Reliable Cellular Automata with Self-Organization

Peter Gács[1]
Computer Science Department
Boston University
Boston, MA 02215
gacs@bu.edu

Abstract

In a noisy cellular automaton, even if it is infinite, it is nontrivial to keep a bit of information for more than a constant number of steps. A clever solution in 2 dimensions has been applied to a simple 3-dimensional fault-tolerant cellular automaton. This technique did not solve the following problems: remembering a bit of information in 1 dimension; computing in dimensions lower than 3, or with non-synchronized transitions.

With a more complex technique using a hierarchy of simulations, we construct an asynchronous one-dimensional reliable cellular automaton, which is also "self-organizing". This means that if the input information has constant size, the initial configuration can be homogenous: the hierarchy organizes itself.

An application to information storage in positive-temperature Gibbs states is also given.

Keywords: probabilistic cellular automata, interacting particle systems, renormalization, ergodicity, reliability, fault-tolerance, error-correction, simulation, hierarchy, self-organization, Gibbs states.

1. Introduction

A *probabilistic cellular automaton* is a homogenous array (of dimension one, two or higher) of identical, locally communicating probabilistic finite-state automata. The model is also called *interacting particle system*. Fault-tolerant computation and information storage in cellular automata is a natural and challenging mathematical problem but there are also some arguments indicating an eventual practical significance since there are advantages in uniform structure for parallel computers.

Fault-tolerant cellular automata belong to the larger category of reliable computing devices built from unreliable components, in which the error probability of the individual components is not required to decrease as the size of the device increases. In any such model, it is essential that the faults are assumed to be transient: they change the local state but not the local transition rule.

Due to the homogeneity of cellular automat, since large groups of errors can destroy large parts of any kind of structure, "self-stabilization" techniques are needed in conjunction with traditional error-correction.

For a version of the present abstract containing all proofs, see [7].

1.1. Historical remarks

The problem of reliable computation with unreliable components was addressed in [24] in the context of Boolean circuits. Von Neumann's solution, as well as its improved versions in [4] and [18], rely on high connectivity and non-uniform constructs. For information storage, constant redundancy has been achieved in [15] and, in a completely constructive network, in [20]. Space redundancy has been substantially reduced in [21], for computations whose computing time is larger than the storage requirement.

Our interest is in probabilistic cellular automata in which all local transition probabilities are positive (let us call such automata *noisy*): "perturbation" from a deterministic cellular automaton leads to such a system. The automaton may have e.g. two distinguished initial configurations: ξ_0 in which all cells have state 0 and ξ_1 in which all have state 1 (there may be other states besides 0 and 1). Let $p_i(x,t)$ be the probability that, starting from initial configuration ξ_i, the state of cell x at time t is i. If $p_i(x,t) > 2/3$ for all x, t then we can say that the automaton remembers the initial configuration forever.

Informally speaking, a probabilistic cellular automaton is called *mixing* if it eventually forgets all about its initial configuration (in [16], this property is called "ergodic"). Finite noisy cellular automata are always mixing. In the example above, one can define a "relaxation time" as the time by which the probability decreases below $2/3$. If an infinite automaton is mixing then the relaxation time of the corresponding finite automaton is bounded independently of size. A minimal requirement of fault-tolerance is therefore that the infinite automaton be non-mixing.

[1] Partially supported by NSF grant CCR-920484.

The difficulty in constructing non-mixing noisy one-dimensional cellular automata is that eventually large islands will randomly occur. It is hard to see how a local rule can eliminate such islands under noise. That the majority rule cannot (its perturbations are mixing) has been shown in [11] and [12].

Non-mixing noisy cellular automata for dimensions 2 and higher were constructed in [22]. The paper [9] applies Toom's work to design a simple three-dimensional fault-tolerant cellular automaton that simulates arbitrary one-dimensional arrays and thus performs an arbitrary computation. The original proof was simplified and adapted to strengthen these results in [3]. Let us remark that a three-dimensional fault-tolerant cellular automaton cannot be built to arbitrary size in physical space: the (inherently irreversible) error-correcting operations produce heat that needs a separate escape route for each cell.

A simple one-dimensional deterministic cellular automaton eliminating finite islands in the absence of failures was defined in [8]. It is now known (see [17]) that noise (at least, if strongly biased) makes this automaton mixing.

1.2. Hierarchical constructions

The limited geometrical possibilities in one dimension suggest that only some non-local organization can cope with the task of eliminating finite islands in noise. The "cellular automaton" in [23] can hold a bit of information indefinitely. However, the transition rule is not uniform either in space or time: the hierarchy is "hardwired" into the way the rule changes.

A nonergodic one-dimensional cellular automaton working in discrete time has been constructed in [5], using some ideas from the very informal paper [14] of Georgii Kurdyumov. The paper [6] constructs a two-dimensional fault-tolerant cellular automaton in which the space requirement of the reliable implementation of a computation is only a constant times greater than that of the original version. (The time requirement increases by a logarithmic factor.) In both of these papers, the cells are organized in blocks that perform a fault-tolerant simulation of a second, generalized cellular automaton. The cells of the latter automaton are also organized in blocks, simulating even more reliably a third generalized automaton, etc. Since all this organization is in software, it must be under repair all the time from breakdown caused by errors. In the two-dimensional case, Toom's transition rule simplifies the repairs.

The work reported here makes several advances beyond the results of [5]: the most important ones are *asynchrony* and *self-organization*. In the three-dimensional fault-tolerant cellular automaton of [9], the components must work in discrete time and switch simultaneously to their next state. This requirement is unrealistic for arbitrarily large arrays. A more natural model for asynchronous probabilistic cellular automata is that of a continuous-time Markov process. This is a much stronger assumption than allowing an adversary scheduler but it still leaves some technical problems to be solved. Informally it allows cells to choose whether to update at the present time independently of the choice their neighbors make.

The paper [3] gives a simple method to implement arbitrary computations on asynchronous machines with otherwise perfectly reliable components. A two-dimensional asynchronous fault-tolerant cellular automaton was constructed in [25]. The present paper constructs a one-dimensional asynchronous fault-tolerant cellular automaton, thus completing the refutation of the so-called Positive Rates Conjecture in [16].

Most hierarchical constructions start from a complex, hierarchical initial configuration (in case of an infinite system, an infinite hierarchy). The present paper avoids this when possible. When the computation's input is only a constant amount of information, (as in the refutation of the Positive Rates Conjecture) then we will give a transition rule that performs this task even if each cell of the initial configuration has the same state. We call this "self-organization" since the hierarchical organization will still emerge during the computation.

2. Basic definitions and some results

2.1. Cellular automata

Having the later construction in mind, we define one-dimensional cellular automata in a slightly more general way than usual. The set $\mathbf{C}$ of *sites* is the set $\mathbb{Z}_m$ of remainders modulo m. For $m = \infty$, this is the set $\mathbb{Z}$ of integers. In the finite case, this means *periodic boundary conditions*.

To each cellular automaton, there belongs a positive number B called the *body size*. The interval $[x, x + B)$ will be called the *body of a cell* at site x. A *(space) configuration* is a set of cells with non-intersecting bodies. It can be described by a function $\xi(x)$ for $x \in \mathbf{C}$. Here, $\xi(x)$ is the state of cell with origin x: we assume it to have the symbolic value *Vac* if the configuration has no cell with origin x. For convenience, we include *Vac* into the finite set $\mathbb{S}$ of local states. We will speak of a *lattice configuration* if the cell bodies fit together into a rectangular lattice with starting point 0. In a *cellular automaton*, all configurations are lattice configurations. Later we need configurations in which the bodies do not cover the space.

A *space-time configuration* is a function $\eta(x, t)$ which for each t is a space configuration. We call u a *switching time* of x if $\eta(x, t) \neq s$ for all $t < u$ sufficiently close to u. We require:

(a) $\eta(x, t)$ is a right-continuous function of t;

(b) each finite time interval contains only finitely many switching times of each cell;

A *deterministic cellular automaton*

$$CA(Tr, B, T).$$

is determined by $B, T > 0$ and a transition rule $Tr : \mathbb{S}^3 \to \mathbb{S}$. A space-time configuration η is a *trajectory* of this automaton if it has body size B, switching times at multiples of T, and

$$\eta(x,t) = Tr(\eta(x-B,t-T), \eta(x,t), \eta(x+B,t-T))$$

holds for all x, t with $t \ge T$. For a space-time configuration η let us write

$$Tr(\eta,x,t) = Tr(\eta(x-B,t), \eta(x,t), \eta(x+B,t)).$$

The initial configuration $\eta(\cdot,0)$ and the transition function completely determine the trajectory of a deterministic cellular automaton.

The space-time configuration of a deterministic cellular automaton can be viewed as "computation". We will deal only with cellular automata in which the set $\mathbb{S}$ of states consists of binary strings of some fixed length $\|\mathbb{S}\|$. The information represented by the state can be broken up naturally into parts. Subsets of $\{0,\dots,\|\mathbb{S}\|-1\}$ will be called *fields*. Fields are generally either disjoint or contained in each other. When we join e.g. the corresponding fields of the cells at different sites we can speak about the input *track*, like a track of some magnetic tape.

2.2. Probabilistic cellular automata

A *random space-time configuration* is a pair (μ, η) where μ is a probability measure over some measurable space $(\Omega, \mathcal{A})$ together with a measurable function $\eta(x,t,\omega)$ which is a space-time configuration for all $\omega \in \Omega$. We will generally omit ω from the arguments of η. When we omit the mention of μ we will use Prob to denote it. For a rectangle W, let $\mathcal{A}(W)$ be the σ-algebra generated by events of the form $\{\,\eta(x,t) = s \text{ for } t_1 \le t < t_2\,\}$ for $s \in \mathbb{S}$, $(x, t_i) \in W$. Let

$$\mathcal{A}_t = \mathcal{A}(\mathbf{C} \times [0,t]).$$

A *probabilistic cellular automaton*

$$PCA(\mathbf{P}, B, T)$$

has, instead of a transition function, a *transition matrix* $\mathbf{P}(s,(r_{-1},r_0,r_1))$. For an arbitrary space-time configuration η, and space-time point (x,t), let

$$\mathbf{P}(\eta,s,x,t) = \mathbf{P}(s,(\eta(x-B,t),\eta(x,t),\eta(x+B,t))).$$

Now a trajectory is not a single space-time configuration but a distribution over space-time configurations with body size B and switching times that are multiples of T and that satisfies the following condition. Let $x_0,\dots,x_{n+1}$ be given with $x_{i+1} = x_i + B$. Let us fix an arbitrary space-time configuration ζ and an arbitrary event $\mathcal{H} \ni \zeta$ of positive probability in $\mathcal{A}_{t-T}$. Then we require

$$\begin{aligned}
&\mathrm{Prob}\{\bigcap_{i=1}^{n}\{\eta(x_i,t) = \zeta(x_i,t)\} \mid \\
&\qquad \mathcal{H} \cap \bigcap_{i=0}^{n+1}\{\,\eta(x_i,t-T) = \zeta(x_i,t-T)\,\}\} \\
&\qquad\qquad = \prod_i \mathbf{P}(\eta, \zeta(x_i,t), x_i, t-T).
\end{aligned}$$

A probabilistic cellular automaton is *noisy* if $\mathbf{P}(s,\mathbf{r}) > 0$ for all $s, \mathbf{r}$.

A trajectory of a probabilistic cellular automaton is a discrete-time Markov process. It is a finite Markov chain as long as the number of sites is finite.

Probabilistic cellular automata in which the cells make a random decision "in each moment" on whether to make a transition to another state, are called *continuous-time interacting particle systems*. An overview of many results available for this model in 1985 can be found in [16]. The system is defined by a matrix $\mathbf{R}(s,\mathbf{r}) \ge 0$ of transition rates in which all "diagonal" elements $\mathbf{R}(r_0,(r_{-1},r_0,r_1))$ are 0. Here, we show an elementary construction of the model. Choose some small $\delta > 0$. Let $M_\delta = PCA(\mathbf{P}, B, \delta)$ with $\mathbf{P}(s,\mathbf{r}) = \delta\mathbf{R}(s,\mathbf{r})$ when $s \ne r_0$ and $1 - \delta\sum_{s' \ne r_0}\mathbf{R}(s',\mathbf{r})$ otherwise. (This definition is sound when δ is small enough to make the last expression nonnegative.) With any fixed initial configuration $\eta(\cdot,0)$, the trajectories η_δ of M_δ will converge (weakly) to a certain random process η which is the continuous-time probabilistic cellular automaton with these rates. For a proof that η_δ converges weakly to a continuous-time process η, see [16], [11] and the works quoted there. An interacting particle system is *noisy* if $\mathbf{R}(s,\mathbf{r}) > 0$ for all $s \ne r_0$.

2.3. Perturbation

Intuitively, a deterministic cellular automaton is fault-tolerant if even after it is "perturbed" into a probabilistic cellular automaton, its trajectories can keep the most important properties of a trajectory of the original deterministic cellular automaton. We will say that a random space-time configuration (μ, η) is a trajectory of the ε-*perturbation*

$$CA_\varepsilon(Tr, B, T)$$

of the transition function Tr if the following holds. For all $x_0, \dots x_{n+1}, t$ with $x_{i+1} = x_i + B$ and events $\mathcal{H}$ in $\mathcal{A}_t$

with $\mu(\mathcal{H}) > 0$, for all $0 < i_1 < \cdots < i_k < n$,

$$\mu\{\bigcap_{j=1}^{k}\{\eta(x_{i_j}, t) \neq Tr(\eta, x_{i_j}, t-T)\} \mid \mathcal{H} \cap \bigcap_{i=0}^{n+1}\{\eta(x_i, t-T) = s_i\}\} \le \varepsilon^k.$$

It is best to think of the trajectory of a perturbation as a process created by an "adversary" who is trying to defeat whatever conclusions we want to make about the trajectory, and is only restricted by the above inequalities on the distribution. If we have any probabilistic cellular automaton $PCA(\mathbf{P}, B, T)$ such that $\mathbf{P}(s, \mathbf{r}) \le 1 - \varepsilon$ whenever $s = Tr(\mathbf{r})$ then the trajectories of this are trajectories of $CA_\varepsilon(Tr, B, T)$; however, these do not exhaust the possibilities.

By the *ε-perturbation* of a continuous-time interacting particle system with transition rates given by $\mathbf{R}(s, \mathbf{r})$, we understand the following: in the above construction of a process, we perturb the matrix elements $\mathbf{R}(s, \mathbf{r})$ by some arbitrary amounts smaller than ε.

2.4. Some results

Our general results are about reliable computation, not just information storage. But the simplest (and still challenging) results about information storage deserve a separate statement. Suppose that the cell state has some field F. We will say that Tr *remembers* field F if there is an $\varepsilon > 0$ such that for each string $s \in \{0,1\}^{|F|}$ there is a configuration ξ_s such that for all trajectories (μ, η) of the ε-perturbation $CA_\varepsilon(Tr, 1, 1)$ with $\eta(\cdot, 0) = \xi_s$, for all x, t we have

$$\mu\{\, \eta(x,t).F = s \,\} > 2/3.$$

We define similarly the notions of remembering a field for a probabilistic transition matrix $\mathbf{P}$ and a probabilistic transition rate matrix $\mathbf{R}$. Let us say that remembering the field happens in a *self-organizing way* if the configurations ξ_s can be chosen to be constant: $\xi_s(x)$ does not depends on x. The work [5] implies the following:

Theorem 2.1. *There is a one-dimensional transition function that remembers a field.*

Now there is a continuous-time version, which is technically more difficult:

Theorem 2.2. *There is a one-dimensional transition-rate matrix that remembers a field.*

Both the discrete and the continuous-time versions can also be made to remember a field in a self-organizing way.

3. More general results

3.1. Information storage

In the theorems above, the cellular automata stored only a finite amount of information, even if they were infinite. It is possible, however, to store information in a cellular automaton reliably but so densely that each cell contains a bit. To keep notation simple, we state the result first for infinite cellular automata.

Theorem 3.1. *There are a transition Tr' having a field F, a mapping $\Gamma : \mathbb{S}.F^{\mathbb{Z}} \to \mathbb{S}^{\mathbb{Z}}$, a constant $d_1 > 0$ such that for all $\varepsilon > 0$, $t > 0$, for any configuration ϱ with states in $\mathbb{S}.F$, trajectory η of $CA_\varepsilon(Tr', 1, 1)$ over $\mathbf{C} = \mathbb{Z}$ with initial configuration $\Gamma(\varrho)$, for all y we have*

$$\mathrm{Prob}\{\, \varrho(y) \neq \eta(y,t).F \,\} < d_1\varepsilon.$$

The mapping Γ is given explicitly in the proof and is simple to compute. Essentially, for an appropriate sequence $B_1 < B_2 < \cdots$, we subdivide the configuration into blocks of size B_1 and add some error-check bits in a field different from F, as well as some other periodic information independent of ϱ. Then we repeat this for B_2, B_3, etc., in a process that is guaranteed to converge.

The version of this theorem for a finite space size N also asserts the existence of a constant $c_2 > 0$ such that for $h_2(n) = n^{c_2/\log\log n}$, all $\varepsilon > 0$, $t > 0$, for any configuration ϱ with states in $\mathbb{S}.F$, any trajectory η over $\mathbf{C} = \mathbb{Z}_N$, for all y within $N/2$ of the origin, we have

$$\mathrm{Prob}\{\, \varrho(y) \neq \eta(y,t).F \,\} < d_1\varepsilon + t\varepsilon^{h_2(N)}.$$

Thus, N bits of information can be stored in a one-dimensional cellular automaton of size N for an amount of time that is almost exponential in N.

In all these theorems, a probabilistic cellular automaton can also be replaced with an appropriate continuous-time one.

3.2. Computation

It is convenient to define reliable computation using cellular automata, by "simulating" the trajectory of an arbitrary deterministic cellular automaton. Let us standardize somewhat the automata to be simulated. We call the set

$$\Sigma_0 = \{0, 1, \#, *\}$$

the *standard alphabet*. Symbol $*$ will serve as a "don't-care" symbol, and $\#$ serves as a delimiter. Each field F of a cell state such that the field size is even, can be considered a string of (half as many) symbols in the standard alphabet. If r, s are strings in Σ_0^n then

$$r \preceq s$$

will mean that $s(i) = r(i)$ for all $0 \le i < n$ such that $r(i) \neq *$. This is useful in a field, like a computation output, when at a given time it may happen that only part of the field has been determined yet. Let *Tr* be any deterministic transition function with distinguished fields *Input*, *Output* in the standard input-output alphabet. We say that *Tr* has *monotonic output* if for all trajectories η of $CA(Tr, 1, 1)$ we have

$$\eta(x,t).Output \preceq \eta(x,t+1).Output,$$

further if $\eta(x,t).Input = \#$ then $\eta(x,t+1) = \eta(x,t)$.

Theorem 3.2. *Let Tr be a transition function with monotonic output. There is a transition function Tr' with a field Output, a mapping $\Gamma : \mathbb{S}.F^{\mathbb{Z}} \times \mathbb{R}_+ \to \mathbb{S}^{\mathbb{Z}}$, constants $d_1, c_1 > 0$ such that for $h_1(n) \le n/(c_1 \log n)^{5 \log\log\log n}$, all $\varepsilon > 0$, $t > 0$, for any trajectory ζ of Tr, any trajectory η of $CA_\varepsilon(Tr', 1, 1)$ over $\mathbf{C} = \mathbb{Z}$ with initial configuration $\Gamma(\zeta(\cdot, 0), t)$, for all y we have*

$$\operatorname{Prob}\{\, \zeta(y, h_1(t)).Output \not\preceq \eta(y,t).Output \,\} < d_1\varepsilon.$$

Thus, to find a computation result that would take time u and space s on some universal cellular automaton $CA(Tr, 1, 1)$ with monotonic output, we must do the following. Embed the input into some standard initial configuration ϱ of $CA(Tr, 1, 1)$, in which a segment of size s of the *Input* field is delimited by $\#$. Encode ϱ via $\Gamma(\varrho, t)$ where $t = \min\{\, t' : h_1(t') \ge u \,\}$. Run the trajectory η of $CA_\varepsilon(Tr', 1, 1)$ until time t, and finally look up any bit of the result at the same site y where it would be in $CA(Tr, 1, 1)$.

Instead of a probabilistic cellular automaton, we can use a continuous-time interacting particle system here.

Instead of an infinite lattice, we can use any finite one, with size $\ge 2s$.

The function $t/h_1(t)$ measures *time redundancy*. The "space redundancy" is a constant factor.

The above teorem strengthens the result of [5] also by eliminating the need for the decoding of the computation result.

The mapping Γ is given explicitly in the proof and is simple to compute, just as for the information storage result above. However, now the mapping also depends on the computation time u, in the following way: we stop the process of building the hierarchy of blocks $B_1 < B_2 < \cdots$ after a finite number of steps depending on u.

One would like to see a result here in which the encoding does not depend at all on the time u to which we want to compute. The present framework seems to make this possible, but the actual construction still needs to be carried out (see the open problems at the end).

There are several ways in which a result on self-organizing computation can be formulated, and we believe that the framework developed here supports all results that can be reasonably expected. However, each of these results is somewhat awkward to formulate since if the input is periodic, one must agree on a way to get to the output, which is to be found in somewhat random places.

4. Application to spin systems

4.1. Gibbs states

For the statement of an important implication of the above results, equilibrium states of "spin systems" need to be defined formally. For reference, see e.g. [19]. Let us be given a lattice, say $\mathbb{Z}^2$, and a set $\mathbb{S}$ of local states. A *(partial) configuration* is a (partial) function from $\mathbb{Z}^2$ into $\mathbb{S}$. Let J be a real function over the set of finite partial configurations. For a configuration σ and a finite $\Lambda \subset \mathbb{Z}^2$, we define the *energy* of σ in Λ by

$$E(\sigma, \Lambda) = \sum_{A \cap \Lambda \ne \emptyset} J(\sigma \restriction A).$$

An *equilibrium state*, or *Gibbs state*, with inverse temperature $0 < \beta < \infty$, is a probability distribution over the set of configurations with the property that for all choices of Λ, for all possible values of σ over $\mathbb{Z}^2 \setminus \Lambda$, the probabilities of the various values of σ on Λ are proportional to $e^{-\beta E(\sigma,\Lambda)}$. The best known example is a ferromagnet (idealization of the surface of magnetic disks). Here, $\mathbb{S} = \{-1, 1\}$, $J(\sigma \restriction \{x\}) = \alpha\sigma(x)$ (α is the strength of some external field) for all x, $J(\sigma \restriction \{x, y\}) = -\sigma(x)\sigma(y)$ if $|x - y| = 1$ (maximum norm) and 0 otherwise.

Given an equilibrium system, the most important question concerns its number of equilibrium states (at a given temperature). If there is more than one possible equilibrium state then we speak of *phase transition* (of the first kind). In the ferromagnetic example, it is well known that there is phase transition if and only if $\alpha = 0$ and the temperature is low. Let us call a Gibbs state *extremal* if it is not the convex combination of any other Gibbs states.

4.2. Non-periodic Gibbs states

Even though the most commonly used mass storage device (magnetic disk) is a spin system, a ferromagnetic spin system has been proved in [1] to have only two extremal Gibbs states: therefore a whole disk can store only a single bit of information, even if it is infinitely large. (We store so much more on an actual disk only since the actual interaction also has a long-range component and the latter prefers the formation of finite-size magnetic domains. However, these domains are not stable in the very long run.)

All 2-dimensional spin systems hitherto known have had only a finite number of extremal Gibbs states: thus, theoretically, the amount of information we could store on an $n \times n$ square lattice did not grow with the size of the lattice.

In 3 dimensions this is not true anymore, since we can stack independent 2-dimensional planes: thus, in a cube C_n of size n, we can store n bits of information. More precisely, the information content of C_n can be measured by the dimension of the set of vectors

$$(\mu\{\,\sigma(x) = 1\,\} : x \in C_n)$$

where μ runs through the set of Gibbs states. This dimension can be at most $O(n^{d-1})$ in a d-dimensional lattice, since the Gibbs state on a cube is determined by the distribution on its boundary. The stacking construction shows that storing $\Omega(n^{d-2})$ bits of information is easy. Our contribution is that $\Omega(n^{d-1})$ is achievable: in particular, it is possible to store an infinite sequence in a 2-dimensional spin system in such a way that n bits of it are recoverable from any n sites with different x coordinates.

For this, we apply a transformation from [10] (see also [2]): a probabilistic cellular automaton M in d dimensions gives rise to an equilibrium system M' in $(d+1)$ dimensions. Essentially, the logarithms of the local transition probabilities define the function J and space-time configurations of M become the space-configurations of M'. Non-ergodicity of M corresponds to phase transition in M'. In the cellular automaton of Theorem 3.1, each infinite sequence ϱ gives rise to a space-time configuration storing the bits of ϱ in consecutive cells. Now, each of these space-time configurations gives rise to a separate Gibbs state belonging to one and the same potential.

5. Some open problems

- The main challenge is still to simplify the hierarchical constructions employed in our proofs.
- There are many modes of using a computing device that seem at least as natural as to start it from an encoded input and then to look at its appropriate places at the appropriate times for output. E.g. we could feed input over time at a distinguished site. Self-organization from a configuration that is homogenous except for a single site, should also be investigated.
- As mentioned after Theorem 3.2, it is desirable to make the encoding for computation independent of the computing time. There do not seem to be any obstacles to doing this.
- The Markov updating model differs from the models of asynchrony generally considered in distributed computing, in which the update times of different components can even be set by an adversary. To expect any solution under an adversary update model, we must require that (say, probabilistic) faults can occur only at update events but the times of these events can be set by an adversary.

  As an interesting application, consider whether a fault-tolerant Turing machine can be built if the tape is left undisturbed, only the internal state is subject to faults. This is a special kind of cellular automaton in which only the cell containing the head updates its state.

  The work [13] assumes complete asynchrony and arbitrary initial condition but no noise. However, a closer inspection shows many similarities of the problems solved: therefore we hope to make progress towards a noisy version of [13] and an asynchronous version of the present work.
- An adversary error model should also be investigated. One possibility is to restrict the number of faults in every space-time rectangle of certain dimensions. Our constructions yield some results of this form. Namely, there are constants $\alpha > \beta > 0$ such that given a computation to perform with space complexity S and time complexity T, our machines will perform it reliably if every space-time square (or cube, if the space is 2-dimensional) of size $\log^{\alpha}(ST)$ contains at most $\log^{\beta}(ST)$ faults. This follows from the fact that with a small enough β, every subset of size $\log^{\beta}(ST)$ of a cube of size $\log^{\alpha}(ST)$ is "sparse".

  It is desirable to obtain negative results of this type. (The adversary can clearly not be allowed to make all cells fail at a single instant.)

  Another natural way to alter the fault model is to consider permanent faults in place of transient ones (in dimensions higher than 1 or in other topologies).

6. On the proof method

6.1. Codes

Let x be a cell and Q a positive integer. The set of Q cells $x + iB$ for $i \in [0, Q-1]$ is called the *Q-colony* with *base* x, and cell $x + iB$ is said to have *address* i in this colony. Our cells will have an *address field* *Addr* and $\xi(x).$*Addr* an integer in $[0, Q-1]$. Grouping into colonies may help preserving information if each colony has its information with redundancy. The transition rule should involve the colony members in a coordinated periodic activity, repeated after a period of U steps for some integer U (a *work period*), of restoring this information from the gradual degradation caused by faults. A cell will typically have a field called *Age* $\in [0, U-1]$ showing the number of the current program step being performed by it. We can expect such a transition rule to keep information or even perform computation correctly if no more than one fault occurs during some colony work period. Let us call this a *single-fault-correcting* computation.

Even if a large group of failures makes internal correction no more possible the information is not entirely lost if some of it is repeated over neighbor colonies as well. But using this redundancy requires organization reaching wider than a single colony.

Let us denote by M_1 the fault-tolerant cellular automaton to be built. In this automaton, a colony $\mathcal{C}$ with base x will be involved in two kinds of activity during each of its work periods:

Simulation: of some cellular automaton M_2 with body size QB;

Error-correction: Using the collective information the state of the simulated cell to correct each cell within its colony;

The first activity is directed "upwards", the second one "downwards". The cells of the simulated automaton M_2 will not be immune to errors. But they can also be grouped into colonies simulating an automaton M_3, etc. leading to a *hierarchy of simulations*.

The decoding process $M_1 \to M_2 \to \cdots$ is straightforward. The encoding needs to solve the following (somewhat technical) problems:

- convergence of the "infinite regress" involved;
- achieving a redundancy that is only a constant factor;
- clarifying which simulation level can change which fields, in particular, a way of distinguishing an *active level* in the case of reliable computation.

6.2. Simulation

The basic tools of the proof are the concepts of *medium* and *simulation*. In the most general setting, a medium M is just a way to distinguish random processes that are trajectories from those that are not. A *simulation* of cellular automaton M_2 by M_1 is given by a pair (Φ^*, φ_*) where Φ^* is a mapping of the set of space-time configurations of M_1 into those of M_2 (the decoding), and φ_* is a mapping of the set of configurations of M_2 to the set of configurations of M_1 (the encoding for initialization). Let us denote

$$\eta^* = \Phi_*(\eta).$$

We require, for each trajectory η for which the initial configuration has the encoded form $\eta(\cdot, 0) = \varphi_*(\xi)$, that η^* is a trajectory of M_2 with $\eta^*(\cdot, 0) = \xi$.

We construct a sequence $M_1, M_2, \dots$ of media that are somewhat similar to ε-perturbations in that an error parameter ε_k will belong to M_k, where ε_k converges to 0 rapidly. Along with this, we define a series φ_k^* of simulations, of M_{k+1} by M_k. Thus, medium M_1 simulates the more reliable medium M_2 which simulates the more reliable M_2, etc. A sequence like this is called an *amplifier*.

To each decoding function φ_k^* belongs also an encoding function φ_{k*}. Our amplifiers have an *error-correction* property: in small noise, colonies of M_k re-encode the relevant information in themselves via the encoding functions φ_{k*}, from the cells of M_{k+1} they represent. This is how the "small cells" benefit from the robustness of the "big ones".

6.3. Media

The simulation property defined above is very strong and will only be satisfied if the notion of trajectory (i.e. our medium) is defined loosely. We must allow for all the possibilities (e.g. cells that are not aligned with each other on a lattice) that can arise in a simulated medium when arbitrary combinations of faults occur in the simulating one. Let us call an *event funtion* any function of the trajectories taking values 0 or 1. A *medium* is defined by a set $\mathbb{S}$ of states, a set $\mathbb{T}$ of *condition types*, and functions $b : \mathbb{T} \to [0, 1]$ and $g(\alpha, V, \eta)$. Here g is an event function for each $\alpha \in \mathbb{T}$ and each rational rectangle V in space-time, and the pair b, g expresses for each α and V a condition $\mathbf{E}g(\alpha, V, \eta) \le b(\alpha)$, where $\mathbf{E}$ denotes the expected value. If $V_1, V_2, \dots$ are disjoint then we require that the bounds on the probability of violating individual conditions multiply:

$$\mathbf{E}(\prod_i g(\alpha_i, V_i, \eta)) \le \prod_i b(\alpha_i).$$

A similar relation is required even if the rectangles themselves are chosen randomly—however, this choice must happen according to a "random stopping time", (i.e. without lookahead in time). It can be shown that both discrete- and continuous-time cellular automata give rise to media in natural ways.

We define a special class of simulations between media M_1, M_2 called *patent simulations*, which will automatically satisfy the simulation property. The defining property is that for each local condition (α, V) of M_2 there is a system of local conditions

$$\big(\, \beta(\alpha, i, j), W(\alpha, i, j, V) \,\big)$$

of M_1 with $i = 1, \dots, n(\alpha)$, $j = 1, \dots, k(\alpha, i)$, such that for each fixed i, the rectangles $W(\alpha, i, j, V)$ are disjoint and

$$g_2(\alpha, V, \eta^*) \le \sum_i \prod_j g_1(\alpha, W(\alpha, i, j, V), \eta),$$

$$b_2(\alpha) \ge \sum_i \prod_j b_1(\beta(\alpha, i, j)).$$

Just as conditions must be generalized to allow for randomly chosen rectangles, there is a corresponding generalization of the notion of patent simulation. The simulations of the hierarchical construction are patent simulations.

6.4. Robust media

The media actually used in our amplifiers will have a special structure and will be called *robust media*. These have a special set of states *Bad* $\subset \mathbb{S}$. For a space-time configuration η we define the *damage set*

$$\mathit{Damage}(\eta) = \{\, (x,t) : \eta(x,t) \in \mathit{Bad} \,\}.$$

The interpretation of the set $\mathit{Damage}(\eta)$ is that when $\eta(x,t) \in \mathit{Bad}$ then in the neighborhood of (x,t), we will not be able to make any predictions of η, i.e. in some sense, η behaves completely "lawlessly" there.

According to the general definition of a medium, the set of trajectories will be defined by a pair $b(\cdot), g(\cdot,\cdot,\cdot)$. For a medium with cellsize B, and dwell-period lower-bound $T_\bullet$ (these are parameters and there will be some others, too, e.g. a dwell-period upper bound $T^\bullet$, an error bound ε and a transition function *Tr*), let

$$V = [-B/4, B/4) \times [-T_\bullet/4+, 0].$$

We formulate the conditions $g(\cdot,\cdot,\cdot)$ in terms of two *axioms*. The Restoration Axiom, which depends on a new parameter ε, requires that for each window of the form $(x,t)+2V$, the probability that there is damage in $(x,t)+V$ is at most ε.

Consider a simulation $\eta^* = \Phi^*(\eta)$ between two robust media. We define the damage set of η^* to be used in such simulations essentially as follows: we say $\eta^*(x,t) \in \mathit{Bad}$ iff $\mathit{Damage}(\eta)$ contains at least two points u, v such that $u+2V, v+2V$ are disjoint and contained in $(x,t)+4V^* + (B^*/2, 0)$. Thus, if $\eta^*(x,t) \neq \mathit{Bad}$ then essentially all damage in $(x,t) + 4V^* + (B^*/2, 0)$ can be covered by a single rectangle of the form $(y,u)+2V$. This will be called the *damage rectangle*.

The value of η^* provided it is not in *Bad*, will be essentially defined by a block code φ as

$$\varphi^*(\eta(x + [0, QB-1], t)).$$

However, there will be some look-back for stabilization.

The other axiom for robust media is called the Computation Axiom. It essentially says that in case there is no damage the trajectory must obey the transition function. The actual formulation of the axiom is rather technical, for the following reasons.

- the sense of obeying a transition function by a continuous-time trajectory, must be defined;
- communication between non-adjacent cells must also be considered;
- cells occasionally disappear or will be born, and the sense in which these events obey the transition function must be carefully circumscribed;

It can be shown that if

$$\varepsilon^* \geq 25((B^*/B)(T_\bullet{}^*/T_\bullet)\varepsilon)^2$$

then Φ^* is a patent simulation map.

6.5. **Amplifiers**

Our goal is an amplifier (M_k, Φ_k) with a fast decreasing sequence ε_k and we also want the robust media M_k to simulate a computation with a universal transition function.

Unlike the simulations used in universal Turing machines, the simulations Φ_k must not have any freely choosable parameters like programs, other than what is computable from the transition rule of M_1. (The construction of [6] allowed such parameters but this does not seem permissible in one dimension.) We call such simulations *hard-wired*. We do not lose universality this way since the transition function Tr_k of each of the media M_k can be that of a universal cellular automaton capable of many simulations: only the simulation Φ_k is hard-wired.

There will be a field called Rider^k where the "useful" computation of M_k will be carried out, on the "active level" (see above). An *amplifier frame* is a sequence of parameters (everything but the transition function Tr_k) for the media M_k and also a sequence $\mathit{Rd\_tr}_k()$ defining the transition function for the field Rider_k in each M_k.

The main lemma of the construction, the *Amplifier Lemma*, says (essentially) that if the sequences in some amplifier frame *Frame* satisfy some reasonable bounds on their size and complexity then there is an amplifier having the parameters specified in *Frame*. This lemma is then applied to the proof of the theorems by choosing the parameters and in particular the transition functions $\mathit{Rd\_tr}_k()$ appropriately and by exploiting the simulation and error-correction properties of the amplifiers.

6.6. **Outline of the program**

What was described above can be considered the "mathematical shell" of the construction. It must be "filled out" by the proof of the amplifier lemma, and this requires the actual definition of the "program" (the transition function Tr_k) of the robust medium M_k, followed by the proof that if M_k satisfies the Computation Axiom then so does M_{k+1}. The program is best presented as a collection of *rules*, applicable to various situations, and governing various fields.

Recall from the definition of damage for simulations in robust media that if $\eta^*(x,t) \neq \mathit{Bad}$ then essentially all damage in $(x,t)+4V^*+(B^*/2,0)$ can be covered by a single small rectangle of the form $(y,u)+2V$ called the *damage rectangle*. Since the Computation Axiom for M_{k+1} needs to be satisfied only in the absence of damage in M_{k+1}, it looks like only some single-fault-correcting computation, mentioned in Subsection 6.1 above, is needed. This is not so since we cannot assume that there has never been larger damage in the past. Therefore the program must lead to a satisfaction of the Computation Axiom even when it starts from an arbitrary initial configuration, in which cells are not neatly arranged into colonies. This setup is similar to problems of *self-stabilization*. Typical results on self-stabilization as [13] assume arbitrary initial configuration; however, they also assume error-free operation. Our main rule (called *Decay*) to achieve self-stabilization is very simple in principle: it says that if a cell sees incompatible neighbors for a while then it dies. This rule kills partial colonies. (Basically the requirement of local consistency just means that *Addr* and *Age* vary continuously. But in practice, due to various legitimate processes of growth and shrinking, we must define carefully when we call an edge *exposed*, i.e. subject to decay.)

The main lemma showing self-stabilization is called the *Attribution Lemma*: it shows that if Damage^* is empty then each live cell can be attributed to a whole colony, so the evolution can be discussed in terms of the interaction of whole colonies. This lemma can be proved before all the computation rules have been defined: the proof only requires the rules having to do with the creation and destruction of cells. A decay rule is sufficient for self-stabilization because of the "hard-wired" property of the simulation mentioned above. This implies namely that once a colony of cells of M_k is locally consistent it can be relied on to simulate a cell of M_{k+1}.

The program contains some predictable rules concerned with single-error-correcting computation in case the cells can be assumed to be organized in colonies. Even here, there is a new question (with respect to [5] and [6]): how to organize the computation asynchronously. Our solution is to require *Age* to vary continuously within each colony. There is no such coordination across colony boundaries, therefore the global computation program has some safeguards making sure that the information obtained from neighbor colonies does not represent some transitional situation.

Other rules worth mentioning are the rule *Heal*, to make local repairs, and the rule *Grow*, which allows a colony to create a new colony on an adjacent empty area. To avoid the accidental killing of a colony, cells have different values of *strength*: regular member cells are stronger than growth cells and can kill the latter ones during their *Heal*. The weakest cells are called *latent*: they essentially serve as place-holders and are erased only if place is needed for another cell. ("Killing" a cell generally only means turning it latent.)

6.7. Self-organization

When the initial configuration is homogeneous only a constant amount of information can be expected to be remembered. Let us call the field to be remembered the *Color* field. In what follows, it is convenient to speak of some particular value of this field: let it be called "blue".

When a cell turns from a vacant to a non-vacant state, we can distinguish two ways in which this can happen. The first one is if the cell has a time with no non-vacant neighbor shortly before the event: let us call this *spontaneous birth*; the second one is when it does not: let us call this *creation*. Self-organization will happen via "spontaneous" births, but in a controlled way. If the initial configuration has consisted of blue cells of M_1 then blue cells of M_2 will emerge after a while, then later blue cells of M_3, etc. To prevent the spontaneous birth of cells of other color, we make sure that everything is tightly populated by blue cells. This motivates the following definitions, which we give only approximatively: namely, "nearby" means within a distance of $O(B)$, and "in the near past" within time $O(T_\bullet)$.

In a space-time configuration η of M, a point (x, t) is said to be *blue* if a certain small neighborhood of it in the near past contains a cell and contains only blue cells. A medium has the *lasting control* property if under damage-free conditions, a blue area remains blue. Let M_1, M_2 be robust media, and Φ a simulation between them. We say that this simulation has *control delegation* property if blueness under M_2, in the absence of damage in M_1, implies blueness under M_1.

For some parameters $D, \sigma > 0$, we will say that the trajectory (μ, η) of medium M is (D, σ)-*blue* at time t if the following holds. Let $I_1, \dots, I_n$ be any system of intervals of size D such that they are separated at least at a distance $2D$. Then the probability that each of them is non-blue in the near past of time t, is at most σ^n.

Let us be given an amplifier, constants C_1, C_2, κ_1, and a sequence $\sigma_k > 0$ with $\sigma_k U_k \to 0$ which also does not go too fast to 0. We will call this a *self-organizing amplifier* if the following properties hold:

(a) all media in it have the lasting control property;
(b) each simulation has the control delegation property;
(c) if trajectory (μ, η) of M_k is $(C_1 B_k, \sigma_k)$-blue at time t, then trajectory $(\mu, \Phi_k^*(\eta))$ of M_{k+1} is $(C_1 B_{k+1}, \sigma_{k+1})$-blue at time $t + C_2 T^\bullet{}_{k+1}$;

Now, the Amplifier Lemma can be modified to assert the existence of a self-organizing amplifier, and then this can be used to prove the theorems on self-organizing information storage and computation.

Properties (a) and (b) (as a proof shows) are already satisfied by the original construction. To satisfy property (c), a special sort of cell, called *germ cells*, is introduced, that is stronger than latent cells but weaker than all others. A *germ* starts out from a single cell called its *leader* and tries to grow into an area covering 3 colonies by growing right. If a conflict with another growing germ cell is discovered a random choice will be made.

More precisely, the germ growth rule is based on *work periods* just as colonies. Each work period is associated with an integer called the *level*, whose value is stored in a field *Level* of each cell of the germ. A level s germ is expected to stretch to a length $\geq 2^s$. Its right edge cell notices if this is not so and becomes exposed (subject to decay). The work period of level s lasts $O(2^s)$ steps.

Here are the rules using these properties. Higher-level (not necessarily adjacent) germ cells on the left with the same color have priority, and will kill a germ cell. Germ cells of the same color and level are distinguished using a field *Active* $\in \{0,1\}$: an active germ cell on the left has priority. The work period is divided into two parts: *growth and computation*. In growth, the right end attempts to grow to size 2^{s+1}. The computation part checks whether the germ has reached the desired size. If yes then in the next work period, each cell increases its level. Otherwise, the leader cell decides by a random choice whether the germ will be active or passive in the next work period.

References

1. Michael Aizenman, *Translation invariance and instability of phase coexistence in the two-dimensional Ising system*, Comm. Math. Phys **73** (1980), no. 1, 83–94.
2. Charles H. Bennett and Geoffrey Grinstein, *Role of irreversibility in stabilizing complex and nonergodic behavior in locally interacting discrete systems*, Physical Review Letters **55** (1985), 657–660.
3. 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.
4. R. L. Dobrushin and S. I. Ortyukov, *Upper bound on the redundancy of self-correcting arrangements of unreliable elements*, Problems of Information Transmission **13** (1977), no. 3, 201–208.
5. Peter Gács, *Reliable computation with cellular automata*, Journal of Computer System Science **32** (1986), no. 1, 15–78.
6. ______, *Self-correcting two-dimensional arrays*, Randomness in Computation (Silvio Micali, ed.), Advances in Computing Research (a scientific annual), vol. 5, JAI Press, Greenwich, Conn., 1989, pp. 223–326.
7. ______, *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.
8. Peter Gács, Georgii L. Kurdyumov, and Leonid A. Levin, *One-dimensional homogenuous media dissolving finite islands*, Problems of Inf. Transm. **14** (1978), no. 3, 92–96.
9. 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.
10. Sheldon Goldstein, Roelof Kuik, Joel L. Lebowitz, and Christian Maes, *From PCA's to equilibrium systems and back*, Commun. Math. Phys. **125** (1989), 71–79.
11. Lawrence F. Gray, *The positive rates problem for attractive nearest neighbor spin systems on* **Z**, Z. Wahrscheinlichkeitstheorie verw. Gebiete **61** (1982), 389–404.
12. ______, *The behavior of processes with statistical mechanical properties*, Percolation Theory and Ergodic Theory of Infinite Particle Systems, Springer-Verlag, 1987, pp. 131–167.
13. Gene Itkis and Leonid Levin, *Fast and lean self-stabilizing asynchronous protocols*, Proc. of the IEEE Symp. on Foundations of Computer Science, 1994, pp. 226–239.
14. G. L. Kurdyumov, *An example of a nonergodic homogenous one-dimensional random medium with positive transition probabilities*, Soviet Mathematical Doklady **19** (1978), no. 1, 211–214.
15. A. V. Kuznetsov, *Information storage in a memory assembled from unreliable components*, Problems of Information Transmission **9** (1973), no. 3, 254–264, Translated from Russian.
16. Thomas M. Liggett, *Interacting particle systems*, Grundlehren der mathematischen Wissenschaften, vol. 276, Springer Verlag, New York, 1985.
17. Kihong Park, *Ergodicity and mixing rate of one-dimensional cellular automata*, Ph.D. thesis, Boston University, Boston, MA 02215, 1996.
18. Nicholas Pippenger, *On networks of noisy gates*, Proc. of the 26-th IEEE FOCS Symposium, 1985, pp. 30–38.
19. David Ruelle, *Thermodynamic formalism*, Addison-Wesley, Reading, Massachusetts, 1978.
20. Daniel A. Spielman, *Linear-time encodable and decodable error-correcting codes*, Proc. of the 27th ACM STOC Symposium, 1995, pp. 388–397.
21. ______, *Highly fault-tolerant parallel computation*, Proc. of the 37th IEEE FOCS Symposium, 1996, pp. 154–163.
22. 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.
23. Boris S. Tsirel'son, *Reliable information storage in a system of locally interacting unreliable elements*, Interacting Markov Processes in Biology (V. I. Kryukov R. L. Dobrushin and A. L. Toom, eds.), Scientific Centre of Biological Research, Pushchino, 1977, In Russian. Translation by Springer., pp. 24–38.
24. 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.
25. Weiguo Wang, *An asynchronous two-dimensional self-correcting cellular automaton*, Ph.D. thesis, Boston University, Boston, MA 02215, 1990, Short version: *Proc. 32nd IEEE Symposium on the Foundations of Computer Science*, 1991.