Self-stabilizing synchronization in 3 dimensions (draft)

Matthew Cook
cook@ini.phys.ethz.ch
ETH Zurich

Peter Gács
gacs@bu.edu
Boston University

Erik Winfree
winfree@centrosome.dna.caltech.edu
California Institute of Technology

October 20, 2008

Abstract

The simplest known fault-tolerant model of computation is the three-dimensional cellular automaton introduced by Gács and Reif (based on Toom's two-dimensional nonergodic automaton). However, this automaton works only in discrete time (that is, if synchronization is provided for free); the only known asynchronous reliable cellular automata are vastly more complex. Surprisingly, a simple scheme is known to introduce asynchrony into any kind of network: each cell keeps track of which neighbors are ahead or behind (at most one step difference is allowed), for example via a mod 3 clock. This mechanism is very efficient in the absence of errors, or if the errors leave the local ordering in a consistent state (no cycles). But errors could create topological inconsistency in spacetime which, unless resolved quickly, may become fatal. We found a simple way to resolve localized topological inconsistencies of arbitrary size. The present paper shows that the scheme is self-stabilizing: it corrects inconsistency of an arbitrary size fast, in the absence of further errors.

1 Introduction

In a parallel computation model, a useful distinction is made between synchronous and asynchrononous modes of operation. In the synchronous case, the system evolves in discrete time: at each of the time steps $0, 1, 2, \dots$, each part of the system is updated according to a local transition function. In an asynchronous model, the order in which different parts of the system are updated is not deterministic. In the most natural example, each part is updated at a sequence of random times that form a Poisson process (the system is a continuous-time Markov process). In many problems of distributed computing, on the other hand, it is customary to assume that the order of updating is chosen by an adversary. How different are the capabilities of synchronous and asynchronous automata?

Some computations are naturally "asynchronous", in the sense that the order of updating does not matter. For a precise formulation of this property, see [3]. The present paper is motivated, however, by a particular asynchronous computation, which is very sensitive to the update order. This is the 3-dimensional cellular automaton introduced in [6], that simulates reliably a 1-dimensional cellular automaton. We will describe this automaton below and its sensitivity will be evident.

Perhaps surprisingly, a simple, general local scheme is known to simulate any kind of synchronous network $\mathbf{A}$ (not only cellular automata) by an asynchronous network $\widetilde{\mathbf{A}}$ with the same set of sites (but possibly larger neighborhood and state space). We will describe this scheme below: essentially, the property is maintained that the number of the simulated step differs by at most one between neighbor sites. For this purpose, each site stores the mod 3 remainder of the step number of the simulated computation (as well as the previous local state), and never progresses until it has a neighbor whose step number is smaller (having the step number modulo 3 is sufficient to check this). Let us call the scheme the *modulo 3 trick*. The waiting slows the computation, but it has been shown in [1] that the slowdown is at most by a constant factor.

The present paper is devoted to the attempt of applying the modulo 3 synchronization trick to the Gács-Reif-Toom automaton, with appropriate adaptations in the algorithm and the proof. A mechanical application cannot be expected to work automatically, for two different reasons:

1. Even if the global slowdown is only by a constant factor, if local slowdowns are excessive then they will get in the way of the error-correction mechanism, which assumes that progress is made even locally with sufficient speed.

2. The errors will also mess up the mod 3 clocks, introducing inconsistencies (deadlocks).

Problem 1 seems the less interesting one, since according to all experience and intuition, a local slowdown should occur only with probability exponentially small as a function of its size. Nevertheless, this problem has not been solved yet rigorously.

Problem 2 is more interesting. It does not occur in a 1-dimensional cellular automaton. In a 2-dimensional array of clocks, loops of circular waiting always include "point-defects": points with non-zero "winding number". Ideas for getting the poles with positive and negative winding number to find and annihilate each other did not seem too promising. In the late 1980's, Charles Bennett has experimented with several such rules.

Bennett also noted that in a 3-dimensional array of clocks, the topological defects are not signed points but oriented lines. Since errors are bounded, and there is a conservation of the influx and outflux of these lines at each point, the defects consist of (possibly intersecting) loops. The present paper is based on ideas of Cook and Winfree who developed a rule that makes these loops shrink and disappear, at least in the absence of new errors.

They noticed a similarity between the loop elimination problem and the original problem of getting 2-dimensional regions of error to disappear. In Toom's solution of that problem, each cell takes a majority of itself, its north neighbor, and its east neighbor. But here we are pushing loops around in three dimensions; there is no clear interior or exterior of the loop, and the adjustments needed to erase them might extend far beyond them.

Imagine pulling each defective edge into the interior of an individual cell. Portions of such a cell would have different clock values, not consistent with any absolute time, so we could say such a cell has a clock value of "$*$" instead of "0", "1", or "2". We can easily have a rule that if you are a cell with a defective edge, then you become a $*$. This translates the topological problem to the need to get rid of the $*$s.

Naively, we might want to change a clump of $*$s to have a constant clock value, so that there can be no defects. Before changing them all, we would want the surface of the clump to be at a constant value, so that it would be safe to replace all the $*$s with this value. We can get the surface to be at a constant value by just letting the CA advance the clocks in the normal way (this assumes that the advance is not held back by the presence of a "long steep slope" of clock values, leading back to Problem 1 above), but only allowing a clock on the surface to advance if there is a neighbor clock on the surface who is ahead of it. So all the clocks on the surface are limited by the most advanced clock on the surface, with which they will all synchronize.

Advancing the surface to a constant value can be impossible if the $*$s are in the shape of a ring, so that integrating "delta time" along a loop on the surface might not give zero, in which case there is no "most advanced" clock. To avoid this sort of problem, the $*$s will need to grow so as to become solid clumps. We will make sure that their growing and shrinking do not interfere with each other in a way that prevents progress.

Here is an outline of the rest of the paper. Section 2 introduces our models formally. Section 3 introduces the modulo 3 trick. Section 4 introduces the Gács-Reif-Toom model and formally states the present paper's main theorem about its synchronization. Section 5 develops the discrete-topological toolbox to be used in the proof. Section 6 defines the transition function for the main theorem (presented as a combination of several "rules"). Section 7 proves the main theorem. It uses the tools developed in Section 5 as well the "blame sequence" idea from [1].

2 Asynchrony

Here, we develop the framework of asynchronous computation. We will restrict attention to cellular automata, though most definitions relating to update order have wider applicability.

2.1 Cellular automata

Let us define cellular automata in the form that we will use them.

Definition 2.1 (Sites)**.** To be more specific, let us call the elementary parts of the system *sites*, or *cells*, or *processors* (we will use these terms interchangeably). The set of sites is denoted by $\mathbb{C}$. Each site x has a set $\mathbb{S}$ of possible states (also called "local states"). An arbitrary function $\xi \in \mathbb{S}^{\mathbb{C}}$ is called a *space-configuration*, or simply "configuration", or "global state". The value $\xi(x)$ is the state of site x in ξ. ⌟

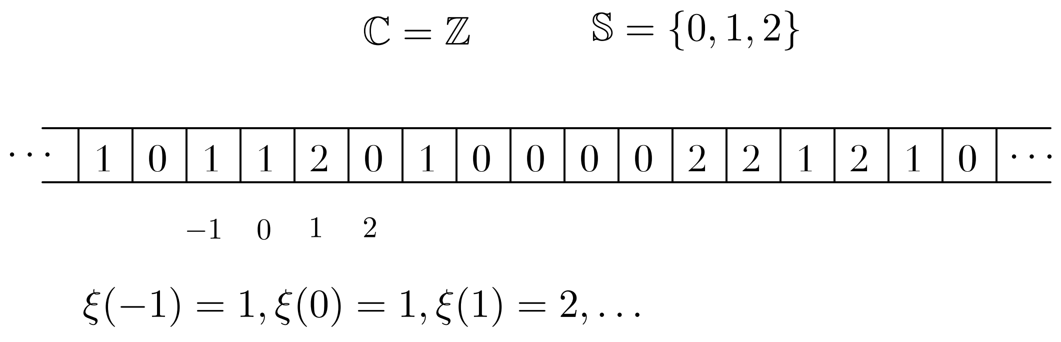

$\mathbb{C} = \mathbb{Z}$
$\mathbb{S} = \{0,1,2\}$
$\cdots$ 1 0 1 1 2 0 1 0 0 0 0 2 2 1 2 1 0 $\cdots$
-1 0 1 2
$\xi(-1) = 1, \xi(0) = 1, \xi(1) = 2, \ldots$

Figure 1: The (space-)configuration of a one-dimensional cellular automaton

Definition 2.2 (Neighborhood)**.** As is the case with typical computations models, the evolution is determined by local transitions. A function $N : \mathbb{C} \to 2^{\mathbb{C}}$ will be called a *neighborhood function* assigning to each $x \in \mathbb{C}$, a finite set $N(x) \subseteq \mathbb{C}$ called the *neighborhood* of x.

$$x \in N(x),$$

and symmetric:

$$x \in N(y) \Rightarrow y \in N(x).$$

The *neighbor relation graph* $G_N = (\mathbb{C}, E)$ is an undirected graph over the set of sites in which two sites are connected by an edge if they are neighbors. ⌟

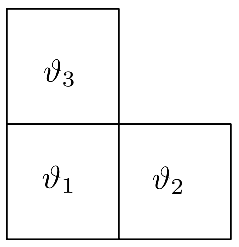


Figure 2: The Toom neighborhood

Example 2.1. If $\mathbb{C} = \mathbb{Z}^d$ then the *von Neumann neighborhood* is the set of points differing from x by at most 1, in at most one coordinate, and the *Moore neighborhood* is the set of points differing from x by at most 1, in any coordinate. ⌟

Definition 2.3 (Transition)**.** A function $f : \mathbb{S}^{\mathbb{C}} \to \mathbb{S}^{\mathbb{C}}$ is called a *transition function* if $f(\xi)(x)$ depends only on $\xi \upharpoonright N(x)$, that is

$$\xi_1 \upharpoonright N(x) = \xi_2 \upharpoonright N(x) \Rightarrow f(\xi_1)(x) = f(\xi_2)(x).$$

The transition function locally determines a possible "next" configuration from the "current" one. Our system is then determined by the 4-tuple, can be written as

$$\mathrm{Aut}(\mathbb{C},\mathbb{S},N(\cdot),f(\cdot))$$

and will be called an *automaton* (not necessarily a finite one).

We restrict our models to the case where each neighborhood $N(x)$ has the same cardinality and can be listed in some *fixed order* as

$$N(x) = (\vartheta_1(x),\dots,\vartheta_r(x)).$$

A (deterministic, synchronous) *cellular automaton* is thus given as

$$\mathbf{A} = \mathrm{CA}(\mathbb{C},\mathbb{S},r,\vartheta,g)$$

where g is the *local transition function* determining the global transition function f by

$$f(\xi)(x) = g(\xi(\vartheta_1(x)),\dots,\xi(\vartheta_r(x))).$$

In our cellular automata, the set $\mathbb{C}$ of sites will allways have the structure of a commutative group (for example, the d-dimensional lattice $\mathbb{Z}^d$), and the neighborhood function will also be homogeneous:

$$\vartheta_i(x) = x + \vartheta_i(\mathbf{0}),$$

where $\mathbf{0}$ is the unit of the group, say $(0,0)$ for $\mathbb{C} = \mathbb{Z}^2$. ⌟

Example 2.2. A one-dimensional cellular automaton. Let $\mathbb{C} = \mathbb{Z}$, the set of integers, $N(x) = \{x-1,x,x+1\}$. In a configuration, each site x has a value $\xi(x) \in \mathbb{S}$. There is a local transition function $g : \mathbb{S}^3 \to \mathbb{S}$. The result of transition at site x is

$$f(\xi)(x) = g(\xi(x-1),\xi(x),\xi(x+1)).$$

⌟

Example 2.3 (The Toom Rule). Let $\mathbb{C} = \mathbb{Z}^2$, $\mathbb{S} = \{0,1\}$,

$$\begin{aligned}(\vartheta_1(\mathbf{0}),\vartheta_2(\mathbf{0}),\vartheta_3(\mathbf{0})) &= ((0,0),(0,1),(1,0)),\\ g(x,y,z) &= \mathrm{Maj}(x,y,z).\end{aligned}$$

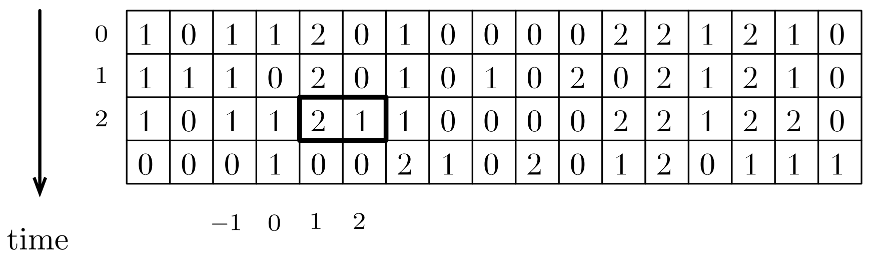
Thus, every time a cell updates its state, its new state is the majority of itself, its northern and its eastern neighbor (see Figure 2). ⌟

Definition 2.4 (Space-time configuration)**.** The evolution of our system is described by a function

$$\eta(x,t)$$

called the *space-time configuration*. Here, x runs over the set $\mathbb{C}$ of sites, and t runs over the set of possible times. (When the value of the time argument is implicitly clear from the context, then we may delete the time argument from the notation.) The domain of the time variable can be the set $\mathbb{Z}_+$ of positive integers, or the set $\mathbb{R}_+$ of positive real numbers. In the first case, we will talk about a discrete-time model, in the second case, about a continuous-time model.

In the discrete model, a space-time configuration can also be viewed as a sequence $\eta : \mathbb{Z}_+ \to \mathbb{S}^{\mathbb{C}}$ of space-configurations. ⌟



$$\eta(1,2) = 2, \eta(2,2) = 1, \ldots$$

Figure 3: Space-time configuration of a one-dimensional cellular automaton

Definition 2.5 (Synchronous trajectory)**.** We will say that a space-time configuration η is a *synchronous trajectory* of the automaton Aut if for all x,t we have $\eta(\cdot,t+1) = f(\eta(\cdot,t))$. In other words,

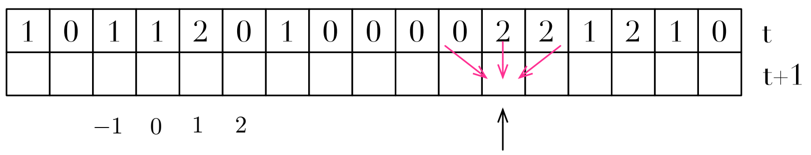
$$\eta(x,t) = f(\eta(\cdot,t-1))(x),$$

that is in η, each site is “updated” every time by the function f (though the update might not change the state).

More generally, for cellular automata it is a trajectory over the space-time area $E \subseteq \mathbb{C} \times \mathbb{Z}_+$ if for all $(x,t) \in E$ such that $N(x) \times \{t-1\} \subseteq E$ we have

$$\eta(x,t+1) = g(\eta(\vartheta_1(x),t),\ldots,\eta(\vartheta_r(x),t)). \tag{2.1}$$

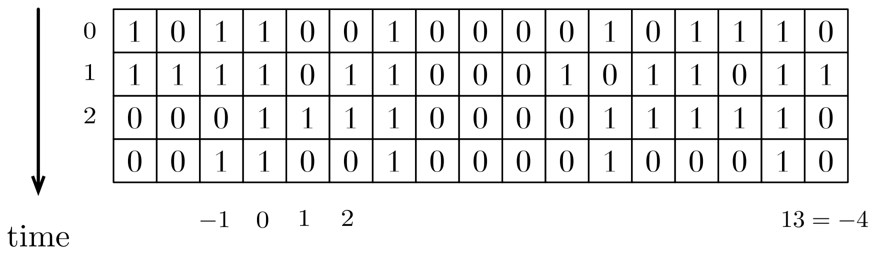
$\lrcorner$



$$\eta(x,t+1) = g(0,2,2)$$

Figure 4: Trajectory of a one-dimensional cellular automaton with transition function $g(\cdot,\cdot,\cdot)$.

Thus, given a deterministic, synchronous cellular automaton, an evolution is completely determined by its initial configuration $\eta(\cdot,0)$.



0	1	0	1	1	0	0	1	0	0	0	0	1	0	1	1	1	0
1	1	1	1	1	0	1	1	0	0	0	1	0	1	1	0	1	1
2	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	0
	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	1	0
			-1	0	1	2											$13=-4$

Figure 5: Trajectory of Wolfram's rule 110 on $\mathbb{Z}/(17\mathbb{Z})$. The rule says: "If your right neighbor is 1 and the neighborhood state is not 111 then your next state is 1, otherwise 0".

2.2 Asynchronous updating

We are interested in situations when time is continuous, and at any one time, only some of the sites are updated (typically, only one). In Example 2.2, we would have

$$\eta(x,t) = g(\eta(x-1,t-\varepsilon),\eta(x,t-\varepsilon),\eta(x+1,t-\varepsilon)), \tag{2.2}$$

if there is an $\varepsilon = \varepsilon(x,t)$ such that the neighborhood $N(x)$ does not change during $[t-\varepsilon,t)$. (In our examples, there will always be such an ε.)

For mere convenience, we introduce the following property.

Definition 2.6 (Time marking)**.** A transition function f has the *time marking property* if it always changes the state of the cell it is applied to, that is we have

$$f(\eta(\cdot,t))(x) \neq \eta(x,t).$$

⌟

Definition 2.7 (Update times)**.** Let us call the function $\tau : \mathbb{C}\times\mathbb{Z}_+ \to \mathbb{R}_+$ a *system of update times*

$$(\tau(x,n) : x\in\mathbb{C},\ n=0,1,2,\dots),$$

$\tau(x,0)=0$, $\tau(x,n)<\tau(x,n+1)$, and

$$\lim_n \tau(x,n) = \infty \tag{2.3}$$

for each x. We say to a space-time configuration $\eta(x,t)$ *belongs* the system of update times $\tau(x,n)$ if each cell x can change its state only at times $t=\tau(x,n)$, so $\eta(x,t)$, as a function of time, is constant on the half-closed intervals $[\tau(x,n),\tau(x,n+1))$. ⌟

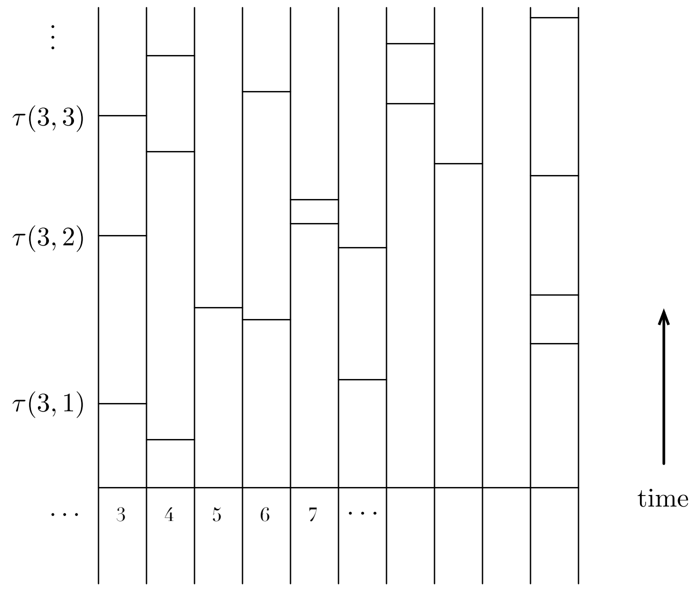


Figure 6: Updating in continuous time

Definition 2.8 (Update events)**.** Let us call any space-time point (x,t) with $t = \tau(x,n)$ for some n an *update event*. The set of update events is denoted by

$$\mathscr{U} = \{\,(x,\tau(x,n)) : x \in \mathbb{C},\ n = 1,2,\dots\}.$$

An *update time* is a time that is the update time of any cell. Let r be the size of the neighborhoods, $i \leqslant r$. We define the space-time neighbor function

$$\Theta_i^{\mathscr{U}} : \mathscr{U} \to \mathscr{U}$$

as follows.

$$\begin{aligned} \underline{\tau}(x,t) &= \max_{\tau(x,k)<t} \tau(x,k), \\ \Theta_i^{\mathscr{U}}(z) &= (\vartheta_i(x), \underline{\tau}(\vartheta_i(x),t)). \end{aligned}$$

Thus, $\Theta_i^{\mathscr{U}}(z)$ is the event at which neighbor cell $\vartheta_i(x)$ obtained the state influencing z. The set of events

$$\hat{N}(z) = \{\,\Theta_i^{\mathscr{U}}(z) : i = 1,\dots,r\}$$

is called the set of *space-time neighbors* of the point z. ⌟

Remarks 2.4. Note the following.

– Note that the space-time neighbor relation is *not reflexive*.

– The function $\hat{N}(z)$ can be defined just in terms of the function $N(x)$, without using the sequence $\vartheta(x)$.

⌟

Definition 2.9 (Update graph)**.** The space-time neighbor relation is directed backward in time, and turns the set $\mathscr{U}$ of update events into a directed graph

$$\mathscr{G} = (\mathscr{U}, \{ (z,z') : z \in \mathscr{U}, z' \in \hat{N}(u) \}) \tag{2.4}$$

called the *update graph*. It is *well-funded* if it contains no infinite directed path. ⌟

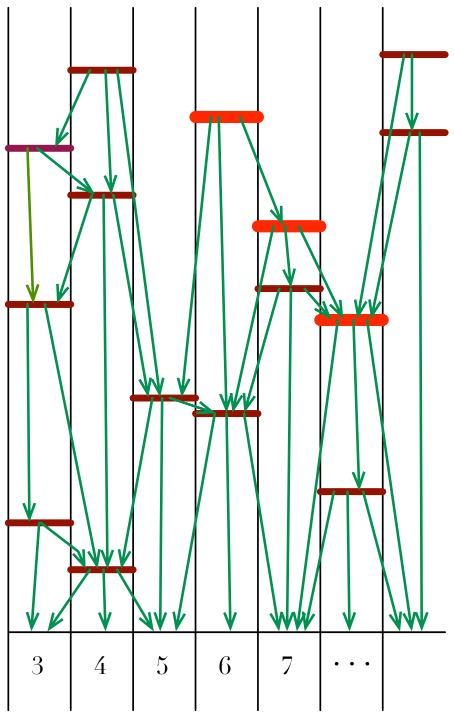


Figure 7: Update graph

Well-fundedness does not hold, for example, when $\mathbb{C} = \mathbb{Z}$, and $\tau(x,n) = n + 1/(1+|x|)$. But, it holds with probability 1 in the models we will study, in which the set $\mathscr{U}$ is random.

Definition 2.10 (Asynchronous trajectory)**.** We will say that $\eta(x,t)$ is an *asynchronous trajectory* for a well-funded set of update events $\mathscr{U}$ if instead of equation (2.1) we require, in the spirit of our intentions (2.2), the condition

$$\eta(z) = g(\eta(\Theta_1^{\mathscr{U}}(z)), \dots, \eta(\Theta_r^{\mathscr{U}}(z))), \tag{2.5}$$

(where we have identified $\eta(x,t)$ with $\eta((x,t))$).

We will say that $\eta(x,t)$ is an *asynchronous trajectory* if it is an asynchronous trajectory for some well-funded set of update events $\mathscr{U}$. ⌟

The following is immediate, and is the reason for introducing the time marking property.

Proposition 2.5. *If an automaton has the time marking property and η is an asynchronous trajectory for the sets of update times $\mathscr{U}_1$ and $\mathscr{U}_2$ then $\mathscr{U}_1 = \mathscr{U}_2$.*

From now on, for asynchronous purposes we will always use automata that have the time marking property.

If the graph $\mathscr{G}$ is well-funded then in an asynchronous trajectory η, the initial configuration $\eta(\cdot, 0)$ determines η uniquely. The sets of update times we will consider will always be well-funded. They will also have the following convenient property:

Condition 2.1. If $(x_1, t_1), (x_2, t_2) \in \mathscr{U}$ then $t_1 \neq t_2$. ⌟

This is only for convenience: at the expense of some extra case distinctions, it could be avoided throughout. Anyway, it holds with probability 1 in the random update model introduced below.

2.3 Random updating

Definition 2.11. For $m = 2, 3 \dots$, let $\mathbb{Z}_m$ be the set of remainders modulo m. From now on, assume that the site space $\mathbb{C}$ is the commutative group of form $\mathbb{C}_1 \times \cdots \times \mathbb{C}_d$ where each $\mathbb{C}_i$ is either $\mathbb{Z}$ or $\mathbb{Z}_m$ for some m. In other words, our space is either a finite-dimensional lattice or the product of a finite-dimensional torus and a finite-dimensional lattice. ⌟

In a probabilistic cellular automaton (synchronous or asynchronous), the space-time configuration $\eta(x, t)$ is a stochastic process. We will restrict ourselves to the simplest, asynchronous (continuous-time) case.

Definition 2.12. We will say that the random process is a *trajectory* of the *continuous-time probabilistic cellular automaton* (sometimes called *interacting particle system*)

$$\mathbf{A} = \mathrm{CPCA}(\mathbb{C}, \mathbb{S}, r, \vartheta, g) \tag{2.6}$$

if the (random) update set $\mathscr{U}$ has the following properties.

(a) The different sequences $(\tau(x, n) : n = 0, 1, 2, \dots)$ for $x \in \mathbb{C}$ are independent of each other.

(b) The sequence of increments $\tau(x, n+1) - \tau(x, n)$ is independent.

(c) Each variable $\tau(x, n+1) - \tau(x, n)$ has the same exponential distribution with rate 1:

$$\mathbf{P}[\tau(n+1, x) - \tau(n, x) > t] = e^{-t}.$$

⌟

Remark 2.6. Properties (b) and (c) together say that for each x, the sequence $(\tau(x, n) : n > 0)$ is a Poisson process with rate 1. A process η satisfying these conditions along with the update equation (2.5) is a continuous-time Markov process. The precise statement and the proof of this fact can be found in any standard treatment of interacting particle systems, like [8]. ⌟

It is easy to see that a trajectory of a continuous-time cellular automaton satisfies the requirement (2.3) and Condition 2.1 with probability 1. It is also well-funded. Let us formulate a complementary, related statement.

Definition 2.13. Consider a sequence of space-time points $w_0, w_1, \dots, w_n$ going backwards in time with $w_i = (u_i, t_i)$ in which u_{i+1} is a neighbor of u_i. This sequence will be called a *forward blame sequence* if t_i is the first update time of u_i after t_{i+1}. It will be called a *backward blame sequence* if t_{i+1} is the last update time of u_{i+1} before t_i. The difference $t_0 - t_n$ is the *time span* of the blame sequence, and n is its *length*. A *blame sequence* is a forward or backward blame sequence. ⌟

Proposition 2.7. *Let* **A** *be a continuous-time probabilistic cellular automaton given in* (2.6)*. There are constants* $\gamma, \delta > 0$ *such that the following holds. For all* $n > 0$*, for all space-time points* (x,t)*, the probability that a blame sequence of length* $\leqslant n$ *and time span* $\geqslant \gamma n$ *starts at* (x,t) *is less than* $e^{-\delta n}$.

3 Simulating a synchronous computation by an asynchronous one

3.1 Local clock differences

Our main concern will be introduced later: there are situations, where the timing of many parallel updates becomes important. We will review later our main example, the fault-tolerant 3-dimensional cellular automaton of [6].

Definition 3.1 (Asynchronous simulation)**.** Consider two automata, $\mathbf{A} = \mathrm{Aut}(\mathbb{C}, \mathbb{S}, \vartheta(\cdot), g(\cdot))$ and $\widetilde{\mathbf{A}} = \mathrm{Aut}(\mathbb{C}, \widetilde{\mathbb{S}}, \vartheta(\cdot), \widetilde{g}(\cdot))$, where $\widetilde{\mathbf{A}}$ has the time-marking property.

These define the neighborhood function $N(x)$. An *asynchronous simulation* of $\mathbf{A}$ by $\widetilde{\mathbf{A}}$ is a tuple $(\mathbf{A}, \widetilde{\mathbf{A}}, \phi, \Psi)$ where ϕ, Ψ are mappings with the following properties.

(a) Let ξ be an arbitrary space configuration of $\mathbf{A}$. Then $\phi(\xi)$ is a space configuration of $\widetilde{\mathbf{A}}$.

(b) Let η be an arbitrary asynchronous trajectory of $\widetilde{\mathbf{A}}$ with $\eta(\cdot, 0) = \phi(\xi)$. Then $\Psi(\eta)$ is a synchronous trajectory ζ of $\mathbf{A}$ with $\zeta(\cdot, 0) = \xi$.

⌟

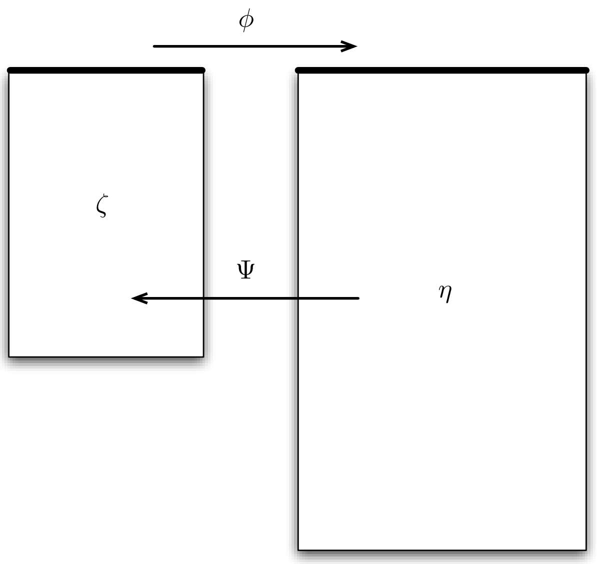


Figure 8: A simulation (ϕ,Ψ) of ζ by η. Encoding the input by ϕ, decoding the process by Ψ.

In what follows we restrict ourselves to a very special kind of asynchronous simulation.

Notation 3.2. Denote the remainder of x with smallest absolute value modulo m by

$$x \text{ amod } m = (x + \lfloor m/2 \rfloor) \bmod m - \lfloor m/2 \rfloor.$$

⌟

Definition 3.3 (Clocked extension)**.** For a set of local states $\mathbb{S}$, we will say that the set $\widetilde{\mathbb{S}}$ is its *clocked extension* if there is a finite set $\mathbb{S}_{\text{Clock}}$ with $\mathbb{Z}_3 \cup \{*\} \subset \mathbb{S}_{\text{Clock}}$ such that

$$\widetilde{\mathbb{S}} = \mathbb{S}^2 \times \mathbb{S}_{\text{Clock}}.$$

If $s = (u,v,w) \in \widetilde{\mathbb{S}}$ then we will write

$$u = s.\text{Cur},\ v = s.\text{Prev},\ w = s.\text{Clock}.$$

For an arbitrary space-time configuration η with states from $\widetilde{\mathbb{S}}$ whenever $x,y \in \mathbb{C}$ and $\eta(x,t).\text{Clock}, \eta(y,t).\text{Clock} \in \mathbb{Z}_3$ we define, for space-time configuration η and space configuration ξ:

$$\begin{aligned}\Delta(u,v) &= (v.\text{Clock} - u.\text{Clock}) \text{ amod } 3,\\ \Delta^{\xi}(x,y) &= \Delta(\xi(x),\xi(y)),\\ \Delta^{\eta}(x,y,t) &= \Delta(\eta(x,t),\eta(y,t)).\end{aligned}$$

When a fixed time is implicitly clear from the context, then we may delete the time argument in the notation for η. ⌟

The local asynchronous simulation will have a clocked extension as its state space. If $\widetilde{g}$ is its local transition function, this will have to satisfy some conditions which we present here in terms of *rules*. Suppose that the transition rule changes the state $s = \eta(z)$ with $z = (x,t)$ to some state $\overline{s}$, further $s.\mathrm{Clock} \in \mathbb{Z}^3$.

The first rule says that the clock does not decrease, and it does not increase either if there is a neighbor that would be "left behind".

Rule 3.1 (Wait)**.** We have

(a) $(\overline{s}.\mathrm{Clock} - s.\mathrm{Clock})$ amod $3 \neq -1$, that is the clock will not "decrease".

(b) If z has a neighbor $z' \in \hat{N}(z)$ with state $s' = \eta(z')$ and with $s'.\mathrm{Clock} \in \mathbb{Z}_3$, $\Delta(s,s') < 0$ then $\overline{s} = s$.

⌟

Such rules will be called, informally, to obey the *marching soldiers* scheme.

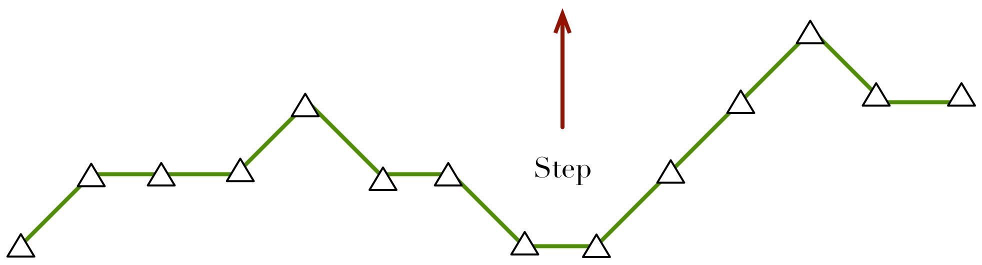


Figure 9: Marching soldiers, not leaving any neighbor behind

The following rule performs the actual simulation. For its definition, for a space-time point z let

$$\begin{aligned} s_i &= \eta(\Theta_i^{\mathscr{U}}(z)), \\ q_i &= \begin{cases} s_i.\mathrm{Cur} & \text{if } \Delta(s,s_i) = 0, \\ s_i.\mathrm{Prev} & \text{if } \Delta(s,s_i) = 1, \end{cases} \\ \mathrm{Trans}^{\eta}(z) &= g(q_1,\dots,q_r). \end{aligned}$$

Rule 3.2 (Emulate)**.** If the states s' in all neighbors have $s'.\mathrm{Clock} \in \mathbb{Z}_3$ and $\Delta(s,s') \geqslant 0$, then set

$$\begin{aligned} \overline{s}.\mathrm{Cur} &:= \mathrm{Trans}^{\eta}(z), \\ \overline{s}.\mathrm{Prev} &:= s.\mathrm{Cur}, \\ \overline{s}.\mathrm{Clock} &:= s.\mathrm{Clock} + 1 \bmod 3. \end{aligned}$$

⌟

It is easy to see that both rules can be expressed as just a property of the transition function $\widetilde{g}$.

Definition 3.4 (Clocked simulation)**.** Consider two automata, $\mathbf{A} = \mathrm{Aut}(\mathbb{C}, \mathbb{S}, \vartheta(\cdot), g(\cdot))$ and $\widetilde{\mathbf{A}} = \mathrm{Aut}(\mathbb{C}, \widetilde{\mathbb{S}}, \vartheta(\cdot), \widetilde{g}(\cdot))$ where $\widetilde{\mathbf{A}}$ has the time marking property. We will say that the pair $(\mathbf{A}, \widetilde{\mathbf{A}})$ is a *clocked simulation* of $\mathbf{A}$ by $\widetilde{\mathbf{A}}$ if the following conditions hold.

1. $\widetilde{\mathbb{S}}$ is a clocked extension of $\mathbb{S}$.

2. If $\xi' = \phi(\xi)$ then we have $\xi'.\mathrm{Cur} = \xi$, $\xi'.\mathrm{Clock} = 0$.

3. The transition function $\widetilde{g}$ obeys the Wait and Emulate rules.

⌟

Now it is not difficult to prove the following proposition:

Proposition 3.1 (Clocked asynchronous simulation)**.** *For a clocked simulation a pair of mappings ϕ, Ψ can be defined turning it into an asynchronous simulation* $(\mathbf{A}, \widetilde{\mathbf{A}}, \phi, \Psi)$.

Later we have to deal with space-time configurations that are not perfect asynchronous trajectories. To characterize their imperfections we will be more formal in describing the sense in which clocks must be consistent.

Definition 3.5 (Consistency)**.** Let η be a space-time configuration. Let a connected, undirected graph $G = (D, E)$ be defined on a (finite or infinite) set D of space-time points. Consider a path $P = (v_0, \dots, v_n)$, in G. If $v_n = v_0$ then we call the path *closed*; a closed path is also called a *loop*. In the notation

$$P = \mathrm{loop}(v_0, \dots, v_{n-1})$$

the repeated element is omitted: $P = (v_0, \dots, v_{n-1}, v_0)$. For a given space-time configuration η let us define

$$\mathrm{lag}^{\eta}(P) = \sum_{i=0}^{n-1} \Delta(\eta(v_i), \eta(v_{i+1})). \tag{3.1}$$

The space-time configuration η is *consistent* on the graph G if the "integral" $\mathrm{lag}^{\eta}(P)$ of every closed path P in D is 0.

The same concepts will also be defined for sets and configurations just in space, not in space-time. ⌟

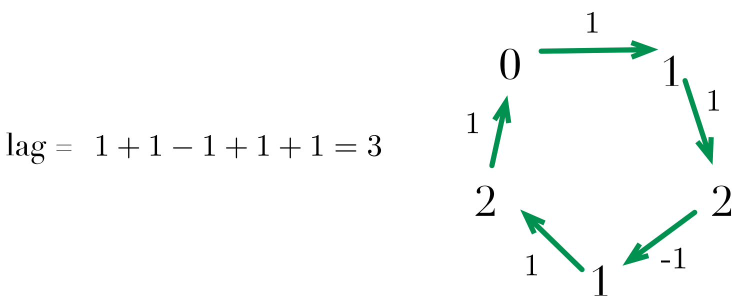


Figure 10: The lag of a loop

The following fact is straightforward to verify.

Proposition 3.2. *Let $G = (D, E)$ be a graph of space-time points as above. Then η is consistent on G if and only if there is a function $\mathrm{Step}(\cdot) : D \to \mathbb{Z}$ such that for all $v \in B$ we have*

$$\eta(v).\mathrm{Clock} = \mathrm{Step}(v) \bmod 3.$$

When exists, the function $\mathrm{Step}(v)$ is determined uniquely to within an additive constant.

Definition 3.6 (Space-time neighbor graph)**.** Let D be a set of space-time points and η a space-time configuration. We define a graph $G = (D, E)$ on it by introducing the following two kinds of edges.

1. Between $(x,t), (y,t) \in D$ if x, y are neighbors. These will be called *horizontal edges*.
2. Between $(x,t), (x,u)$ if the set $\{x\} \times [t,u] \subseteq D$ and $\eta(x,t')$ changes value at most once between t and u. These will be called *vertical edges*.

We say that η is *consistent* on D if it is consistent on G.

If D has the form $D = B \times \{t\}$ then we will say that η is consistent on the set of sites B at time t. ⌟

In practice, to construct the function $\mathrm{Step}(v)$ we will need to show that for every edge (u,v) in the graph G we have

$$\Delta(\eta(u), \eta(v)) = \mathrm{Step}(v) - \mathrm{Step}(u). \tag{3.2}$$

The following is an extension of Proposition 3.1 when the space-time configuration is restricted to a subset of space-time.

Proposition 3.3. *Let $(\mathbf{A}, \widetilde{\mathbf{A}})$ be a clocked simulation. Let $\zeta(x,p)$ be a synchronous trajectory of automaton $\mathbf{A}$ over $\mathbb{C}$. Let $B \subset \mathbb{C}$ be a set of sites and $[t_0, t_1)$ a time interval. Let η be a trajectory of automaton $\widetilde{\mathbf{A}}$ over $B \times [t_0, t_1)$ that is consistent over B at time t_0. Then it is also consistent on $B \times [t_0, t_1)$.*

If, in addition, we have $t_0 = 0$, $B = \mathbb{C}$, and $\eta(x,t).\mathrm{Cur} = \zeta(x,0)$ for all $x \in \mathbb{C}$, then the step function Step can be made unique by requiring $\mathrm{Step}(x,0) = 0$, and then the relation

$$\eta(x,t) = \zeta(x, \mathrm{Step}(x,t)) \tag{3.3}$$

defines implicitly ζ in a unique way for $x \in \mathbb{C}$, $t < t_1$.

3.2 The slowdown

Since the update rule of the asynchronous simulation includes now update attempts in which nothing happens, the question arises: what is the price in slowdown? In other words, how slowly can the function $\mathrm{Step}(x,t)$ grow in Proposition 3.3? For the random updating model introduced in 2.3 above, it has been shown in [1] that the slowdown is at most by a constant factor. Here is a precise formulation of the result.

Proposition 3.4. *For every cellular automaton* $\mathbf{A}$ *there are constants* $c, d > 0$ *with the following property. Let* $\zeta(x,t)$ *be a computation of a synchronous automaton* $\mathbf{A}$ *over* $\mathbb{C}$ *with transition function* $g(s_1, \dots, s_k)$*. Let* $\widetilde{\mathbf{A}}$ *be the corresponding continuous-time cellular automaton whose transition function obeys the* Emulate *rule, and let the random process* η *be a trajectory of* $\mathbf{A}$ *with* $\eta(x,0) = \zeta(x,0)$ *for all* $x \in \mathbb{C}$*. Then, with the unique function* $\mathrm{Step}^{\eta}(x,t)$ *that exists due to Proposition 3.3, for each* x, t *we have*

$$\mathbf{P}[\,\mathrm{Step}^{\eta}(x,t) < ct\,] < e^{-dt}.$$

The proof goes by constructing a blame sequence and using Proposition 2.7.

4 Fault-tolerance

4.1 The problem

The object of the present paper is to combine asychrony with fault-tolerance. The construction of (theoretically) reliable computers from unreliable components has an interesting history, starting with von Neumann's work [10]. The question is particularly natural and challenging for the model of cellular automata, since the homogeneity does not allow error-correction capabilities frozen into the structure. In this setting, even the modest goal of keeping a single bit of information required a nontrivial solution: it is achieved in 2 dimensions by Toom's rule (see Example 2.3). The proof that this model has the desired error-correcting property is quite complex (even though the main ideas are intuitive and simple). For the discrete-time (synchronous) case, the proof can be found in [9]; for continuous time, in [7]. (For simplified versions of the same proof, see [1, 4].) The same task, as well as the task of performing arbitrary computation, can also be accomplished on one-dimensional cellular automata. However, the transition rules (and the corresponding proofs), given in [2] and [5], are very complex.

4.2 A 3-dimensional fault-tolerant synchronous automaton

The simplest known fault-tolerant computation model is the three-dimensional cellular automaton introduced in [6].

Definition 4.1 (3-dimensional fault-tolerant simulation)**.** Let $\mathbf{U}$ be an arbitrary 1-dimensional cellular automaton (since it is arbitrary, it might as well be chosen computationally universal). In the initial configuration of a 3-dimensional automaton

$$\mathbf{U}' \tag{4.1}$$

constructed from $\mathbf{U}$, we slice the space into planes by the value of the first coordinate. Every cell with coordinates x, y, z will have the initial state of cell x of automaton $\mathbf{U}$. The transition rule of $\mathbf{U}'$ goes as follows. In each step, first perform Toom's Rule (extended naturally to larger alphabets) within each plane, and then the transition rule of $\mathbf{U}$ across the planes. ⌟

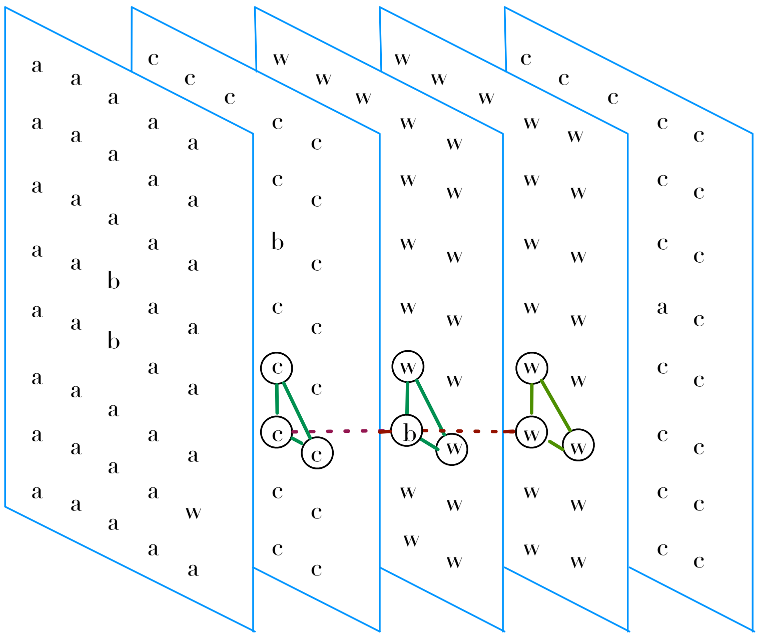


Figure 11: Three-dimensional fault-tolerant simulation of a one-dimensional cellular automaton

In what sense is this computation fault-tolerant? Consider a random process $\eta(u,t)$ ($u \in \mathbb{Z}^3$, $t \in \mathbb{Z}_+$) that follows the transition rule $\mathbf{U}'$ only approximately: at each space-time point (u,t), the transition rule is applied but then, with some probability $< \varepsilon$, a *fault* occurs, and the actual value $\eta(u,t)$ becomes something else. We assume that faults occur independently of each other. The paper [6] proves the following theorem.

Proposition 4.1. *There is a constant c with the following property. Let $\zeta(x,t)$ be a computation (space-time configuration) of the deterministic 1-dimensional cellular automaton $\mathbf{U}$, and let $\eta(u,t)$ be a space-time configuration of the 3-dimensional cellular automaton $\mathbf{U}'$ with noise bound ε, such that for all x,y,z we have $\eta((x,y,z),0) = \zeta(x,0)$. Then for all $x,y,z \in \mathbb{Z}$, $t \in \mathbb{Z}_+$ we have*

$$\mathbf{P}[\eta((x,y,z),t) \neq \zeta(x,t)] \leqslant c\varepsilon.$$

The proof is essentially the same as the proof mentioned above, that Toom's Rule remembers one bit. (Some extra sophistication is needed, even in the formulation of the result, for finite spaces.)

4.3 An asynchronous version of the 3-dimensional automaton

The 3-dimensional fault-tolerant cellular automaton is attractively simple, but has a peculiar limitation: it works only in discrete time (that is, if synchronization is provided for free). Indeed, the Toom Rule corrects errors by maintaining the property that in each computation step t, the value $\eta(u,t)$ is constant (except for the effect of faults) as u runs through any plane $\{x\} \times \mathbb{Z}^2$. This value will typically change in the whole plane, as we move from t to $t+1$. Such a coordinated change depends on synchrony. The 1-dimensional cellular automaton in [5] is asynchronous, but it is vastly more complex.

The present work is the first step in our project to save the simplicity of the 3-dimensional construction in an asynchronous setting, via the mechanism developed in Section 3. What is the challenge?

As mentioned in the introduction, one problem is that random updating by itself may introduce delays, by creating a long queue of cell waiting for each other.

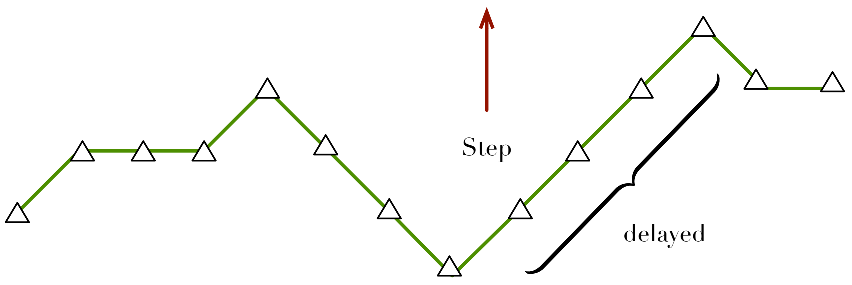


Figure 12: Delay introduced by a long slope (illustrated in 1 dimension). While there is no advance, the system is “rotting”: new and new faults occur, while the old ones are not being corrected.

If we want to use the asynchronous simulation in noise then this simulation itself must survive noise. The new element is the mod 3 clocks. Noise can change their values, but the threat is not the change in the values in itself: rather, that the errors can introduce inconsistency into the clock values, in the sense of Definition 3.5. Indeed, though the Wait rule guarantees that a cycle with nonzero lag will not appear, errors could just create it. For example, let sites $u_1, \dots, u_6 \in \mathbb{C} = \mathbb{Z}^2$ be defined by

$$u_1 = (0,0),\ u_2 = (1,0),\ u_3 = (0,2),\ u_4 = (2,1), u_5 = (1,1),\ u_6 = (1,0),$$

then errors could give them the clock values $0,1,2,0,1,2$. When the neighborhood is the von Neumann neighborhood then these cells form a loop. The Wait rule will not allow them to proceed in the computations: they are deadlocked, waiting for each other forever.

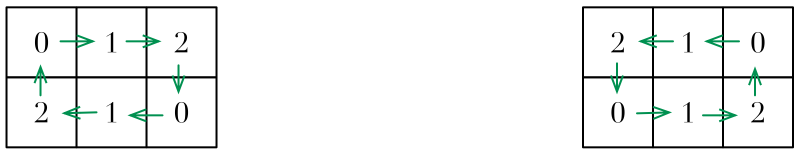


Figure 13: Two opposite small loops can be far from each other, and everything else may seem normal locally.

Remark 4.2. Inconsistency is impossible on a tree and, in particular, in 1 dimension, since every pair of points is connected by only one simple path. ⌟

The challenge is to come up with some simple additions to the rules Wait and Emulate that "fight" inconsistencies. What does this mean?

First, it will mean the so-called *self-stabilization* property: even if the clocks are messed up in a large but finite area by noise, if a noise-free period can be quaranteed after that, then our rules will restore consistency, in reasonable time (linear in the size of the mess).

Second, we hope that the rules created for self-stabilization will actually achieve more: a continuous-time version of the 3-dimensional fault-tolerant automaton. There are ample resons to believe that our rules achieve this goal, and we will outline the next steps needed for the proof. Since we are far from this latter goal yet, we will not even define completely formally the notion of an asynchronous fault-tolerant computation here. Approximately, we want a noisy version of Proposition 3.3:

> Let $\zeta(x,p)$ be a computation of the 3-dimensional automaton $\mathbf{U}'$ of (4.1). Let $\eta(x,t)$ be a space-time configuration of the asynchronous automaton $\widetilde{\mathbf{U}}'$ (with the appropriate rule that includes Wait and Emulate), such that $\eta(x,0) = \zeta(x,0)$ for all $x \in \mathbb{C}$. Then there is a (random) space-time set $D(\eta)$ such for every site x and time t we have $\mathbf{P}[\, x \in D^t(\eta)\,] > 2/3$, and a step function Step^η over D satisfying (3.3) for all $(x,t) \in D$.

Let us formulate the self-stabilization theorem, the main result of the present paper. This theorem says that with large probability, out rule will clear a finite domain B of any clock inconsistency, in time linear in the size of B. But there is an assumption: that at the start of the correction, around B, there is no extremely long queue of cells delaying each other.

Definition 4.2. Let $\widetilde{\mathbf{A}}$ be a clocked simulation in 2- or 3-dimensional space, and ξ a space configuration of ξ. A *defect* of ξ is a circular path of length 4 (a square) with nonzero lag. ⌟

Definition 4.3. For integers $L < G$ and a site v_0, let us define the cube and "ball"

$$\Lambda(L) = \mathbb{Z}^3 \cap [0,L]^3, \quad \Gamma(G) = \{\, u \in \mathbb{Z}^3 : |u| \leqslant G \,\}.$$

We clearly have $|\Lambda(L)| = (L+1)^3$. The volume of the convex hull of $\Gamma(G)$ in $\mathbb{R}^3$ is $\frac{8}{3}G^3$, and the number of sites in it is bounded by $\frac{8}{3}(G+1)^3$, so for sufficiently large L,G we certainly have

$$|\Lambda(L)| \leqslant 2L^3, \quad |\Gamma(G)| \leqslant 3G^3. \tag{4.2}$$

We say that a space-configuration ξ of an automaton with a modulo 3 Clock field satisfies the condition

$$\mathrm{Slack}(L,G)$$

at site v_0 if the following holds for $\Lambda = v_0 + \Lambda(L)$, $\Gamma = v_0 + \Gamma(G+6L)$.

(a) All defects of ξ are contained in the interior of the cube Λ. Therefore (as we will see) ξ is consistent on $\Gamma \setminus \Lambda$, and an absolute clock value $\mathrm{Step}(u)$ can be defined for its cells, uniquely to within an additive constant, with $\mathrm{Clock}^\xi(u) = \mathrm{Step}(u) \bmod 3$.

(b) For $u,v \in \Gamma \setminus \Lambda$, $|u-v| \leqslant G+3L$ we have $\mathrm{Step}(u) - \mathrm{Step}(v) \leqslant G$.

We say that a space-time configuration η satisfies the condition $\mathrm{Slack}(L,G)$ at time T_0 at site v_0 if $\eta(\cdot,T_0)$ satisfies it. ⌟

The $\mathrm{Slack}(L,G)$ condition will be applied typically with G chosen much larger than L. Its part (b) says, in essence, that there are no "long, steep slopes": the complement of the defects does not have a path $(u_0,u_1,\dots,u_{G+3L})$ along which the $\mathrm{Step}(u_i,T_0)$ is decreasing over all but L edges. Such a path could prevent $\mathrm{Step}(u_0,t)$ from making L steps of progress, for an amount of time proportional to G. In other words, the chain into which the Step values of neighbors are forced is pulled too "tight". So, the $\mathrm{Slack}(L,G)$ condition mandates some slackness in these chains over all long paths. We believe that this condition is violated only with a probability exponentially small in G; however, we cannot prove this yet.

Theorem 4.1. *There are constants $c_1,c_2,d>0$ with the following properties.*

Let $\mathbf{A}$ be an arbitrary 3-dimensional synchronous cellular automaton. There is a corresponding continous-time cellular automaton $\widetilde{\mathbf{A}}$ obeying the rules Wait *and* Emulate*, such that the following holds.*

Let site v_0, time T_0 and number $L>0$ be given, with

$$G > L, \quad T_1 = T_0 + c_1 L + c_2 G, \quad \Lambda = v_0 + \Lambda(L), \quad \Gamma = v_0 + \Gamma(G+6L).$$

Let the stochastic process η be a trajectory of $\widetilde{\mathbf{A}}$ in the set $\Gamma \times [T_0,T_1]$, satisfying the $\mathrm{Slack}(L,G)$ *condition at time T_0 for site v_0. Then with probability $> 1-e^{-dL}$, there is a step function over*

$$\Gamma \times [T_0,T_1] \setminus \Lambda \times [T_0,T_1).$$

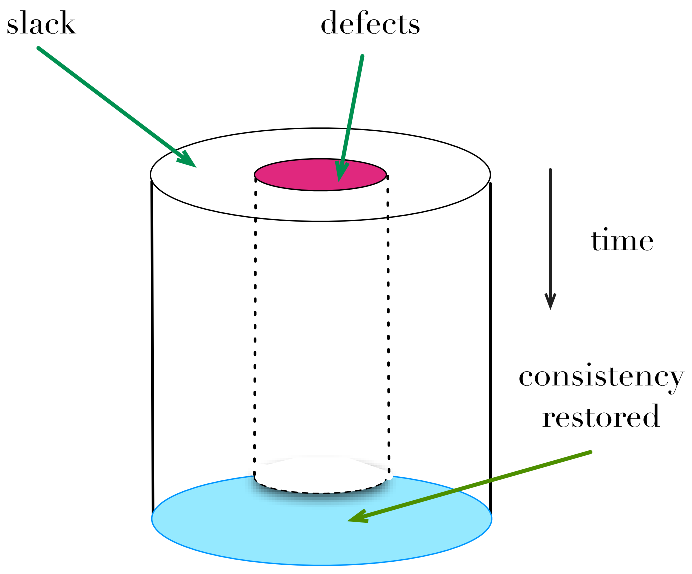


Figure 14: The main theorem

In other words the consistency of the clocks, possibly disturbed inside Λ at time T_0, will be restored by time T_1. The theorem does not mention the requirement (3.3). For this requirement to hold, we would have to take $\mathbf{A} = \mathbf{U}'$, with the fault-tolerant synchronous cellular automaton $\mathbf{U}'$ introduced above. And instead of the set $\Lambda \times [T_0, T_1]$, we will have to cut out a (constant times) larger piece of space-time $\Lambda' \times [T_0, T_1']$: the errors that arose at time T_0 and propagated during $[T_0, T_1]$ to a larger cube Λ', will then be corrected by the time T_1' via the fault-tolerant behavior of $\mathbf{U}'$.

5 The topology of defects

Let us fix some time t, which we will omit from the notation now. Consider a fixed space configuration $\eta(\cdot, t)$. We can denote then the lag of any path P, as defined in (3.1), by $\text{lag}(P) = \text{lag}^{\eta}(P, t)$. Every closed path with nonzero lag contains a closed path with nonzero lag that is simple (does not intersect itself).

5.1 Two-dimensional case

In the 2-dimensional case, it is known that every simple path divides the plane into an inside and outside domain. Without loss of generality, we will only consider simple paths that go around the inside domain in the clockwise direction. If the length of the path is 4 then it surrounds a square. We will call the middle point of this square a "point defect" if the lag of this path is nonzero. The lag of a point defect is the lag of the surrounding path: it can only take the values 3 and -3 (positive and negative defects).

If the length of path P is not 4 then the inside domain can be divided into two nonempty parts by a path, and for the closed paths P_1, P_2 surrounding them it is true that $\text{lag}(P) = \text{lag}(P_1) + \text{lag}(P_2)$. Continuing the subdivision process we find that $\text{lag}(P)$ is the sum of the lags of the defects surrounded by P.

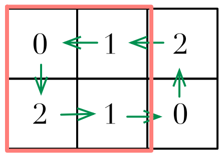


Figure 15: Each loop of nonzero lag contains a defect.

If we start from a consistent configuration and change it locally within some large square Λ then the lag of any path around Λ is still 0, therefore the number of positive defects in Λ equals the number of negative defects. One can say that "defects come in pairs". For example, we could have just two defects arbitrarily far from each other (within Λ). Intuitively it seems that a simple local rule is needed to get the $+3$ poles and -3 poles to find (and "annihilate") each other. Ideas proposed for this did not seem too promising.

5.2 Three-dimensional case: some elementary topology

Three-dimensional clocks will be much more amenable to error-correction; the reason is that while in two dimensions defects can be separated from each other, in three dimensions they "form loops". Let us make this precise.

Definition 5.1. A *corner cube* for $\mathbb{Z}^3$ is a cube of the form $(x,y,z)+\{0,1\}^3$. ⌟

Definition 5.2. Given a cell $u=(x,y,z)\in\mathbb{C}=\mathbb{Z}^3$, we denote its *height* in the $(1,1,1)$ direction by

$$h(u)=x+y+z.$$

Given a set S, let

$$\|S\|$$

be the size of the smallest cube enclosing it. A path of neighbor cells is called *rising* if the height $h(u)$ is increasing on it. ⌟

It is convenient to generalize the notion of path of neighbor cells introduced in Definition 3.5, and to embed paths into the algebraic structure of a commutative ring over the integers.

Definition 5.3. Let $A_{\mathbb{C}}$ be the set of formal sums

$$c_1e_1+\cdots+c_ne_n$$

where c_i are integers called the *weights* and e_i are directed edges between neighbors in $\mathbb{C}$, modulo the relations $0\cdot e=0=$ the empty sum, and $(u,v)+(v,u)=0$. We can always make the weights positive by reversing the edges.

A path $P=(u_1,\dots,u_n)$ corresponds to the element $(u_1,u_2)+\cdots+(u_{n-1},u_n)$ of this ring, and $-P$ corresponds to the reverse path. Elements of the subring $L_{\mathbb{C}}$ of $A_{\mathbb{C}}$ generated by closed paths will be called *chains*.

Suppose that a configuration ξ is given for which the function $\Delta^{\xi}(x,y)$ is defined that is used in the definition of lags. That definition of lags, given in Definition 3.5, can be extended easily to chains via $\mathrm{lag}^{\xi}(\sum_i P_i)=\sum_i \mathrm{lag}^{\xi}(P_i)$. ⌟

It is easy to see the following.

Proposition 5.1. *An element p of $A_{\mathbb{C}}$ is a chain if and only if for every point u of $\mathbb{C}$, the sum of weights on edges in p leaving u is 0. (In other words, the number—with multiplicity—of ingoing edges is equal to the number of outgoing edges.)*

Let us approximate the notion of a continuous transformation of closed paths.

Definition 5.4. A loop of four points going around one of the faces of a corner cube will be called a *plaquette*. Note that every cube face has two plaquettes associated with it that are the negatives of each other. In a configuration ξ for which $\Delta^{\xi}(x,y)$ is defined, a *defect* is a plaquette with nonzero lag. (Note that the lag of a plaquette is always one of the numbers $-3,0,3$.)

Let $B\subset\mathbb{C}$ be a set of sites. We will say that two chains P,Q in B are *equivalent* in B if there is a set of plaquettes $p_1,\dots,p_n$ in B such that $Q-P=\sum_i p_i$. We will say that a connected set of sites B is *simply connected* if every chain in it is equivalent to 0. ⌟

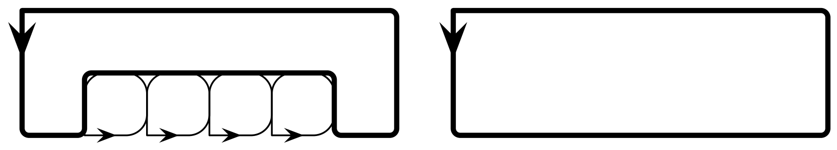


Figure 16: Equivalent loops: their difference is a sum of plaquettes.

The following statement is easy to verify.

Proposition 5.2. *Consider a configuration ξ for which $\Delta^{\xi}(x,y)$ is defined. Let B be a set of sites that is defect-free in ξ. Then for every pair P,Q of equivalent closed paths in B, we have* $\mathrm{lag}^{\xi}(P) = \mathrm{lag}^{\xi}(Q)$.

Corollary 5.3. *Consider a configuration ξ for which $\Delta^{\xi}(x,y)$ is defined. Let $B \subset \mathbb{C}$ be a simply connected set of sites that is defect-free in ξ. Then ξ is consistent in B.*

Proof. Indeed, every closed path is equivalent to an empty chain, so its lag is 0. □

Proposition 5.4. *A cube B is simply connected.*

Proof. Let C be an arbitrary chain: we will show that it is equivalent to 0. Let $u_1 = (x,y,z)$ be a point adjacent to a nonzero edge of C for which the height $h(u_1)$ defined above is lowest. Without loss of generality we can assume that $u_1 = (0,0,0)$, and for $u_0 = (0,1,0)$ and $u_2 = (1,0,0)$, the chain contains weighted edges $c \cdot (u_0,u_1)$ and $d \cdot (u_1,u_2)$ with coefficients $c \geqslant d > 0$. Let $u_3 = (1,1,0)$. We can then transform C into an equivalent chain by adding the plaquette $d \cdot \mathrm{loop}(u_2,u_1,u_0,u_3)$. The new chain is still in the cube B, but the transformation decreased the absolute value of the coefficients of the edges leaving the point u_1, without increasing any such coefficients for any other lowest point. Continuing these transformations will turn the chain to 0. □

This implies a role for three-dimensional defects similar to the two-dimensional defects: in a cube, inconsistency implies defects. If we start from a consistent configuration and change it locally within some large cube Λ then the lag of any closed path outside Λ is still 0. If inconsistency arose then now Λ contains defects. It turns out that defects are themselves organized into a certain dual chain.

Definition 5.5. For the lattice $\mathbb{C} = \mathbb{Z}^3$, let us define the *dual lattice*

$$\mathbb{C}' = (1/2,1/2,1/2) + \mathbb{Z}^3,$$

whose sites are the centers of the corner cubes. The neighbor relation on this lattice will again be defined by the von Neumann neighborhood. For our purposes it is useful to imagine the cells of the original lattice as not points but cubes, whose corners are the elements of the dual lattice. These will be called *site cubes* in contrast to the corner cubes.

Let us set up a one-to-one correspondence between plaquettes of $\mathbb{C}$ and directed edges of $\mathbb{C}'$. Each plaquette p is on the common face of two corner cubes. We assign to p an edge $E(p)$

connecting the centers of these cubes. It is directed in such a way that, looking in its direction, the points of p are listed clockwise. We will call $E(p)$ the *plaquette vector* corresponding to plaquette p. In the graphical representation where site cubes are assigned to cells, the plaquette vector is a directed version of the common edge of all the site cubes of the sites of the plaquette.

We define the ring $A_{\mathbb{C}'}$ of formal sums $\sum_i c_i E_i$ of plaquette vectors just as we defined formal sums of edges. Again, such a formal sum is a *chain* if it is the linear combination of closed paths. We will call them *dual chains*.

Consider a configuration ξ for which $\Delta^{\xi}(x,y)$ is defined. The *lag* of a plaquette vector is the lag of the corresponding plaquette. If a plaquette is a defect then its plaquette vector will also be called a *defect vector*. ⌟

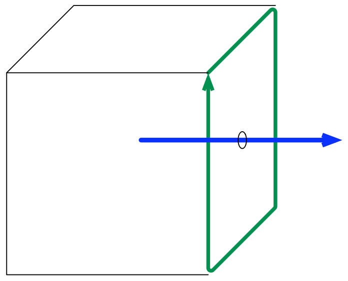


Figure 17: A defect vector

Now we are in a position to give a precise form to the statement that "defects form loops". The following lemma is easy to verify.

Lemma 5.5. *Consider a configuration ξ for which $\Delta^{\xi}(x,y)$ is defined. Let B be a corner cube, and let $E_1,\dots,E_6$ be the plaquette vectors directed across each face of B, towards the center. Then* $\sum_i \operatorname{lag}(E_i) = 0$.

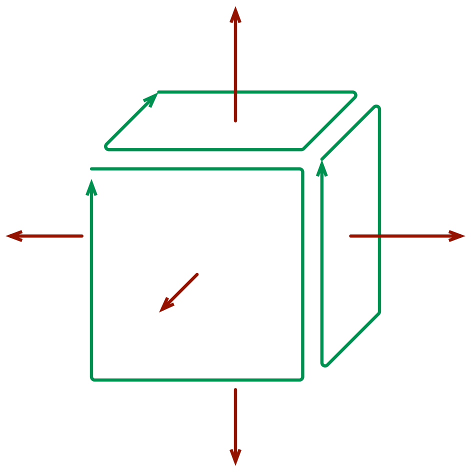


Figure 18: The sum of defect vectors on the faces of a cube is 0.

Proposition 5.6. *Consider a configuration ξ for which $\Delta^{\xi}(x,y)$ is defined. Let $B \subset \mathbb{C}$ be a cube such that ξ is defect-free outside B. Let $E_1,\dots,E_n$ be the defect vectors. Then the formal sum $\sum_i \operatorname{lag}(E_i)\cdot E_i$ is a dual chain that is the union of some disjoint closed dual paths, each taken with multiplicity 3.*

We will call any of these closed dual paths a *defect loop*.

Proof. According to Proposition 5.1, a formal sum p is a chain if and only if for every point u of $\mathbb{C}'$, the sum of weights on edges in p leaving u is 0. Lemma 5.5 shows that our formal sum satisfies this condition.

We have seen that a chain can always be viewed as just a number of closed paths overlaid over each other: if an edge appears in several of the paths, it is taken with multiplicity. However, the lag of a plaquette, if positive, can only be 3. $\square$

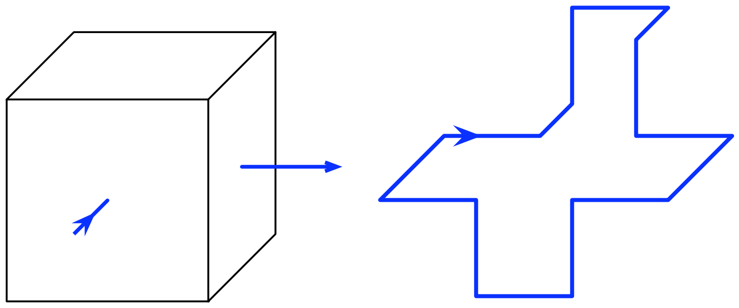


Figure 19: If a defect vector enters a corner cube, another one must leave it. So, if a set of defects is bounded, it is made up of closed paths.

6 The transition rule

From this section on, we set $\mathbb{C} = \mathbb{Z}^3$, with the von Neumann neighborhood.

6.1 Motivation

Our goal is to formulate rules that shrink away a defect loop. This seems similar to the original problem of getting 2-dimensional regions of error to disappear, solved by the Toom rule: each cell takes a majority of itself, its north neighbor, and its east neighbor. But here we are pushing a defect loop (or more) around in three dimensions; there is no clear interior or exterior of the loop, and the adjustments needed to erase them might extend far beyond them.

Defects themselves are easy to get rid of, by a rule that if you are a cell that is part of a defect, then you become a $*$. It is, however, wrong to think that by just killing the bearer of the bad news, the bad news itself goes away. Indeed, the complement of the $*$s may very well be inconsistent itself. Corollary 5.3 implies consistency in the absence of defects only in simply connected sets. If the $*$s are in the shape of a ring then their complement is not simply connected. Indeed, this complement may be inconsistent: it may contain a closed loop with nonzero lag linked through this ring. To avoid this sort of problem, the $*$s will grow so as to fill in any holes.

This leaves us with the problem of how to get rid of the $*$s. Without knowing any of the details it is hard to imagine how to replace the $*$s with consistent clock values. A simple idea is to strive for a constant clock value: this would guarantee the absence of defects. We can get the surface to be at a constant value by just letting the Emulate rule advance the clocks in the normal way, but only allowing a clock on the surface to advance if there is a neighbor clock on the surface who is ahead of it. So all the clocks on the surface are limited by the most advanced clock on the surface, with which they will all synchronize.

We will make sure that growing and shrinking do not interfere with each other in a way that prevents progress.

Below, we define a rule with a proof that errors will be fixed *if* there are no further errors. We will look at the effects of errors later, under various conditions. At first sight, further errors do not appear to cause a setback in the process beyond roughly adding their own size to what needs to be fixed.

6.2 Definitions for the rule

We will sometimes think of sites in terms of site cubes introduced in Definition 5.5.

Definition 6.1. Two neighbor sites share a face of their site cube, therefore they will be called *face-adjacent*. Two sites that are neighbors but participate in a common plaquette share an edge of their site cubes, so they will be called *edge-adjacent*. Two sites sharing a vertex but not an edge of their site cubes are called *vertex-adjacent*.

Our distance measure in the cell space will be the “Manhattan distance” defined by the L_1 norm:

$$|(x,y,z)| = |x| + |y| + |z|, \quad d(u,v) = |u - v|.$$

For a set S and a site u we will write, as usual:

$$d(u,S) = \max_{v \in S} d(u,v).$$

⌟

Definition 6.2. A defect vector is also called a *hot edge*. In configuration ξ, the the set of cells with value $*$s will be denoted by $\mathrm{Mess}(\xi)$. We will also write

$$\mathrm{Mess}(t) = \mathrm{Mess}(\eta(\cdot,t)).$$

A $(1,1,1)$ *corner block* for a cell (x,y,z) is the corner cube $(x,y,z) + (1,1,1) - \{0,1\}^3$. Similarly for $(-1,1,-1)$ corner block, and so on.

A *corner square*, of a point u is a square of size 1 in one of the coordinate planes, with one corner being u. ⌟

6.3 The transition rule

We write the transition rule as the union of several non-contradictory subrules. The rules Wait and Emulate have already been defined above, in a more general setting, since they are not dependent on the lattice $\mathbb{Z}^3$.

Rule 6.1 (Form)**.** If you participate in a defect then become a $*$. $\lrcorner$

Rule 6.2 (Swell)**.** If you are not a $*$, but there is a $*$ in each of the corner squares containing $(0,1,1)$, $(1,1,0)$ and $(1,0,1)$, then become a $*$. $\lrcorner$

Here are some other ways of formulating this rule. If you are not a $*$ then turn into a $*$ if one of the following two, equivalent conditions holds:

– You cannot be separated from the $*$s in the $(1,1,1)$ corner block by a plane parallel to one of the coordinate planes.

– "Toom's rule" on $*$s in some plane across $(0,0,0)$ turns you into $*$, or the three corners $(0,1,1)$, etc. are all $*$s.

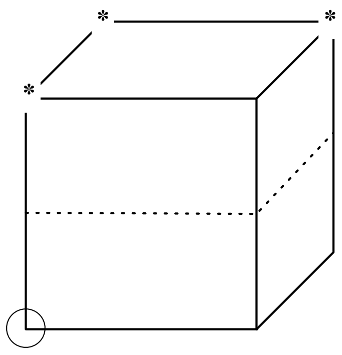


Figure 20: Swell Rule: If you cannot be separated from the Mess in the $(1,1,1)$ corner block by a plane parallel to one of the coordinate planes, then become a $*$.

Definition 6.3. Given a set H of points, the set $\mathrm{Swell}(H)$ is the set of $*$s obtained by first putting $*$s into H and then applying the Swell rule repeatedly as long as it creates new $*$s. Thus, $\mathrm{Swell}(\mathrm{Swell}(H)) = \mathrm{Swell}(H)$. A set H is called *saturated* if $\mathrm{Swell}(H) = H$. $\lrcorner$

The following statement is a straightforward consequence of the definitions.

Proposition 6.1. Form *and* Swell *are the only rules creating $*$s.* Swell *is the only rule creating $*$s in the absence of defects. For a set H, the set* $\mathrm{Swell}(H)$ *never exceeds the bounding box of H.*

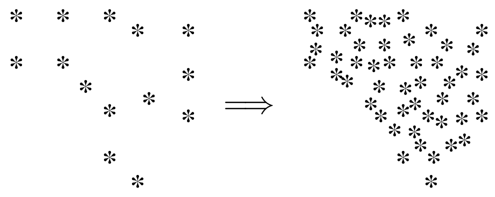


Figure 21: The Swell rules fills holes in the Mess, without letting it grow beyond an enclosing cube

Proposition 6.2. *Let H be a saturated set, and let B be a cube containing H at a distance > 1 in its interior. Then $B \setminus H$ is simply connected. Moreover, any two points in $B \setminus H$ are connected by a path of size $O(\|B\|)$.*

Proof. We will talk of the elements of H as the $*$s. Let us first show that $B \setminus H$ is connected. Let w be the point of B with maximum height $h(v)$. Let us show that every point u_1 in $B \setminus H$ is connected to w by a rising path. This also shows that any two points in $B \setminus H$ are connected by a path of size $O(\|B\|)$. Let us start a rising path from u_1 and continue it as long as we can. If we could not then we would hit a point u_i such that all three rising edges from u_i end in a $*$. This is, however, excluded by the saturatedness of H, since then the Swell rule would turn u_i into a $*$. So, the path continues until it hits a site in the outermost layer of the cube B. Then it can slide inside this layer while still increasing the height until it hits w.

The proof that $B \setminus H$ is also simply connected, has a structure similar to the proof of Proposition 5.4.

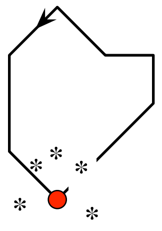


Figure 22: The complement of the saturated Mess is simply connected: if a loop cannot be contracted in it, its stuck bottom would have been turned into a star by the Swell rule.

Let C be an arbitrary chain: we will show that it is equivalent to 0. Let $u_1 = (x, y, z)$ be a point adjacent to a nonzero edge of C for which the height $h(u_1)$ is lowest. Without loss of generality we can assume that $u_1 = (0,0,0)$, and for $u_0 = (0,1,0)$ and $u_2 = (1,0,0)$, the chain contains weighted edges $c \cdot (u_0, u_1)$ and $d \cdot (u_1, u_2)$ with coefficients $c \geqslant d > 0$. If $(1,1,0)$ is not a $*$ we can proceed as in the proof of Proposition 5.4; suppose therefore that it is a $*$.

For transparency, let us write 101 for $(1,0,1)$, and so on. The point 101 cannot be $*$, since then the Swell rule would turn 100 into $*$. Similarly, 011 and 001 cannot be $*$. Then we can replace C

with an equivalent chain by adding

$$d \cdot \mathrm{loop}(100,000,001,101) + d \cdot \mathrm{loop}(000,010,011,001).$$

The rest of the proof continues like the one for Proposition 5.4. □

Definition 6.4. The definition here refers to a particular set S. A *surface neighbor of an element of S* is a cell in the complement of S which is face-adjacent or edge-adjacent to it. It is a *strong* surface neighbor if it is face-adjacent to it. The *surface* of S is the set of its surface neighbors. ⌟

Lemma 6.3. *Let w be an element of the saturated set S, with no higher neighbors in S. Any two strong surface neighbors of w are connected by a path of cells all of which are surface neighbors of w.*

Proof. Let $H_>$ be the set of the 6 surface neighbors u of w having $h(u) > h(w) =: L$. It is easy to see that $H_>$ is connected.

Elements of the set $H_=$ of the 6 potential surface neighbors u of w with $h(u) = L$ are already connected to $H_>$.

Let u belong to the set $H_<$ of the 6 potential surface neighbors v of w with $h(v) < L$. Then the u is connected to two neighbors above it (in terms of $h(\cdot)$). If both of these are in S then the Swell rule would add u to S, which is impossible since S is saturated. □

Rule 6.3 (Shrivel)**.** Suppose that you are a $*$ with no higher neighbors that are $*$s and all your non-$*$ neighbors have the same clock value. Then change to this common clock value. ⌟

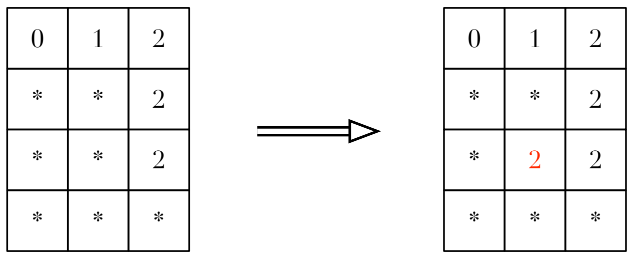


Figure 23: The Shrivel rule

Clearly, a $*$ turned into a non-$*$ by the Shrivel rule will not be immediately turned back into a $*$.

To help the Shrivel rule, we will try to bring all cells on the *surface* (appropriately defined) of the Mess to a common clock value. It helps that in the consistent environment the Step values are not static in time: they keep growing as far as they can. Therefore we only need to adjust *upwards*.

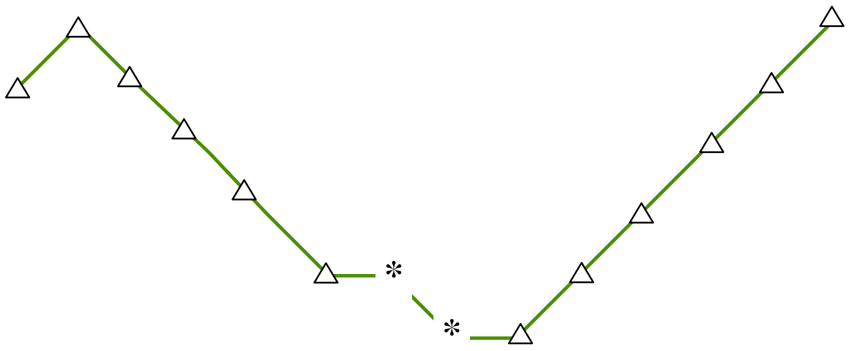


Figure 24: If the broad environment is pulling up the clock values, this helps bringing the surface of the Mess to a common value.

The following rule will help bring closer to each other the maximum and minimum times, on the surface of any simply connected island of $*$s.

We use the following notation:

Notation 6.5. Let $z = (x,t)$, and

$$t-_x = \max\{\, \tau(y,n) : y \in N(x),\ \tau(y,n) < t \,\}$$

(exists because of (2.3)). ⌟

Rule 6.4 (Synchronize)**.** Let x be the point whom we are updating at time t, with $c = \eta(x,t-_x)$.Clock. Suppose that, at time $t-_x$, all neighbors of x have Clock $\in \{*, c, c+1 \bmod 3\}$. Suppose also that both x and one of its neighbors y are surface neighbors of a common $*$, with $\eta(y,t-_x)$.Clock $= c+1 \bmod 3$. Then $\eta(x,t)$.Clock $:= c+1$. ⌟

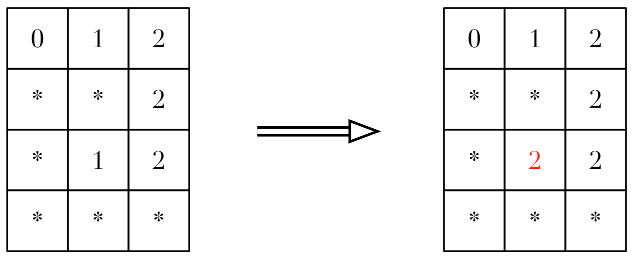


Figure 25: The Synchronize rule

The following proposition is an immediate consequence of the rules.

Proposition 6.4. *Suppose that no errors are made.*

(a) *Suppose that at some time, the* Form *rule is not applicable. Then it will not be applicable at later times either.*

(b) *The same is true about the* Swell *rule.*

(c) *Suppose that at some time, the complement of* Mess *is consistent. Then it will stay consistent for all later times.*

7 Proof of Theorem 4.1

The constants c_1, c_2, d will be found by the end of the proof. Let $\widetilde{\mathbf{A}}$ be the continuous-time probabilistic cellular automaton whose transition funtion satisfies all our subrules: Wait, Emulate, Form, Swell, Shrivel, Syncronize. Assume that v_0, T_0, L, G, T_1, Λ, Γ are defined as in the statement of the theorem, and let the stochastic process η be a trajectory of $\widetilde{\mathbf{A}}$ in the set $\Gamma \times [T_0, T_1]$, satisfying the $\mathrm{Slack}(L, G)$ condition at time T_0 for site v_0.

It follows from Proposition 3.3 that a function $\mathrm{Step}(x,t)$ unique to within an additive constant can be defined for (x,t) in $(\Gamma \setminus \Lambda) \times [T_0, T_1]$. The proof of the theorem will be completed if we show that all defects and $*$s disappear from $\Gamma \times [T_0, T_1]$ before time T_1. Indeed, this will mean that η is consistent at time T_1 over the set Γ, and hence $\mathrm{Step}(\cdot, T_1)$ can be extended uniquely from $\Gamma \setminus \Lambda$ to Γ.

All milestones $T_{\mathrm{Form}}, T_{\mathrm{Swell}}, \dots$ defined below depend on η, so they are random; we could indicate it by writing T^{η}_{Form}, but most of the time we will not.

Definition 7.1. We define some milestones.

– Let $T_{\mathrm{Form}} = T^{\eta}_{\mathrm{Form}}$ be the first time after which there are no defects (hot edges). According to Proposition 6.4, this condition will then hold for all later times.

– Let T_{Swell} be the first time following T_{Form} after which the Swell rule will no longer be applicable. According to Lemma 6.2, for all times t after T_{Swell}, the complement of $\mathrm{Mess}(t)$ is consistent. We will see in Lemma 7.1 below that $\mathrm{Step}(\cdot, t)$ can be extended uniquely to this set.

– Let $M = \max_{u \in \Lambda} \mathrm{Step}(u, T_{\mathrm{Swell}})$. Let T_{Pull} be the first time t after T_{Swell} after which $\mathrm{Step}(v,t) \geqslant M$ for all cells v with $d(v, \Lambda) = 3L$. Since the area outside Λ stays consistent and since the only rule that can change Step there is Emulate, this condition will continue to hold for all later times.

– Let T_{Sync} be the first time after T_{Pull} after which, for all cells $u \in \mathrm{Mess}(t)$ without higher neighbors in $\mathrm{Mess}(t)$, the function $\mathrm{Step}(\cdot, t)$ is constant over the surface neighbors of u.

– Let T_{Done} be the first time after T_{Pull} at which $\mathrm{Mess} = \emptyset$.

⌟

Lemma 7.1. *The function* $\mathrm{Step}(x,t)$ *defined above for* (x,t) *in* $(\Gamma \setminus \Lambda) \times [T_0, T_1]$, *can be extended, for* $t \geqslant T_{\mathrm{Swell}}$, *uniquely over all* x *with* $\eta(x,t).\mathrm{Cur} \in \mathbb{Z}_3$.

Proof. Since the complement of $*$s is simply connected, it is certainly true separately for each t that $\text{Step}(\cdot,t)$ can be extended uniquely, to satisfy (3.2) for horizontal edges. It remains to show that it also satisfies (3.2) for vertical edges. This is easy to check: at any one time, only one update needs to be considered. If this update changes a clock value then this change will clearly obey (3.2). If it removes a $*$ then (3.2) does not apply. □

To bound the differences differences between consecutive members of the sequence T_{Form}, T_{Swell}, T_{Pull}, T_{Sync}, T_{Done}, we will construct a blame sequence for each one of them.

Lemma 7.2. *Suppose that $t_0 < T_{\text{Swell}}$ is not an update time. There is a forward blame sequence $w_0, w_1, \dots, w_n$, with $w_i = (u_i, t_i)$, $n \leqslant 3L$ and $t_n \leqslant T_{\text{Form}}$.*

Proof. Let $S = \{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\}$. Let u_0 be be a non-$*$ at time t_0 to which the Swell rule is applicable. We present now the general creation rule of the sequence w_i, starting with $i = 0$. We have the following cases.

(1) $t_i \leqslant T_{\text{Form}}$, so we set $n = i$ and stop.

(2) Let t'_i be the previous update time of u_i. The Swell rule became applicable to u_i during (t'_i, t_i): thus, there was some cell

$$u_{i+1} \in (u_i + S) \cap \text{Mess}(t_i) \setminus \text{Mess}(t'_i).$$

Let t_{i+1} be the time during (t'_i, t_i) at which u_{i+1} became a $*$.

Applying this rule repeatedly, we eventually hit the case (1) and stop. We have $n \leqslant 3L$ since $h(u_{i+1}) = h(u_i) + 1$ holds for each i. □

Lemma 7.3. *Suppose that $t_0 < T_{\text{Pull}}$ is not an update time. There is a backward blame sequence $w_0, w_1, \dots, w_n$, with $w_i = (u_i, t_i)$, further $d(u_0, \Lambda) = 3L$, $n \in [0, G+3L]$ and $t_n \leqslant T_{\text{Swell}}$.*

Proof. Since $t_0 < T_{\text{Pull}}$, there is a site u_0 with $d(u_0, \Lambda) = 3L$ and $\text{Step}(u_0, t_0) < M$. We present the generation rule of the sequence $w_i = (u_i, t_i)$, starting with $i = 0$. (It is enough to define u_{i+1}, since in any backward blame sequence, t_{i+1} is the last update time of u_{i+1} before t_i.) The sequence will also have the property

$$\text{Step}(u_i, t_i) = \text{Step}(u_0, t_0) - i. \tag{7.1}$$

We have the following cases.

(1) $t_i \leqslant T_{\text{Swell}}$: let $n = i$ and stop.

Otherwise we cannot have $u_i \in \Lambda$. Indeed then $d(u_0, \Lambda) = 3L$ implies $i \geqslant |u_i - u_0| \geqslant 3L$, and

$$\begin{aligned} \text{Step}(u_0, t_0) &= \text{Step}(u_i, t_i) + i \\ &\geqslant \text{Step}(u_i, T_{\text{Swell}}) + 3L \\ &\geqslant M && \text{since } h(\cdot) \text{ cannot change more than } 3L \text{ in } \Lambda. \end{aligned}$$

(2) At time t_i the Emulate rule was applicable to u_i: let $u_{i+1} = u_i$.

(3) The Emulate rule was not applicable, so at time t_i, site u_i had a neighbor u_{i+1} with

$$\mathrm{Step}(u_{i+1},t_{i+1}) = \mathrm{Step}(u_i,t_i) - 1.$$

The relation (7.1) follows from the construction. Suppose that we reach $i = G+3L$. In this case, noting that $u_i \in \Gamma \setminus \Lambda$,

$$\begin{aligned}\mathrm{Step}(u_0,t_0) = \mathrm{Step}(u_0,t_0) &= \mathrm{Step}(u_i,t_i) + G + 3L \\ &\geqslant \mathrm{Step}(u_i,T_{\mathrm{Swell}}) + G + 3L \\ &\geqslant \mathrm{Step}(u_0,T_{\mathrm{Swell}}) + 3L && \text{by the Slack}(L,G)\text{ condition,} \\ &\geqslant M && \text{since } d(u_0,\Lambda) = 3L.\end{aligned}$$

This can only happen if $t_0 \geqslant T_{\mathrm{Pull}}$, therefore we finish by the time we reach $n = G+3L$. □

Lemma 7.4. *For $t \geqslant T_{\mathrm{Pull}}$, for all cells $u \notin \mathrm{Mess}(t)$ with $d(u,\Lambda) \leqslant 3L$ we have $\mathrm{Step}(u,t) \geqslant M - 6L$.*

Proof. Because we have $t \geqslant T_{\mathrm{Swell}}$, for every $u \notin \mathrm{Mess}(t)$ there is a neighbor $u' \notin \mathrm{Mess}(t)$ with $h(u') = h(u) + 1$. Therefore we can move from u on a height-increasing path. Within $3L$ steps we will leave Λ, within $6L$ steps we will reach a cell v that is at a distance $3L$ from Λ. Since $t \geqslant T_{\mathrm{Pull}}$, we have $\mathrm{Step}(v,t) \geqslant M$. □

Lemma 7.5. *Suppose $t_0 < T_{\mathrm{Sync}}$. There is a backward blame sequence $w_0,\dots,w_n$ with $w_i = (u_i,t_i)$, further $n \leqslant 18L$ and $t_n \leqslant T_{\mathrm{Pull}}$.*

Proof. Since $t_0 < T_{\mathrm{Sync}}$, there is a cell u_0 on the surface of $\mathrm{Mess}(t_0)$ that has a neighbor v on the surface with $\mathrm{Step}(v,t_0) > \mathrm{Step}(u_0,t_0)$.

1. We will generate a backward blame sequence $w_0,\dots,w_n$, $n \leqslant 3L$.

   Let $w_i = (u_i,t_i)$; we have the following cases.

   (1) $t_i < T_{\mathrm{Pull}}$, so let $n = i$ and stop.

   (2) Either the Emulate or the Syncronize rule is applicable to u_i at time t_i: let $u_{i+1} = u_i$.
   In the other cases, we have $\mathrm{Step}(u_i,t_i) = \mathrm{Step}(u_i,t_i{-}_{u_i})$. Thus, either u_i is not on the surface and Emulate is not applicable to it, or it is on the surface but Syncronize is not applicable to it at time t_i.

   (3) For some neighbor u_{i+1} of u_i we have $\mathrm{Step}(u_{i+1},t_i) = \mathrm{Step}(u_i,t_i) - 1$.

   (4) u_i is on the surface, all of its non-$*$ neighbors have the same Step value at time t_i as u_i. Then u_i has a non-$*$ neighbor u_{i+1} with $h(u_{i+1}) = h(u_i) + 1$. Indeed, otherwise u_i would be turned into a $*$ by the Swell rule at time t_i which is not possible, since $t_i \geqslant T_{\mathrm{Swell}}$.

It remains to upperbound the termination time n. Note that $\mathrm{Step}(u_i,t_i)$ is nondecreasing: $\mathrm{Step}(u_{i+1},t_{i+1})$ is either $\mathrm{Step}(u_i,t_i)$ or $\mathrm{Step}(u_i,t_i) - 1$. Let us call the indices i with $\mathrm{Step}(u_{i+1},t_{i+1}) = \mathrm{Step}(u_i,t_i)$ the *level* stages, and the indices i with $\mathrm{Step}(u_{i+1},t_{i+1}) = \mathrm{Step}(u_i,t_i) - 1$ the *progress* stages.

2. The number of progress stages is at most $6L$.

   Proof. By the Syncronize rule, the Step value of surface cells never progresses beyond M. Since $\mathrm{Step}(u_i, t_i)$ is nonincreasing, and since $t_i > T_{\mathrm{Pull}}$, the cells u_i stay within distance $3L$ of Λ. Lemma 7.4 implies from this that $\mathrm{Step}(u_i, t_i)$ cannot decrease below $M - 6L$.

3. The number of level stages is at most $12L$.

   Proof. Every level stage increases $h(u_i)$ by 1, and every progress stage decreases it by at most 1. If the number of level stages is by $6L$ more than the number of progress stages then we would move behond distance $3L$ of Λ, which is impossible since we are after T_{Pull}.

So we have $n \leqslant 6L + 12L = 18L$. □

Lemma 7.6. *Suppose $t_0 < T_{\mathrm{Done}}$. There is a backward blame sequence $w_0, \dots, w_n$ $w_i = (u_i, t_i)$ with $t_n \leqslant T_{\mathrm{Sync}}$ and $n \leqslant 3L$.*

Proof. Let u_0 be any cell in $\mathrm{Mess}(t_0)$. Let us generate the rest of the blame sequence with $u_{i+1} \in \mathrm{Mess}(t_i)$. We have the following cases.

(1) $t_i \leqslant T_{\mathrm{Sync}}$, so let $n = i$ and stop.

(2) Since $t_i \geqslant T_{\mathrm{Sync}}$, all neighbors of u_i belong to $\mathrm{Mess}(t_i)$. Let $u_{i+1} = u_0 + (0, 0, 1)$.

We have $n \leqslant 3L$, since $h(u_i)$ is increasing in every step, and u_i must still be in Λ. □

Let us draw the probabilistic conclusions from these lemmas, using Proposition 2.7. Our goal is to upperbound the probability of the event $[T_1 \leqslant T^{\eta}_{\mathrm{Done}}]$. We will introduce constant deadlines $t_{\mathrm{Form}}, t_{\mathrm{Swell}}, \dots$ for each milestone time $T_{\mathrm{Form}}, T_{\mathrm{Swell}}, \dots$, and we will bound the probability for each milestone to occur past deadline.

1. Let $t_{\mathrm{Form}} = T_0 + L$. For each defect cell the probability that the Form rule will not get applied to it for a time L is at most e^{-L}, therefore, using (4.2) we have

$$\mathbf{P}[\, t_{\mathrm{Form}} < T_{\mathrm{Form}} \,] \leqslant 2L^3 e^{-L}.$$

2. Let $t_{\mathrm{Swell}} = t_{\mathrm{Form}} + \gamma(3L)$. Lemma 7.2 says that if $t_{\mathrm{Form}} \geqslant T_{\mathrm{Form}}$ and $t_{\mathrm{Swell}} < T_{\mathrm{Swell}}$ then there is a forward blame sequence with length $\leqslant 3L$ and span $\geqslant \gamma(3L)$. Just as above, we conclude from here

$$\mathbf{P}[\, T_{\mathrm{Form}} \leqslant t_{\mathrm{Form}} < t_{\mathrm{Swell}} < T_{\mathrm{Swell}} \,] \leqslant 2L^3 e^{-\delta(3L)}.$$

3. Let $t_{\mathrm{Pull}} = t_{\mathrm{Swell}} + \gamma(G + 3L)$. Using Lemma 7.3 says that if $t_{\mathrm{Swell}} \geqslant T_{\mathrm{Swell}}$ and $t_{\mathrm{Pull}} < T_{\mathrm{Pull}}$ then there is a backward blame sequence with length $\leqslant (G + 3L)$ and span $\geqslant \gamma(G + 3L)$, starting from a cell at distance $3L$ from Λ. It is easy to see that for large L, the number of such cells is at most $8 \cdot 49L^2 < 400L^2$, so

$$\mathbf{P}[\, T_{\mathrm{Swell}} \leqslant t_{\mathrm{Swell}} < t_{\mathrm{Pull}} < T_{\mathrm{Pull}} \,] \leqslant 400L^2 e^{-\delta(G+3L)}.$$

4. Let $t_{\text{Sync}} = t_{\text{Pull}} + \gamma(18)L$. Using Lemma 7.5 the same way as the others, we conclude

$$\mathbf{P}[\,T_{\text{Pull}} \leqslant t_{\text{Pull}} < t_{\text{Sync}} < T_{\text{Sync}}\,] \leqslant 2L^3 e^{-\delta(18L)}.$$

5. Let $t_{\text{Done}} = t_{\text{Sync}} + \gamma(3L)$. Using Lemma 7.6 the same way as the others, we conclude

$$\mathbf{P}[\,T_{\text{Sync}} \leqslant t_{\text{Sync}} < t_{\text{Done}} < T_{\text{Done}}\,] \leqslant 2L^3 e^{-\delta(3L)}.$$

Chaining the definitions we find $t_{\text{Done}} = (27\gamma + 1)L + \gamma G$. Chaining the probability estimates we find

$$\begin{aligned}\mathbf{P}[\,t_{\text{Done}} < T_{\text{Done}}\,] &\leqslant 2L^3 e^{-L} + 2L^3 e^{-\delta(3L)} + 400L^2 e^{-\delta(G+3L)} + 2L^3 e^{-\delta(18L)} + 2L^3 e^{-\delta(3L)} \\ &\leqslant e^{-dL}\end{aligned}$$

for an appropriately chosen constant $d > 0$.

References

[1] Piotr Berman and Janos Simon, *Investigations of fault-tolerant networks of computers*, Proc. of the 20-th Annual ACM Symp. on the Theory of Computing, 1988, pp. 66–77. 1, 1, 3.2, 4.1

[2] Peter Gács, *Reliable computation with cellular automata*, Journal of Computer System Science **32** (1986), no. 1, 15–78. 4.1

[3] ______, *Deterministic parallel computations whose history is independent of the order of updating*, cs.DC/0101026, 1995. 1

[4] ______, *A new version of Toom's proof*, Tech. report, Department of Computer Science, Boston University, TR 95-009, Boston, MA 02215, 1995. 4.1

[5] ______, *Reliable cellular automata with self-organization*, Journal of Statistical Physics **103** (2001), no. 1/2, 45–267, See also www.arXiv.org/abs/math.PR/0003117 and the proceedings of the 1997 Symposium on the Theory of Computing. 4.1, 4.3

[6] Peter Gács and John Reif, *A simple three-dimensional real-time reliable cellular array*, Journal of Computer and System Sciences **36** (1988), no. 2, 125–147. 1, 3.1, 4.2, 4.2

[7] Lawrence F. Gray, *Toom's stability theorem in continuous time*, Perplexing Problems in Probability (Maury Bramson and Rick Durrett, eds.), Birkhäuser, Boston, 1999, pp. 331–353. 4.1

[8] Thomas M. Liggett, *Interacting particle systems*, Grundlehren der mathematischen Wissenschaften, vol. 276, Springer Verlag, New York, 1985. 2.6

[9] Andrei L. Toom, *Stable and attractive trajectories in multicomponent systems*, Multicomponent Systems (R. L. Dobrushin, ed.), Advances in Probability, vol. 6, Dekker, New York, 1980, Translation from Russian, pp. 549–575. 4.1

[10] John von Neumann, *Probabilistic logics and the synthesis of reliable organisms from unreliable components*, Automata Studies (C. Shannon and McCarthy, eds.), Princeton University Press, Princeton, NJ., 1956. 4.1