-1}[B] \wedge (\exists j | b_j[B]), \\ \delta[B] &= (\star(B) < |B|) \wedge \Theta_{-r}[B] \wedge (B(1) = \triangle_{B(0)}) \wedge L^\neg(B), \end{aligned}$$

and

$$\epsilon[B] = (\star(B) > 1) \wedge \Theta_{-r-2}[B] \wedge \Theta_{-r-1}[B] \wedge (B(0) = \triangleleft) \wedge L^\neg[B^{-1}].$$

The formal definition  $F_\Theta$  is:

---

```

if  $\alpha_1[B] \vee \alpha_2[B] \vee \alpha_3[B] \vee \alpha_4[B]$  then
  |  $F(B) = \triangleright$ 
else if  $\beta_1[B] \vee \beta_2[B] \vee \beta_3[B]$  then
  |  $F(B) = \triangleleft$ 
else if  $\gamma_1[B]$  then
  |  $F(B) = \min\{j \in [0, m-1] \mid \Theta[B[-2r : -1] \circ j]\}$ 
else if  $\gamma_2[x]$  then
  |  $F(B) = \min\{j \in [0, m-1] \mid b_j[B]\}$ 
else if  $\delta[B]$  then
  |  $F(B) = \min\{j > B(0) \mid \Theta[B[-2r : -1] \circ j]\}$ 
else if  $\epsilon[B]$  then
  |  $F(B) = \triangle_{B(-1)}$ 
else
  |  $F(B) = B(0)$ .
end

```

---

Let

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

be any function such that  $O(b, \cdot)$  is a bijection from  $\{0, \dots, |R(b)| - 1\}$  to  $R(b)$ . Let  $O^{-1}(B)$  denote the unique  $j$  such that  $O(B[-2r-1 : -1], j) = B(0)$  if such exists. Let  $F_\Theta^O$  be defined by modifying  $F_\Theta$  with the replacements:

- In **Rule 5**: the boolean  $(\exists j | \Theta[B[-2r : -1] \circ j])$  replaced with  $(|R(B[-2r-1 : -1])| = 0)$ ,
- In **Rule 2**: the state  $\min\{j \in [0, m-1] | \Theta_{-r}[B[-2r : -1] \circ j]\}$  in replaced with  $O(B[-2r-1 : -1], 1)$ ,
- In **Rule 6**: the boolean  $L(B)$  replaced with the boolean  $(B(0) = O(B[-2r-1 : -1], |R(B[-2r-1 : -1])|))$ , and
- In **Rule 8**: the state  $\min\{j > B(0) | \Theta[B[-2r : -1] \circ j]\}$  replaced with  $O(B[-2r-1 : -1], O^{-1}(B) + 1)$ .

## Appendix 2: Details of Linear-Time Algorithm

Recall definitions of  $L(b)$  and  $R(b)$ . Recall the definition of a choice function  $O$ . Analogously, let  $P : \mathcal{B}_{2r+1} \times [m] \rightarrow [m]$  be any function such that restricted to  $b \in \Theta^{-1}(1)$  and  $j \in \{0, \dots, |L(b)| - 1\}$ ,  $P(b, j) \in L(b)$  and  $P(b, j) = P(b, i)$  only if  $i = j$ . Let  $P^{-1}(B)$  denote the unique  $j$  such that  $P(B[1 : 2r+1], j) = B(0)$  if such exists.  $O$  is a “right choice function” and  $P$  is a “left choice function”.

Let  $O$  be a right choice function such that if for  $b$  of length  $2r+1$  then if  $b' = b[2 : 2r+1] \circ O(b, i)$ , and  $b'' = b[2 : 2r+1] \circ O(b, j)$  then  $i > j$  if there is no path in  $G(\Theta)$  from  $b$  to  $b'$  but there is from  $b$  to  $b''$ . Let  $P$  be a left choice function such that if for  $b$  of length  $2r+1$  then if  $b' = P(b, i) \circ b[1 : 2r]$ , and  $b'' = P(b, j) \circ b[1 : 2r]$  then  $i > j$  if there is no path in  $G(\Theta)$  from  $b'$  to  $b$  in  $G(\Theta)$  but there is one from  $b''$  to  $b$ .

Let

$$\lambda(B) = (B(0) = O(B[-2r-1 : -1], |R(B[-2r-1 : -1])|),$$

and  $\Lambda_L(B)$  is the boolean that is 1 if  $B(0) = O(B[-2r-1 : -1], j)$  where  $j$  is the maximal value such that  $B[-2r : -1] \circ O(B[-2r-1 : -1], j)$  is in  $\Theta$  and connected by a path in  $G(\Theta)$  to  $B[2 : 2r+1]$ , and  $\Lambda_R(B)$  is the boolean that is 1 if  $B(0) = P(B[1 : 2r+1], j)$  where  $j$  is the maximal value such that  $P(B[1 : 2r+1], j) \circ B[2 : 2r]$  is in  $\Theta$  and is accessible by a path in  $G(\Theta)$  from  $B[-2r-2 : -2]$ . Let  $\Gamma(a, b)$  be the boolean which is 1 when  $a, b' \circ b \in G(\Theta)$  for some string  $b'$  with  $|b'| = \max(0, r+1 - |b|)$ , and there is a path in  $G(\Theta)$  connecting  $a$  to  $b' \circ b$ . Let  $Sol(B)$  be the boolean that is 1 if there is a state  $j$  such that  $\bigwedge_{i=0}^{2r} B[-2r-i : -1] \circ j \circ B[1 : i]$ . Let

$$Solv(B) = \min\{j \mid \bigwedge_{i=0}^{2r} B[-2r-i : -1] \circ j \circ B[1 : i]\}.$$

Let  $T_R(b) = 1$  if  $\Theta(b)$  and there is no cycle  $C$  in  $G(\Theta)$  for which there is a path in  $G(\Theta)$  from  $b$  to  $C$ . Let  $T_L(b) = 1$  if  $\Theta(b)$  and there is no cycle  $C$  in  $G(\Theta)$  for which there is a path in  $G(\Theta)$  from  $C$  to  $b$ .

- **Rule 1:** If

$$\alpha_1[B] = \Theta_{-r-1}[B] \wedge \Theta_{-r}^-[B] \wedge (B(0) \in S)$$

or

$$\alpha_2[B] = (\star(B) > 1) \wedge (B(-1) = \triangleright_1) \wedge \Theta_{-r-2}[B]$$

or

$$\alpha_3[B] = (\exists j(B(0) = \triangle_j)) \wedge (\lambda[B^{-1}] \vee (B(-1) \neq j))$$

or

$$\alpha_4[B] = (B(0) = \triangleleft_{1,2,3}) \wedge (\star(B) = 1)$$

holds then let  $\tilde{F}(B) = \triangleright_1$ .

- **Rule 2:** Else if

$$\alpha_5[B] = (\star(B) < |B|) \wedge \Theta_{-r-1}[B] \wedge (B(0) = \triangleright_{1,2,3}) \wedge (\bigwedge \Theta^-[B[-2r : -1] \circ j])$$

or

$$\alpha_6[B] = (\star(B) < |B|) \wedge (B(1) = \triangleleft_1) \wedge \Theta_{-r-1}[B] \wedge \Theta_{-r}[B] \wedge \lambda(B) \wedge \Theta_{r+2}(B)$$

holds then let  $\tilde{F}(B) = \triangleleft_1$ .

- **Rule 3:** Else if

$$\alpha_7[B] = (\star(B) = |B|) \wedge (B(0) \neq \triangleright_3) \wedge \Theta_{-r-1}[B] \wedge (\bigwedge_{j=0}^{m-1} b_j^-[B])$$

holds then let

$$\tilde{F}(B) = \begin{cases} \triangleleft_1, & \text{if } T_R(B[-2r-1 : -1]) \text{ or } T_L(B[-2r-1 : -1]) \\ \triangleleft_2, & \text{otherwise} \end{cases}.$$

- **Rule 4:** Else if

$$\alpha_8[B] = (\star(B) > 1) \wedge ((B(0) = B(-1) = \triangleleft_1) \vee (B(0) = B(-1) = \triangleleft_3)) \wedge \Theta_{r+1}(B)$$

holds let  $\tilde{F}(B) = P(B[1 : 2r+1], 1)$ .

- **Rule 5:** Else if

$$\alpha_9[B] = (\star(B) < |B|) \wedge \Theta_{-r-1}[B] \wedge ((B(1) = B(0) = \triangleright_1) \vee (B(1) = B(0) = \triangleright_3)) \wedge (\exists j|\Theta[B[-2r : -1] \circ j])$$

holds let  $\tilde{F}(B) = O(B[-2r-1 : -1], 1)$ .

- **Rule 6:** Else if

$$\alpha_{10}[B] = (\star(B) = |B|) \wedge \Theta_{-r-1}[B] \wedge Sol(B)$$

or

$$\alpha_{11}[B] = (B(0) = \triangleright_2) \wedge \Theta_{-r-1}[B] \wedge \Theta_{r+1}[B] \wedge T_R[B[1 : 2r+1]] \wedge T_L^-[B[-2r-1 : -1]] \wedge \Gamma(B[-2r-1 : -1], B[1 : 2r+1]) \wedge Sol(B)$$

holds let  $\tilde{F}(B) = Solv(B)$ .

- **Rule 7:** Else if

$$\alpha_{12}[B] = (\star(B) < |B|) \wedge \Theta_{-r}[B] \wedge (B(1) = \triangle_{B(0)}^1) \wedge \lambda^-(B)$$

holds let  $\tilde{F}(B) = O(B(-2r-1 : -1), O^{-1}(B(0)) + 1)$ .

- **Rule 8:** Else if

$$\alpha_{13}[B] = (\star(B) > 1) \wedge \Theta_{-r-2}[B] \wedge \Theta_{-r-1}[B] \wedge (B(0) = \triangleleft_1) \wedge \lambda^-[B^{-1}]$$

or

$$\alpha_{14}[B] = (\star(B) > 1) \wedge (B(0) = \triangleleft_3) \wedge \Theta_{r+1}[B] \wedge \Theta_{-r-1}[B] \wedge \Theta_{-r-2}[B] \wedge T_L(B[-2r-1 : -1]) \wedge \lambda^-(B^{-1})$$

holds let  $\tilde{F}(B) = \triangle_{B(-1)}^1$ .

- **Rule 9:** Else if

$$\alpha_{15}[B] = (\star(B) < |B|) \wedge (B(1) = \triangleleft_2) \wedge \Theta_{-r-1}[B] \wedge \Theta_{-r}[B] \wedge \Theta_{r+2}[B] \wedge (\Lambda_L(B) \vee T_R(B[2 : 2r+2])) \wedge \\ \wedge T_L^-(B[-2r : 0]) \wedge T_R^-(B[-2r : 0]) \wedge \Gamma(B[-2r : 0], B[2 : 2r+2])$$

or

$$\alpha_{16}[B] = (B(0) = \triangleright_2) \wedge \Theta_{-r-1}[B] \wedge \Theta_{r+1}[B] \wedge T_R[B[1 : 2r+1]] \wedge T_L^-(B[-2r-1 : -1]) \wedge \Gamma(B[-2r-1 : -1], B[1 : 2r+1]) \wedge Sol^-(B)$$

or

$$\alpha_{17}[B] = (\exists j(B(0) = \triangle_j^3)) \wedge (\Lambda_R[B^{+1}] \vee (B(1) \neq j))$$

holds then let  $\tilde{F}(B) = \triangleleft_2$ .

- **Rule 10:** Else if

$$\alpha_{18}[B] = (\star(B) > 1) \wedge (B(-1) = \triangleleft_2) \wedge (B(0) = \triangleleft_2) \wedge \Theta_{-r-2}[B] \wedge \Theta_{r+1}[B] \wedge \Gamma(B[-2r-2 : -2], B[1 : 2r+1])$$

holds then let  $\tilde{F}(B) = P(B[1 : 2r+1], 1)$ .

- **Rule 11:** Else if  $\alpha_{18}[B^{+1}]$  holds let  $\tilde{F}(B) = B(0)$ .

- **Rule 12:** Else if

$$\alpha_{19}[B] = (\star(B) > 1) \wedge \Theta_{-r-2}[B] \wedge \Theta_{-r-1}[B] \wedge (B(0) = \triangleleft_2) \wedge \Lambda_L^-[B^{-1}] \wedge T_R^-(B[1 : 2r+1]) \wedge T_L^-(B[-2r-1 : -1]) \wedge \\ \wedge \Gamma(B[-2r-1 : -1], B[1 : 2r+1])$$

holds then let  $\tilde{F}(B) = \triangle_{B(-1)}^2$ .

- **Rule 13:** Else if

$$\alpha_{20}[B] = (\exists j(B(0) = \triangle_j^2)) \wedge (\Lambda_L[B^{-1}] \vee (B(-1) \neq j))$$

or

$$\alpha_{21}[B] = (\star(B) > 1) \wedge (B(-1) = \triangleright_2) \wedge \Theta_{-r-2}[B] \wedge \Theta_r[B] \wedge \Theta_{r+1}[B] \wedge T_R^-(B[1 : 2r+1]) \wedge \Gamma(B[-2r-2 : -2], B[0 : 2r])$$

holds let  $\tilde{F}(B) = \triangleright_2$ .

- **Rule 14:** Else if

$$\alpha_{22}[B] = (\star(B) < |B|) \wedge \Theta_{-r}[B] \wedge \Theta_{-r-1}[B] \wedge \Theta_{r+2}(B) \wedge \Gamma(B[-2r : 0], B[2 : 2r+2]) \wedge (B(1) = \triangle_{B(0)}^2) \wedge \Lambda_L^-(B) \wedge T_L^-(B[-2r : 0])$$

or

$$\alpha_{23}[B] = (\star(B) < |B|) \wedge \Theta_{-r-1}[B] \wedge (B(1) = \triangleright_2) \wedge (B(0) = \triangleright_2) \wedge \Theta_{r+2}[B] \wedge \\ \wedge \Gamma(B[-2r : 0], B[2 : 2r+2]) \wedge (\exists j|\Gamma(B[-2r : -1] \circ j, B[2 : 2r+2]))$$

hold let

$$\tilde{F}(B) = O(B(-2r-1 : -1), j)$$

where  $j$  is the minimum  $l > O^{-1}(B(0))$  such that  $\Gamma(B[-2r : -1] \circ l, B[2 : 2r+2])$ .

- **Rule 15:** Else if

$$\alpha_{24}[B] = (B(0) = \triangleleft_2) \wedge (\star(B) > 1) \wedge \Theta_{-r-1}[B] \wedge T_L(B[-2r-1 : -1])$$

or

$$\alpha_{25}[B] = (\star(B) > 1) \wedge (B(-1) = \triangleright_3) \wedge \Theta_{-r-2}[B] \wedge \Theta_r[B] \wedge \Theta_{r+1}[B] \wedge (T_R(B[0 : 2r]) \Rightarrow \Lambda_R(B))$$

hold let  $\tilde{F}(B) = \triangleright_3$ .

- **Rule 16:** Else if

$$\alpha_{26}[B] = (\star(B) < |B|) \wedge (B(0) = \triangleright_3) \wedge \Theta_{-r-1}[B] \wedge \Theta_{r+1}[B] \wedge \Theta_{r+2}[B] \wedge T_R(B[1 : 2r+1]) \wedge \\ \wedge \Gamma(B[-2r-1 : -1], B[1 : 2r+1]) \wedge \Lambda_R^-(B^{+1})$$

holds let  $\tilde{F}(B) = \triangle_{B(1)}^3$ .



- **Rule 17:** Else if

$$\alpha_{27}[B] = (\star(B) > 1) \wedge \Theta_{-r-2}[B] \wedge \Theta_r[B] \wedge \Theta_{r+1}(B) \wedge (B(-1) = \Delta_{B(0)}^3) \wedge T_R(B[0 : 2r]) \wedge \Gamma(B[-2r-2 : -2], B[0 : 2r]) \wedge \Lambda_R^-(B)$$

holds let

$$\tilde{F}(B) = P(B(1 : 2r + 1), j)$$

where  $j$  is the minimum  $l > P^{-1}(B(0))$  such that  $\Gamma(B[-2r-2 : -2], l \circ B[2 : 2r])$ .

- **Rule 18:** Else if

$$\alpha_{28}[B] = (\star(B) = |B|) \wedge (B(0) = \triangleright_3) \wedge \Theta_{-r-1}[B] \wedge \text{Sol}^-(B)$$

or

$$\alpha_{29}[B] = (\star(B) < |B|) \wedge (B(1) = \triangleleft_3) \wedge \Theta_{-r}[B] \wedge \Theta_{r+2}[B] \wedge \Theta_{-r-1}[B] \wedge (T_L(B[-2r : 0]) \Rightarrow \lambda(B))$$

holds let  $\tilde{F}(B) = \triangleleft_3$ .

- **Rule 19:** Else if  $B(0) \in \{\triangleright_{2,3}, \triangleleft_{2,3}, \Delta_i^{2,3}\}$  let  $\tilde{F}(B) = \triangleright_1$ .
- **Rule 20:** Else  $F(B) = B(0)$ .

As define  $\tilde{F}$  does a lexicographic search over initial/terminal segment pairs, and a lexicographic search within each one. Now, we will compose it with the greatest-common-divisor condition that generates the linear time algorithm.

### GCD Condition

Suppose  $b \in \Theta^{-1}(1)$  is such that  $T_R^-(b)$ , but for which there is  $i \in R(b)$  such that  $T_R(b[2 : |b|] \circ i)$ . Introduce new states  $\triangleleft_{2,b}$  and  $\Delta_{B(0),y}^2$  for each such  $b$ , addition to  $\triangleleft_2$  as  $\triangleleft_{2,\emptyset}$ .

Given an acyclic path  $Q$  in  $G(\Theta)$ , recall from §3.3 the definition of  $\text{gcd}(Q)$  denote the greatest common divisor of the induced lengths of the cycles in the non-trivial strongly connected components in  $G(\Theta)$  that  $Q$  intersects.

Let  $y \in G(\Theta)$  satisfy  $T_L^-(y)$ . Define inductively  $y^n = P(y^{n-1}, 1)$ , where  $y^0 = y$ . Because  $y$  is not left-terminal, this set of nodes  $\{y^n | n \in \mathbb{N}\}$  in  $G(\Theta)$  contains a single cycle and a path exiting that cycle. Let  $C(y)$  denote the cycle, and  $D(y)$  the path. Given  $b \in C(y)$ , let  $\text{dist}(b, y, C)$  denote the length of path in  $C(y)$  from  $b$  to  $y$ . Now, let  $\zeta(a, b, y)$  be the boolean which is 1 IFF  $b \in C(y)$  and there is a path  $P$  from  $a$  to  $b$  in  $G(\Theta)$  such that for some  $n, M$ ,

$$|P| + N \cdot \text{gcd}(P) = |D(y)| + \text{dist}(b, y, C(y)) + M \cdot |C(y)| + 2r + 2.$$

This is a simple local computation about relative gcds of cycle lengths along the path  $P$ . Define  $\psi(B)$  as the boolean which is 1 whenever  $B(1) = \triangleleft_{2,y}$  or  $\Delta_{B(0),y}^2$  for some  $y$ , and there is  $j > O^{-1}(B)$  such that  $\zeta(B[-2r : -1] \circ O(B[-2r-1 : -1], j), B[2 : 2r+2], y)$ .

- Replace  $\alpha_{15}$  with this:

$$\alpha_{15}[B] = (\star(B) < |B|) \wedge (\exists x | B(1) = \triangleleft_{2,x}) \wedge \Theta_{-r-1}[B] \wedge \Theta_{-r}[B] \wedge \Theta_{r+2}[B] \wedge (\psi^-(B) \vee T_R(B[2 : 2r+2])) \wedge T_L^-(B[-2r : 0]) \wedge \Gamma(B[-2r : 0], B[2 : 2r+2])$$

If  $\alpha_{15}[B]$  holds, we define  $\tilde{F}_\Theta$  by

$$\tilde{F}_\Theta(B) = \begin{cases} \triangleleft_{2,y}, & \text{if } T_R(B[2 : 2r+2]) \wedge T_R^-(y) \\ B(1), & \text{otherwise} \end{cases}$$

where  $y = P(B[2 : 2r+2], 1) \circ B[2 : 2r+1]$ .

- Replace  $\alpha_{18}$  with:

$$\alpha_{18}[B] = (\star(B) > 1) \wedge (\exists x | B(-1) == B(0) == \triangleleft_{2,x}) \wedge \Theta_{-r-2}[B] \wedge \Theta_{r+1}[B] \wedge \Gamma(B[-2r-2 : -2], B[1 : 2r+1])$$

If  $\alpha_{18}[B]$  holds, we set

$$F(B) = P(B[1 : 2r+1], 1).$$

- Replace  $\alpha_{19}$  with this:

$$\alpha_{19}[B] = (\star(B) > 1) \wedge \Theta_{-r-2}[B] \wedge \Theta_{-r-1}[B] \wedge (\exists x | B(0) == \triangleleft_{2,x}) \wedge T_R^-(B[1 : 2r+1]) \wedge T_L^-(B[-2r-1 : -1]) \wedge \Gamma(B[-2r-1 : -1], B[1 : 2r+1]) \wedge \psi(B^{-1}).$$

If  $\alpha_{19}[B]$  holds, we set

$$\tilde{F}_\Theta(B) = \Delta_{B(-1),x}^2.$$

- Replace  $\alpha_{22}$  with

$$\alpha_{22}[B] = (\star(B) < |B|) \wedge \Theta_{-r}[B] \wedge \Theta_{-r-1}[B] \wedge \Theta_{r+2}(B) \wedge (\exists B(1) == \Delta_{B(0),x}^2) \wedge \Gamma(B[-2r : 0], B[2 : 2r+2]) \wedge T_L^-(B[-2r : 0]) \wedge \psi(B)$$

If  $\alpha_{22}[B]$  holds, we set

$$F(B) = O(B(-2r-1 : -1), j)$$

where  $j$  is the minimum  $l > O^{-1}(B(0))$  such that  $\zeta(B[-2r : -1] \circ l, B[2 : 2r+2])$ .

### Appendix 3: Details of the Local Patching Algorithm

Suppose  $G(\Theta)$  has at least one strongly connected component  $\mathcal{C}$ , the greatest common divisor of whose cycle lengths is 1. Pick any irreducible cycle  $c_1 \in \mathcal{C}$ ; and let  $P$  be a maximal acyclic path through  $c_1$ . Let  $\alpha$  and  $\beta$ , respectively, denote the maximal initial and terminal paths in  $P \cup c_1$ . Let  $C_1$  denote the  $m$ -ary sequence corresponding to  $c_1$ , starting at first node in the  $P \cap c_1$  after the maximal initial path in  $P$ . Suppose we're given a sub-configuration of the form  $Z = C_1^{on_1} \circ C_1[1 : k] \circ C_1^{on_2}$  where  $k < |C_1|$ . If  $n_1$  and  $n_2$  are sufficiently large – greater than  $lcm(|C_1|, \dots, |C_k|)/|C_1| + 2r$  – then because  $gcd(|C_1|, \dots, |C_k|) = 1$ , there is a subconfiguration  $W$  and  $n'_1 \leq n_1$  such that  $Z' = C_1^{on'_1} \circ W \circ C_1^{on_2}$  is  $\Theta$ -consistent,  $|Z'| = |Z|$ , and  $n'_1 > 2r/|C_1|$ . For each  $k < |C_1|$ , pick one such  $W$ , call it  $W_k$ . Let  $w_k = |W_k|$ , and notice that  $w_k \leq lcm(|C_1|, \dots, |C_k|)/|C_1| + 2r$ .

Define an initial condition  $X_0$  to be *special* if it is a concatenation of subconfigurations of the form  $Z$ , together with proper initial and terminal flanking sequences  $\alpha$  and  $\beta$ :  $X_0 = \alpha \circ Z_1 \circ Z_2 \circ \dots \circ Z_l \circ \beta$  where each  $Z_i = C_1^{on_{1,i}} \circ C_1[1 : k_i] \circ C_1^{on_{2,i}}$  with  $n_{1,i}, n_{2,i} \geq lcm(|C_1|, \dots, |C_{k_i}|)/|C_1| + 2r$  for all  $i$ , and such that  $\alpha \circ Z_1$  and  $Z_l \circ \beta$  are  $\Theta$ -consistent.

Now, we design a rule  $\tilde{F}$  which drives all special initial configurations  $Z$  to solved states. Let  $W_k^0 = C_1^{n_1 - n'_1, k} C_1[1 : k]$  and notice that  $|W_k^0| = |W_k| = w_k$ . Define  $W_k^i = W_k^0[1 : w - i] \circ W_k[w - i + 1 : w]$ . Obviously  $W_k^{w_k} = W_k$ . The sequence  $W_k^0, W_k^1, \dots, W_k^{w_k} = W_k$  forms a trajectory of subconfigurations connecting the bad parts with the patch-overs. The  $W_k^i$  may not be distinct for each  $i$ , since there could be overlap between portions of  $W_k$  and  $C_1$ . Nonetheless there is then a unique agent  $j_k^i$  in  $W_k^i$  such that if  $j_k^i$  changed its state from whatever it is to some (unique) new state  $s_k^i$ ,  $W_k^i$  would correspondingly change to  $W_k^{(i,k)^+}$ , where  $(i,k)^+$  is the minimal  $l > i$  such that  $W_k^l \neq W_k^i$ .

Consider the set of configurations  $\mathcal{Z}$  generated from each special configuration  $Z = \alpha \circ \bigcirc Z_{k_i} \circ \beta$  by replacing  $Z_i$  with  $C_1^{on_{1,i}} W_k^{j_i} C_1^{on_{2,i}}$  for each  $j_i \leq w_{k_i}$ . That is, we replace the “bad portion” of each  $Z_i$  with some step  $j$  along the trajectory joining that bad portion and the corrected patch  $W_{k_i}$ . Let  $R = lcm(|C_1|, \dots, |C_k|)/|C_1| + 2r$ , and define the “special balls”  $\mathcal{B}$  as the set of  $R$ -balls that arise in configurations in  $\mathcal{Z}$ .

For each  $b \in \mathcal{B}$ , define a local rule  $F_1$  of radius  $R$  on  $b$  by

$$F_1(b) = \begin{cases} s_k^i, & \text{if } b = B_R(j_k^i, W_k^i) \\ b(0), & \text{otherwise} \end{cases}.$$

Under this rule initial configurations that are special are driven to solved configurations along the trajectories in  $\mathcal{Z}$ . Of course, the above definition of  $F_1$  is only for a subset of all balls of radius  $R$  (those in  $\mathcal{B}$ ), i.e. so far  $\tilde{F}$  is only partially specified.

Now, we have to extend the definition of  $F_1$  to  $\mathcal{B}_R - \mathcal{B}$ , the set of all  $R$ -balls besides those on which  $F_1$  is already defined, so that arbitrary initial conditions are driven to  $\mathcal{Z}$ . Given ball  $b$  in  $\alpha$  (the initial path), there is a unique  $j$  such that  $b \circ j$  is consistent with  $P \cup c_1$ . Call it  $b^+$ . Given  $b \in \beta$ , there is a unique  $j$  such that  $j \circ b$  is consistent with  $P \cup c_1$ . Call it  $b^-$ . Given an  $m$ -ary sequence  $s$ , define the boolean  $C_1[s]$  that holds when  $s = C_1^m \circ C_1[1 : j]$  for some  $j \in \{1, \dots, |C_1|\}$  and some  $m \in \mathbb{N}$ . If  $C_1[s]$  holds, let  $rem(s)$  denote the unique  $j$  for which it holds. Define

$$J(B) = \max\{J \mid C_1[B(\star(B) - J : \star(B) - 1)] \text{ holds}\}$$

and  $rem(B) = rem(B(\star(B) - J) : \star(B) - 1))$ . Also define

$$L(B) = \max\{L \mid C_1[B(\star(B) : \star(B) + L) \text{ holds}]\}.$$

Define the rule  $F_2$  as shown in algorithm 2.

---

**Algorithm 2:** Part of the Local Patching Algorithm,  $F_2$ .

---

```

if  $B(-2r - 1 : -1) \in \alpha$  then
  |  $F_2(B) = B(-2r_1 : -1)^+$ 
else if  $B(1 : 2r + 1) \in \beta$  then
  |  $F_2(B) = (B(1 : 2r + 1))^-$ 
else if  $\neg C_1[B(\star(B) - J(B) : \star(B))]$  then
  | if  $J(B) > 0$  then
    | if  $((J(B) \leq L(B)) \wedge (L(B) < R))$  then
      |  $F_2(B) = C_1(1 + mod(rem(B)), |C_1|)$ 
    | else
      |  $F_2(B) = B(0)$ 
    | end
  | else
    |  $F_2(B) = C_1(1)$ 
  | end
else
  |  $F_2(B) = B(0)$ 
end

```

---

Finally, define  $\tilde{F}$  by

$$\tilde{F}(B) = \begin{cases} F_1(B), & \text{if } B \in \mathcal{B} \\ F_2(B), & \text{otherwise} \end{cases}.$$