Synchronization in 3 dimensions

Matthew Cook[1] Erik Winfree[1] Péter Gács[2]

[1]California Institute of Technology

[2]Department of Computer Science
Boston University

March 21, 2008

Asynchrony

In a parallel computation model, distinction between two update models.

Synchronous Discrete time steps $0, 1, 2, \ldots$, each component is updated by a local (deterministic or random) "transition rule".

Asynchronous The update order is not deterministic. For example, the update times form a random process: typically a Poisson process. This is the case if the whole system is a continuous-time Markov process.

Cellular automata

- Elementary parts: cells, or sites. Set of cells: for example, $\mathbb{C} = \mathbb{Z}^3$, or $\mathbb{C} = \mathbb{Z}/m\mathbb{Z}$ (periodic boundary conditions).
- Finite set $\mathbb{S}$ of (local) states.
- (Space-) configuration: any function $\xi : \mathbb{C} \to \mathbb{S}$.

$$\mathbb{C} = \mathbb{Z} \qquad \mathbb{S} = \{0, 1, 2\}$$

$\cdots$	1	0	1	1	2	0	1	0	0	0	0	2	2	1	2	1	0	$\cdots$
			-1	0	1	2												

$$\xi(-1) = 1, \xi(0) = 1, \xi(1) = 2, \ldots$$

Space-time configuration $\eta(x,t)$.

time

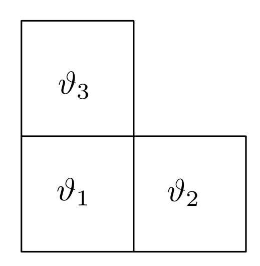
0	1	0	1	1	2	0	1	0	0	0	0	2	2	1	2	1	0
1	1	1	1	0	2	0	1	0	1	0	2	0	2	1	2	1	0
2	1	0	1	1	$\boxed{2}$	$\boxed{1}$	1	0	0	0	0	2	2	1	2	2	0
	0	0	0	1	0	0	2	1	0	2	0	1	2	0	1	1	1
			−1	0	1	2											

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

Neighborhood function: $N(x) = \{\vartheta_1(x), \dots, \vartheta_r(x)\}$.
Normally $\mathbb{C} = \mathbb{Z}^d$ and we have $\vartheta_i(x) = x + \vartheta_i(\mathbf{0})$.

Examples

- von Neumann neighborhood: the 7 nearest neighbors (including itself) of a point, say, in the lattice $\mathbb{Z}^3$.
- Toom neighborhood: $(\vartheta_1(\mathbf{0}), \vartheta_2(\mathbf{0}), \vartheta_3(\mathbf{0})) = ((0,0),(0,1),(1,0))$.

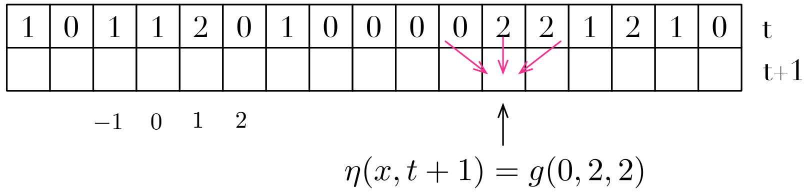


In discrete time, we say η is a trajectory of local transition function $g : \mathbb{S}^r \to \mathbb{S}$ if

$$\eta(x, t+1) = g(\eta(\vartheta_1(x), t), \dots, \eta(\vartheta_r(x), t)).$$

Example

$\mathbb{C} = \mathbb{Z}$, $N = \{-1, 0, 1\}$.



Here is a trajectory of Wolfram's rule 110 on $\mathbb{Z}/(17\,\mathbb{Z})$.

time																	
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											$13 = -4$

The rule says: "If your right neighbor is 1 and the neighborhood state is not 111 then your next state is 1, otherwise 0".

So, a (deterministic, synchronous) cellular automaton is given by these data:

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

Example (The Toom Rule)

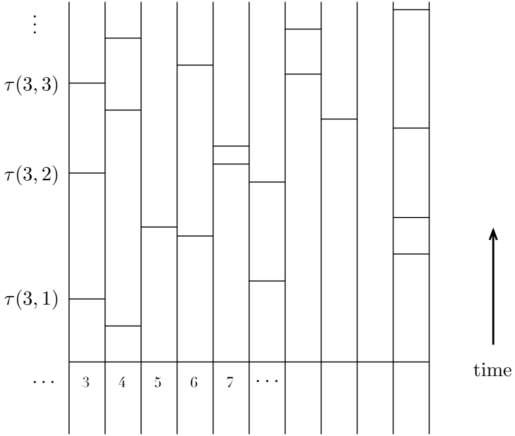
$\mathbb{C} = \mathbb{Z}^2$, $\mathbb{S} = \{0, 1\}$,

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

The new state is the majority of the state of the cell itself, and of its northern and eastern neighbor.
$\mathrm{Maj}(x,y,z)$ can be extended to the case of larger alphabets: when no symbol is in majority, let the result be y.

Asynchronous updating

Continuous time: cell x has update times $\tau(x,n) \in \mathbb{R}$,
$0 < \tau(x,1) < \tau(x,2) < \cdots$
with $\tau(x,n) \underset{n}{\to} \infty$.

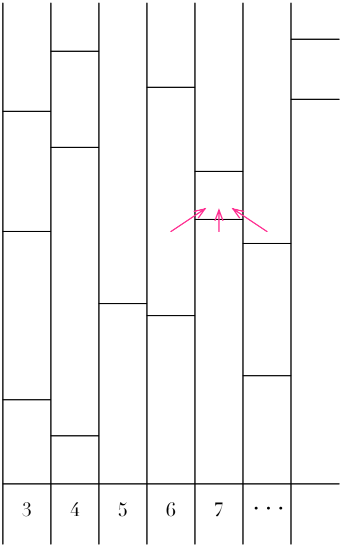


Asynchronous updating

At any one time, only one site is updated:

$$\eta(x,t) = g(\eta(\vartheta_1(x), t-\varepsilon), \ldots, \eta(\vartheta_r(x), t-\varepsilon)),$$

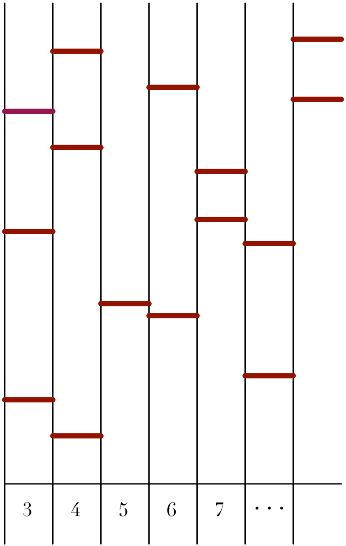
where $\varepsilon = \varepsilon(x,t)$ is such that the neighborhood does not change during $[t-\varepsilon, t)$.



Space-time neighbors

Set of update events

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



Space-time neighbors

Let $z = (x, t)$,

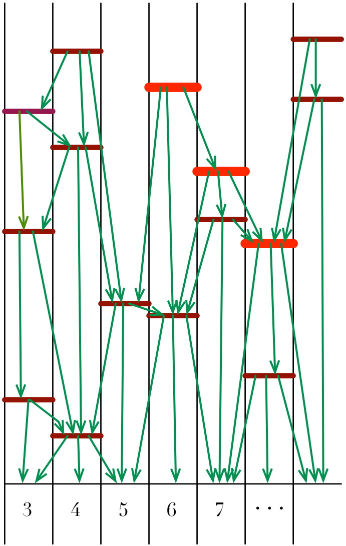
$$\underline{\tau}(x,t) = \max_{\tau(x,k)<t} \tau(x,k),$$

$$\Theta_i^{\mathcal{U}}(z) = (\vartheta_i(x), \underline{\tau}(\vartheta_i(x), t)).$$

Then $\Theta_i^{\mathcal{U}}(z)$ is the event at which neighbor $\vartheta_i(x)$ obtained the state influencing z.

Space-time neighbors of z: the events $\Theta_i^{\mathcal{U}}(z)$, $i = 1, \dots, r$.

Directed graph $\mathcal{G}$ on vertices of $\mathcal{U}$: directed edge from each update event z to each of its space-time neighbors.



We say that η is an asynchronous trajectory if

$$\eta(z) = g(\eta(\Theta_1^{\mathscr{U}}(z)), \ldots, \eta(\Theta_r^{\mathscr{U}}(z))).$$

This recursive definition, along with the initial configuration $\eta(\cdot, 0)$ determine η uniquely if the the graph $\mathscr{G}$ has no infinite directed path. This condition will hold with probability 1 in our models with a random update set $\mathscr{U}$. We will also have, with probability 1:

$$(x_1, t_1), (x_2, t_2) \in \mathscr{U} \Rightarrow t_1 \neq t_2.$$

Random updating

Now, let $\eta(x,t)$ be a stochastic process. It is a trajectory of the continuous-time probabilistic cellular automaton (sometimes called interacting particle system)

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

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

- The different sequences $(\,\tau(x,n) : n = 0,1,2,\dots\,)$ are independent of each other.
- The sequence of increments $\tau(x,n+1) - \tau(x,n)$ is independent.
- Each variable $\tau(x,n+1) - \tau(x,n)$ has the same exponential distribution with rate 1: $\mathbf{P}[\,\tau(n+1,x) - \tau(n,x) > t\,] = e^{-t}$.

Thus, for each x, the sequence $(\,\tau(x,n) : n > 0\,)$ is a Poisson process with rate 1. And, η is a continuous-time Markov process.

Sensitivity to update order

Some computations are naturally asynchronous: the result is independent of the choice of the update set $\mathscr{U}$. (This can be formulated precisely.) Other computations rely substantially on the timing of many parallel updates. Example: the "Toom-layering" itroduced below.

Asynchronously simulating a synchronous computation

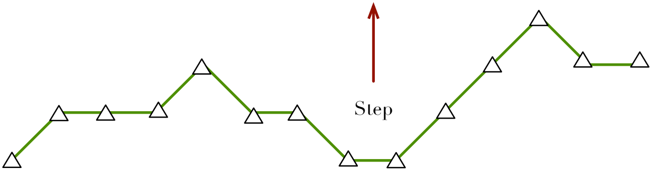
How to simulate a discrete-time computation $\zeta(x,p)$, $p = 0, 1, 2, \ldots$ by a continous-time $\eta(x,t)$? If we can recover $\zeta(x,p)$ from η then we can also recover p. Denote

$$\mathrm{Step}^{\eta}(x,t) = p.$$

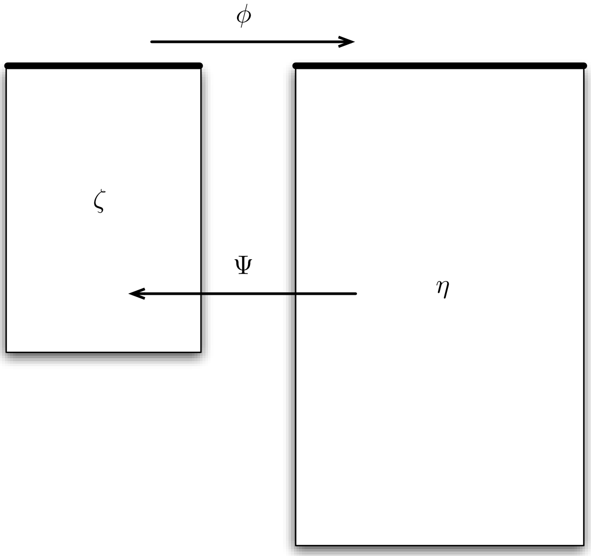
We want to enforce

$$|\mathrm{Step}^{\eta}(y,t) - \mathrm{Step}^{\eta}(x,t)| \leqslant 1$$

for neighbors x,y. This is called the marching soldiers scheme.



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



The mod 3 trick

We cannot store $p = \mathrm{Step}^{\eta}(x,t)$ in a finite state, but will store it modulo 3. Let the state $\eta(x,t)$ have three fields: $\mathrm{Cur}, \mathrm{Prev}, \mathrm{Clock}$ with the *intended values*

$$\begin{aligned}\eta(x,t).\mathrm{Cur} &= \zeta(x,p),\\ \eta(x,t).\mathrm{Prev} &= \zeta(x,p-1),\\ \eta(x,t).\mathrm{Clock} &= p \bmod 3.\end{aligned}$$

Denote n amod m the smallest absolute value remainder,

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

With the intended values we will have

$$\Delta^{\eta}(x,y,t) = \mathrm{Step}^{\eta}(x,t) - \mathrm{Step}^{\eta}(y,t).$$

If

- g is the transition function of ζ,
- $\tilde{g}$ is the transition function for η,

then $\tilde{g}$ will satisfy some conditions called rules here. Suppose that $\tilde{g}$ changes the state $s = \eta(z)$ with $z = (x, t)$ to some state $\bar{s}$, further $s.\text{Clock} \in \mathbb{Z}_3$.

Rule (Wait)

We have

- $(\bar{s}.\text{Clock} - s.\text{Clock})$ amod $3 \neq -1$, that is the clock will not "decrease".
- If z has a neighbor $z' \in \hat{N}(z)$ with state $s' = \eta(z')$ and with $s'.\text{Clock} \in \mathbb{Z}_3$, $\Delta(s, s') < 0$ then $\bar{s} = s$. That is, the clock does not increase if some neighbor that would be "left behind".

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

$$\mathrm{Trans}^{\eta}(z) = g(q_1, \dots, q_r)$$

where

$$s_i = \eta(\Theta_i^{\mathcal{U}}(z)), \quad q_i = \begin{cases} s_i.\mathrm{Cur} & \text{if } \Delta(s, s_i) = 0, \\ s_i.\mathrm{Prev} & \text{if } \Delta(s, s_i) = 1. \end{cases}$$

This is the intended new simulated value.

Rule (Emulate)

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

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

What did we accomplish formally?

Definition (Asynchronous simulation)

An asynchronous simulation is a tuple $(\mathbf{A}, \widetilde{\mathbf{A}}, \phi, \Psi)$ where

$$\mathbf{A} = \mathrm{Aut}(\mathbb{C}, \mathbb{S}, \vartheta(\cdot), g(\cdot)),$$
$$\widetilde{\mathbf{A}} = \mathrm{Aut}(\mathbb{C}, \widetilde{\mathbb{S}}, \vartheta(\cdot), \widetilde{g}(\cdot))$$

and ϕ, Ψ are the (encoding, decoding) mappings such that:

- If ξ is a space configuration of $\mathbf{A}$ then $\phi(\xi)$ is a space configuration of $\widetilde{\mathbf{A}}$.
- If η is an asynchronous trajectory of $\widetilde{\mathbf{A}}$ with $\eta(\cdot, 0) = \phi(\xi)$ then $\Psi(\eta)$ is a synchronous trajectory ζ of $\mathbf{A}$ with $\zeta(\cdot, 0) = \xi$.

Proposition

The mod 3 scheme introduced above defines an asynchronous simulation for appropriate ϕ, Ψ.

The slowdown

In the mod 3 scheme, we have update attempts in which nothing happens.
What is the price in slowdown?
For the random updating model, it is shown in [Berman, Simon 88] that average slowdown is at most by a constant factor.

Fault tolerance

A simple solution

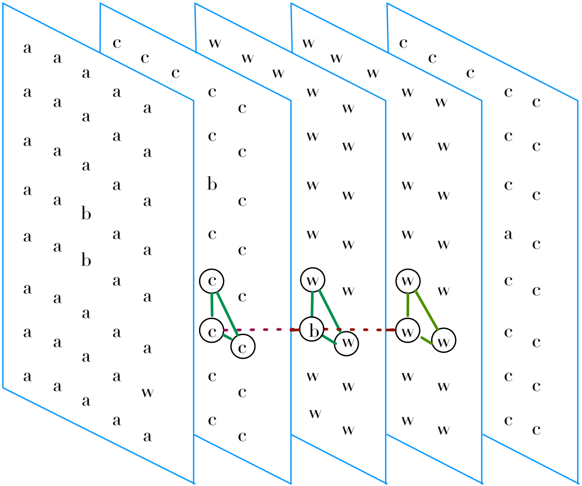
The simplest known fault-tolerant computation model is the three-dimensional cellular automaton introduced in [Gacs-Reif 88].

Definition (Toom-layering)

Let $\mathbf{U}$ be an arbitrary 1-dimensional cellular automaton. We define its Toom-layering as a 3-dimensional automaton

$$\mathbf{U}'.$$

In its initial configuration, 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 $\mathbf{U}'$ is: Toom's rule within each plane, then the rule of $\mathbf{U}$ across the planes.



Transition rule of $\mathbf{U}'$: Toom's rule within each plane, then the rule of $\mathbf{U}$ across the planes.

In what sense is this fault-tolerant? Consider a random process $\eta(u,t)$ (discrete t) that follows the transition rule $\mathbf{U}'$ only approximately: at each space-time point (u,t), the transition rule is applied except with some probability $< \varepsilon$, a fault occurs, when $\eta(u,t)$ becomes something else. We assume that faults occur independently of each other.

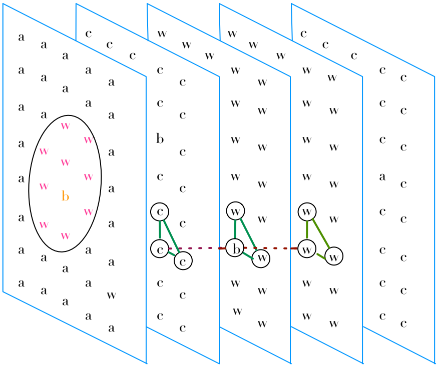
Proposition

There is a constant c with the following property. Let $\zeta(x,t)$ be a computation (space-time configuration) of $\mathbf{U}$, and let $\eta(x,y,z,t)$ be a space-time configuration of the Toom-layering $\mathbf{U}'$ 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

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

The synchronization problem of the 3D model

The Toom-layering is trying to enforce the constancy of $\eta(x,y,z,t)$ in y,z. If some cells update before the others, this property is violated, and the rule will try to “correct” the situation, messing up everyting.

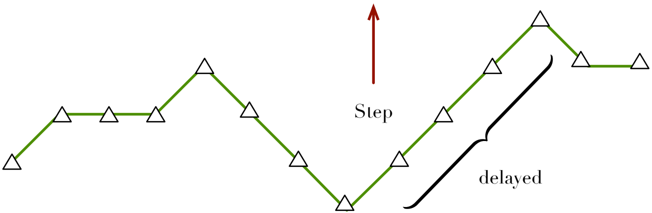


- Toom's rule itself works also in continuous time; only the Toom-layering does not.
- I have constructed continuous-time fault-tolerant cellular automata, (even in 1 dimension), but their program creates and maintains a hierarchy, and is very complex.
- No simple continuous-time fault-tolerant cellular automata are known in any dimension. The present work is trying to define one.

The combination

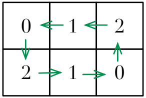
Can we combine the mod 3 synchronization scheme with Toom-layering? Maybe, but several difficulties arise.

First, even if faults do not affect the clocks, local slowdown may hurt us. It is only linear on average, but if steep slopes persist too long locally, then Toom's Rule does not get the necessary speed for error correction.

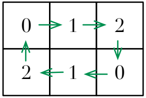


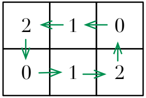
There must be a theorem of probability theory taking care of this, but I have not found it yet.

More interesting is the problem that the faults will affect the clocks, even their consistency. This is not a problem in 1 dimension, but in 2 dimensions, they can already create situations like this:



Unless corrected, these clocks will wait for each other forever.
We do not know how to correct this situation in a simple (non-hierarchical) way. Two opposite small loops can be far from each other, and everything else may seem normal locally.





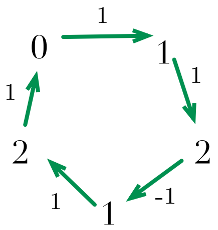
Definition (Lag)

For an arbitrary loop $P = \mathrm{loop}(u_0, \ldots, u_{n-1}) = (u_0, u_1, \ldots, u_{n-1}, u_n)$ where $u_n = u_0$, define its lag as

$$\mathrm{lag}(P, t) = \sum_{i=0}^{n-1} \Delta^{\eta}(u_i, u_{i+1}, t)$$

(each term is in $\{-1, 0, 1\}$).

$$\mathrm{lag} = 1 + 1 - 1 + 1 + 1 = 3$$



Definition (Consistency)

A (configuration in a) domain D in which all loops have zero lags is called consistent (at time t). (We may use the topological term co-boundary.)

The following is easy to prove.

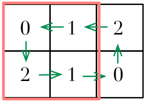
Proposition

In a domain D, the function $\Delta^{\eta}(u,v,t)$ *can be represented as* $\mathrm{Step}^{\eta}(v,t) - \mathrm{Step}^{\eta}(u,t)$ *with integer function* $\mathrm{Step}^{\eta}(u,t)$ *if and only if D is consistent.*

Definition

A loop of size 4 with nonzero lag is called a defect. (A configuration without defects may be called a co-cycle.)

It easy to see that each loop of nonzero lag contains a defect.



More generally, the following holds, for an appropriate definition of simply connected. (See next slide if there is time.)

Proposition

If a (2 or 3-dimensional) domain is simply connected and has no defects then it is consistent. (Co-cycle in a simply connected domain is a co-boundary.)

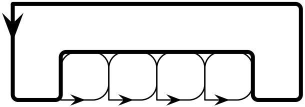
The following definitions work in two as well as three dimensions.

Definition (Addition of paths)

A directed path can be seen as the formal sum $e_1 + \cdots + e_n$ of its directed edges e_i. More generally, we introduce formal sums $\sum_i c_i e_i$ with integers c_i. If e_1 and e_2 are the same edge with opposite directions then $e_1 + e_2 = 0$.

Definition (Equivalence)

A plaquette is a loop of length 4. Two directed paths P and Q are equivalent if $P - Q$ can be represented as the sum of plaquettes. A domain is simply connected if each loop in it is equivalent to 0.



The following definitions work in two as well as three dimensions.

Definition (Addition of paths)

A directed path can be seen as the formal sum $e_1 + \cdots + e_n$ of its directed edges e_i. More generally, we introduce formal sums $\sum_i c_i e_i$ with integers c_i. If e_1 and e_2 are the same edge with opposite directions then $e_1 + e_2 = 0$.

Definition (Equivalence)

A plaquette is a loop of length 4. Two directed paths P and Q are equivalent if $P - Q$ can be represented as the sum of plaquettes. A domain is simply connected if each loop in it is equivalent to 0.

3-dimensions

Interestingly, the situation is more promising in 3 dimensions (where the Toom layering runs). We will restore consistency with a relatively simple rule,

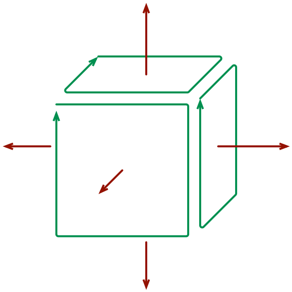
1. in the absence of new faults, and
2. in the absence of steep slopes.

We believe that more careful analysis will remove these conditions, without changing the rule.

3-dimensional topology

Proposition

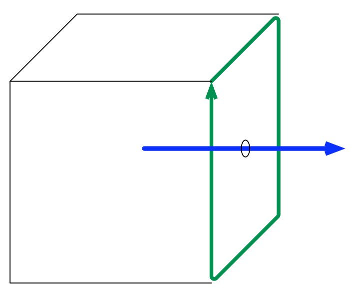
The sum the lags on the faces of a cube is 0, if each is read clockwise in the direction of the normal pointing outside.



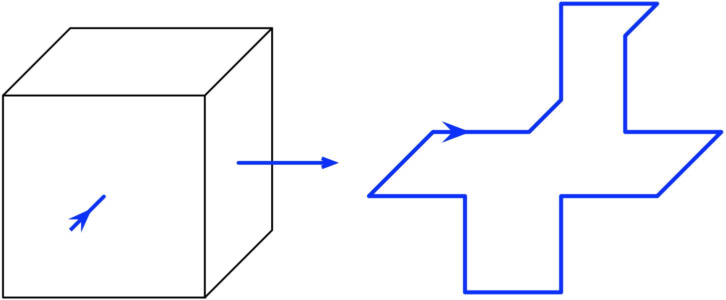
This motivates the following definition.

Definition

In 3 dimensions, each defect defines a defect vector connecting the centers of the two facing corner cubes, in the direction towards which the lag, read counterclockwise, is positive.



The Proposition implies that if a defect vector enters a corner cube, another one must leave it.



So, defects form closed paths.

The program

Here is the plan for eliminating defects. Introduce a new value $*$ for Clock. The set of $*$'s will be called the Mess.

1. Mark all neighbors of each defect with a $*$, creating the initial Mess.
2. Fill in the holes in the loops of the Mess, thus extending it. Now the complement of the Mess is simply connected, hence (by the earlier Proposition) consistent.
3. Propagate consistent clock values into the Mess.

Both part 2 and part 3 are nontrivial; the proof that all this will happen in linear time is also nontrivial, even in the absence of additional faults.

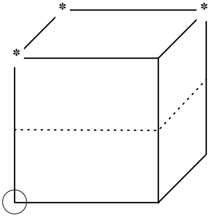
Rule (Form)

If you participate in a defect then become a $*$.

Rule (Swell)

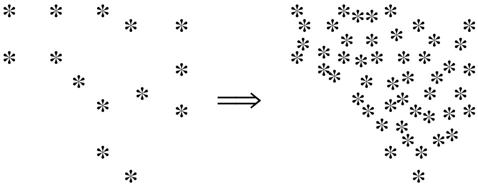
If you cannot be separated from the Mess in the $(1, 1, 1)$ corner block by a plane parallel to one of the coordinate planes, then become a $*$.

(This is in the spirit of the Toom Rule.) Below, the circled point can be separated from the mess in its $(1, 1, 1)$ corner cube, hence does not become $*$.

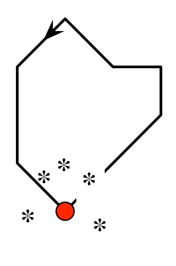


Lemma

The Mess never grows beyond an enclosing cube. When it cannot grow any more, it has no “holes”, in the sense that its complement is simply connected and hence (since has no defects), consistent.



Proof idea. Try to pull a loop gradually together into a point. If it does not go further, there is a “bottom” point that is so surrounded by the Mess that the Swell rule would have turned it into a $*$.



□

How to propagate the clocks consistently into a set from its consistent environment? This is not always possible, but is certainly easy if all your neighbors have the same value:

Rule (Shrivel)

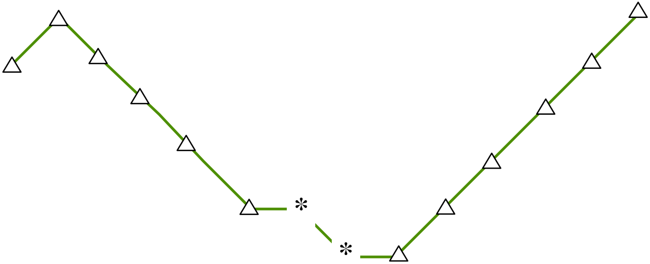
Suppose that you are a $*$ with no higher neighbors that are $*$s and all your non-$*$ neighbors have the same clock value. Then change to this common clock value.

0	1	2
$*$	$*$	2
$*$	$*$	2
$*$	$*$	$*$

$\Longrightarrow$

0	1	2
$*$	$*$	2
$*$	2	2
$*$	$*$	$*$

Now 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.



Rule (Synchronize)

Let x be the point with clock value c. Suppose that

- all neighbors of x have Clock $\in \{*, c, c+1 \bmod 3\}$.
- both x and one of its neighbors y are surface neighbors (defined appropriately) of a common $*$, with y's clock value being $c+1 \bmod 3$.

Then set $\mathrm{Clock}(x) := c+1$.

0	1	2
*	*	2
*	1	2
*	*	*

$\Longrightarrow$

0	1	2
*	*	2
*	2	2
*	*	*

What does all this prove?

Before saying it, let us introduce our conditions.

Definition

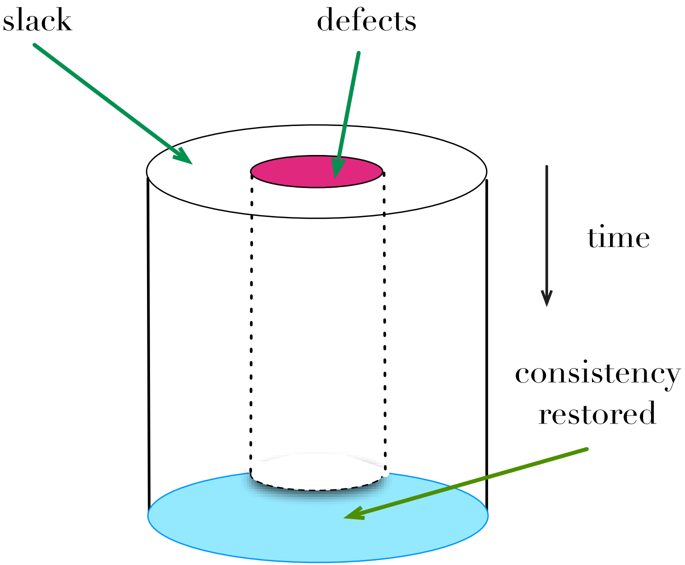
For integers $L < G$ and a site v_0, define the ball

$$B(x,r) = \{\, u \in \mathbb{Z}^3 : |u - x| \leqslant r \,\}.$$

Space-time configuration η is (L,G)-good at point (x,t) if if at time t:

1. All defects are contained in $B(x,L)$.
2. For $u,v \in B(x,G) \smallsetminus B(x,L)$, with $|u-v| \leqslant G + 3L$ we have $\mathrm{Step}(u) - \mathrm{Step}(v) \leqslant G$.

Condition 2 says that there are no steep slopes (enough slack) in the clocks nearby our bunch of defects in $B(x,L)$.


slack
defects
time
consistency
restored

Theorem

There are constants $c_1, c_2, d > 0$ with the following properties.
Let **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, time T_0 and numbers $G > 8L > 0$ be given, with

$$T_1 = T_0 + c_1 L + c_2 G.$$

Let the stochastic process η be a trajectory of $\widetilde{\mathbf{A}}$ in the set $\Gamma(v_0, G) \times [T_0, T_1]$, and let it be (L, G) good in v at time T_0. Then with probability $> 1 - e^{-dL}$, there is a step function over

$$(\Gamma(v, G) \times [T_0, T_1]) \setminus (\Gamma(v, L) \times [T_0, T_1)).$$

In other words the consistency of the clocks, possibly disturbed inside $B(v_0, L)$ at time T_0, will be restored by time T_1.

Proof method: blame sequences

(The method borrowed from [Berman-Simon 88].) Let $t_0 > t_1 > \cdots > t_n$ and consider the sequence $w_0, w_1, \ldots, w_n$ with $w_i = (u_i, t_i)$ in which u_{i+1} is a neighbor of u_i.

- It is a forward blame sequence if t_i is the first update time of u_i after t_{i+1}.
- It is a backward blame sequence if t_{i+1} is the last update time of u_{i+1} before t_i

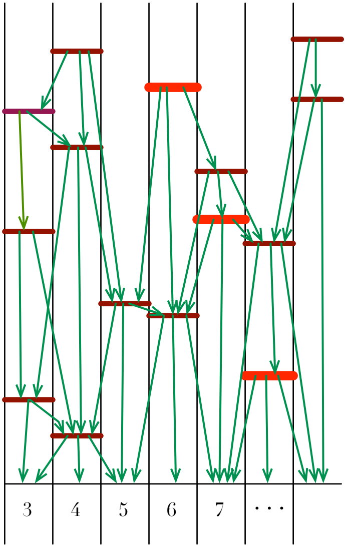
The difference $t_0 - t_n$ is the time span of the blame sequence, and n is its length.

Proof method: blame sequences

(The method borrowed from [Berman-Simon 88].) Let $t_0 > t_1 > \cdots > t_n$ and consider the sequence $w_0, w_1, \ldots, w_n$ with $w_i = (u_i, t_i)$ in which u_{i+1} is a neighbor of u_i.

- It is a forward blame sequence if t_i is the first update time of u_i after t_{i+1}.
- It is 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.

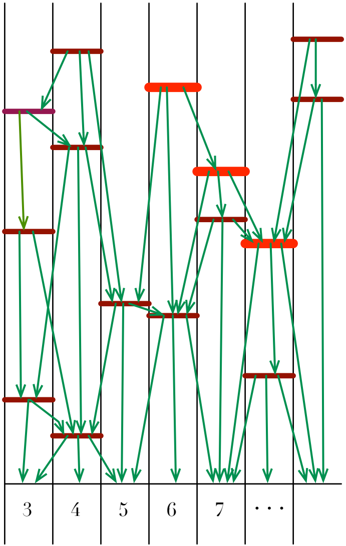


Proof method: blame sequences

(The method borrowed from [Berman-Simon 88].) Let $t_0 > t_1 > \cdots > t_n$ and consider the sequence $w_0, w_1, \ldots, w_n$ with $w_i = (u_i, t_i)$ in which u_{i+1} is a neighbor of u_i.

- It is a forward blame sequence if t_i is the first update time of u_i after t_{i+1}.
- It is 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.

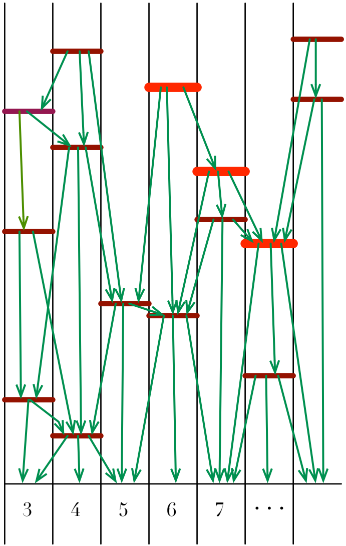


Proof method: blame sequences

(The method borrowed from [Berman-Simon 88].) Let $t_0 > t_1 > \cdots > t_n$ and consider the sequence $w_0, w_1, \ldots, w_n$ with $w_i = (u_i, t_i)$ in which u_{i+1} is a neighbor of u_i.

- It is a forward blame sequence if t_i is the first update time of u_i after t_{i+1}.
- It is 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.



Proposition

Let **A** *be a continuous-time probabilistic cellular automaton. There are constants* $\gamma, \delta > 0$ *such that for all* n*, for all space-time points* z*, the probability that a blame sequence of length* $\leqslant n$ *and time span* $\geqslant \gamma n$ *starts at* z *is less than* $e^{-\delta n}$.