[KUW] R. M. Karp, E. Upfal, and A. Wigderson, "A fast parallel algorithm for the maximal independent set problem," *24th STOC* 266–272 (1984).
[KW] R. M. Karp and A. Wigderson, "A fast parallel algorithm for the maximal independent set problem," *24th STOC* 266–272 (1984).
[Lu] M. Luby, "A simple parallel algorithm for the maximal independent set problem," *17th STOC* (1985).
[Ma] G. A. Margulis, "Explicit construction of graphs without short cycles and low density codes," *Combinatorica* 2(1): 71–78 (1982).
[Ru] W. L. Ruzzo, "On uniform circuit complexity," *J. Comput. Sys. Sci.* 22(3): 365–383 (1981).
[RT] J. H. Reif and J. D. Tygar, "Towards a theory of parallel randomized computation," TR-07-84, Aiken Computation Lab., Harvard University, 1984.
[Si] M. Sipser, "A complexity theoretic approach to randomness," *15th STOC* 330–335 (1983).
[Sh] Shamir, "On the generation of crytographically strong pseudo random sequences," *8th ICALP*, Lecture notes in *Comp. Sci.* 62: 544–550. Springer-Verlag, Berlin, 1981.
[St] L. Stockmeyer, "The complexity of approximate counting," *15th STOC* 118–126 (1983).
[VV] L. G. Valiant and V. V. Vazirani, "NP is as easy as detecting unique solutions," *17th STOC* (1985).
[Ya] A. C. Yao, "Theory and applications of trapdoor functions," *23rd FOCS* 80–91 (1982).

SELF-CORRECTING TWO-DIMENSIONAL ARRAYS

Peter Gacs

ABSTRACT

We continue the program begun in two earlier works: to find simple and efficient ways to implement computations of arbitrary size by a homogeneous array of unreliable elementary components. The homogeneity of cellular arrays makes the *maintenance* of the error-correcting *organization* the biggest technical problem. This problem could be completely avoided in the earlier three-dimensional construction, where almost no structure was needed. All error correction was performed using Toom's voting rule. The structure maintenance problem dominates the earlier one-dimensional construction since there is no Toom's Rule in one dimension. Here, we present a two-dimensional solution, desirable for several reasons. Information maintenance is no longer possible by Toom's Rule, forcing us to adopt a hierarchical design. But Toom's Rule is still applicable to

Advances in Computing Research, Volume 5, pages 223–326.

ISBN: 0-89232-896-7

structure maintenance, giving a significant complexity-dimension trade-off. The second achievement of the present work is the constant space redundancy. Despite the noise in its elementary components, our computing device has a positive capacity as a "communication channel" from input to output. Great care was given to terminology and structured presentation to create a reference point on which further developments (e.g., the introduction of asynchrony) can be built. The first third of the paper serves as an informal overview of the subject.

1. ASSUMPTIONS

There are some general assumptions underlying our investigation that distinguish it from many other studies addressing problems of fault-tolerant computing. These assumptions are as follows.

- We have to build computing devices of *arbitrary size* from a few types of *elementary components*.
- Faults occur in each component *independently* at different space–time positions. Faults thus do not conspire: only the computation creates dependence between the results of faults.
- Faults are transient: they change the local state but not the local transition rule.
- The probability of component faults is bounded by some small parameter ϱ independent of the size of the computing device.
- There are no other restrictions on the type of fault that can occur. Thus, faults can cause arbitrary local state changes.

The goal is to find architectures coping with all combinations of faults likely to arise for devices of a given size. We also want to estimate the needed sacrifice in time and memory to make a computation reliable.

1.1. Transient Faults

The requirement that the faults be transient restricts us severely. This is the kind of fault that is only a one-time violation of the rule of operation. It does not incapacitate the component permanently in which it occurs. In the following steps, the component obeys its rule of operating again, until the next fault.

The toughness of the condition becomes clear if we try to *implement* each component of a theoretical device by a small network of transistors. In a typical implementation, the states of the component are mapped into *some* configurations of the network. But most configurations will be illegal. Therefore the fault of one transistor brings the network probably into an illegal configuration not representing any state of the implemented component. It is better therefore to understand under implementation something like a *computing crystal*: a compound whose transition rules are determined by physical law. It is unlikely, of course, that some chemist will find or synthetize a compound with exactly the same transition laws that we propose. Our constructions must be viewed as *existence proofs*. However, they help to clarify what properties of the local transition rule are needed for reliable computation.

Transient faults are not an exotic kind. Their main source is thermal noise that becomes more significant as the scale of physical devices goes down.

The case of nontransient component fault is of practical importance, but it has not been investigated in the same generality. There are reasons to believe that many of the techniques developed for the case of transient faults will be applicable to the case of permanent faults.

1.2. Parallel Architectures

A reliable computer, all of whose components are unreliable, must use massive parallelism. Indeed, information stored anywhere during computation is subject to decay and therefore must be actively maintained.

Problems concerning serial computers are of less "ideological purity," since they must except a part of the device from the influence of faults. Still, the following problem seems technically very challenging. Manuel Blum asked whether reliable computation is possible with Turing machines in which only the internal state is subject to faults: the tape is safe. Presently, it seems that the solution of this problem requires a construction analogous to the one given in [G].

1.2.1. *Inhomogeneous Devices*

von Neumann [VN] designed a Boolean (acyclic) circuit that works reliably even if its components are unreliable. In his model,

each component had some constant probability of fault. For a deterministic circuit consisting of N components, he built a circuit out of $O(N \log N)$ stochastic components, computing the same function with large probability. (For an efficient realization of his ideas, see [DO2].) The key element of his construction is a *restoring unit*. This unit is supposed to *suppress the minority*. If among its n Boolean input values fewer than $0.3n$ are different from the majority then among the n Boolean output values, fewer than $0.1n$ values differ, even if each component of the restoring unit fails with probability 0.01. The restoring unit of [DO2] achieving this consists of several layers of majority gates. The inputs for each such gate are three randomly selected sites of the previous layer.

The redundancy factor $\log N$ in von Neumann's construction seems to be essential. The number N can be called the *size* of the computation. In other models of computation, it corresponds to the number of elementary events (including information storage) during the whole computation, i.e., roughly to the product of the number of components by the number of computational steps. This estimate is a little weak if the only goal of the computation is information storage.

For purposes of information storage, it is better to use the model of *clocked circuits*. These are Boolean circuits in which special one-bit memory elements called *shift registers* are also permitted. Cycles are permitted but each cycle must contain at least one shift register.

We cannot store even one bit in n registers of such a circuit for longer than 2^n steps, since during that time quite probably a step will occur in which the content of all cells changes simultaneously. But the paper [Kuz] showed that once we have n registers, it is possible to store there $c \cdot n$ bits for 2^{cn} steps, where the positive c depends only on the fault probability of the individual components. This work used Gallagher's low-density parity-check codes, as proposed in [Ta], but improved both in performance and proof by Pinsker.

1.2.2. *Cellular Automata*

The above constructions suffer from the same *theoretical flaw*: the circuits use a rather intricate connection pattern that cannot be realized in three-dimensional space with wires of constant length. On the other hand, the natural assumption about a wire is that as its length grows, its probability of fault converges to 0.5.

A cellular space (medium) is a lattice of automata in, say, three-dimensional space where every automaton takes its input from a few of its closest neighbors. First introduced by von Neumann and Ulam, such devices are now sometimes known as "systolic arrays," or "iterative arrays." Mathematical physicists use the more general term *interactive particle system*.

Typically, all automata are required to have the same transition function and are connected to the same relative neighbors, i.e., the device is translation invariant. The spatial uniformity suggests the possibility of especially simple physical realization. It reduces the number of ingredients that are assumed unchangeable by faults to just one: the transition rule.

Cellular media are desirable computing devices, and it is easy to construct a one-dimensional cellular space that is a universal computer. Let us indicate how a one-tape Turing machine can be simulated by a cellular array, since these machines are better known to be universal.

A Turing machine is defined to have an infinite array of tape cells and a head. Each cell can be in a finite number of possible states, and so can the head. In each instant, the head is scanning a cell. Depending on the state of the scanned cell and its own state, the head makes one of three possible movements (left, right, stay), and changes its own state and the state of the scanned cell. The rule of operation of the machine describes this dependence.

In a possible simulation, the cellular array simulating the Turing machine has one cell corresponding to each tape cell of the Turing machine. A state of a new cell indicates the state of the represented tape cell, the fact whether the head is scanning the tape cell and if yes, what is the head's state. The transition rule of this simulating array is easy to derive from the transition rule of the simulated Turing machine.

2. PHILOSOPHICAL DIGRESSION

2.1. Notions of Complexity

Our remarks make use of several very different notions of *complexity*, and it will be important to keep them apart. Some of these notions have now very adequate mathematical definitions, while some other ones do not.

2.1.1. *Descriptional Complexity*

Kolmogorov and Solomonoff introduced the notion of *descriptional complexity*. In its current version (see [Le]), the complexity $K(x)$ of a finite string can be defined as the length of the shortest self-delimiting program needed to direct some fixed univeral computer to output x. This complexity could be also called "information content," or "measure of disorder," "measure of randomness," or "individual entropy." According to this notion, a typical string of n bits arising from a coin-tossing experiment is maximally complex. Let $\text{Prob}_t(x)$ be the probability to obtain x as output on some fixed universal Turing machine [the same one used in the definition of $K(x)$] in t or fewer steps from a coin-tossing input. The relation $K(x) = -\log \text{Prob}_\infty(x) + O(1)$ is known (see [Le]).

2.1.2. *Computational Complexity*

In the Theory of Computation, another notion of complexity has widespread use. This complexity is the property of a function. The function $g(x)$ is a lower bound on the time complexity of function $f(x)$ on Turing machines if for each Turing machine T there is a positive constant c such that for all x, T takes at least $cg(x)$ steps to compute $f(x)$.

2.1.3. *Logical Depth*

Descriptional complexity helped define individual information and individual randomness. It also helps define a notion that can be considered the "individual computational complexity" of a finite object. Some objects are clearly the result of long development (computation), and are extremely unlikely to arise by any probabilistic algorithm in a small number of steps. This property of objects was formally defined by Bennett as follows (see [B]).

$$\text{depth}_\varepsilon(x) = \min\{t\colon \text{Prob}_t(x)/\text{Prob}_\infty(x) \geqslant \varepsilon\}.$$

Thus, the depth of a string x is at least t with confidence $1 - \varepsilon$ if the conditional probability that x arises in t steps *provided it arises at all* is less than ε.

2.1.4. *Conceptual Complexity*

There is also an informal notion of *conceptual*, or *organizational complexity* that will certainly be applied to the present paper to some extent, and to [G] to a great extent. It will probably never be formally defined, though it may be related to an informal version of $\text{depth}_\varepsilon(x)$ defined above, with the length of x too small to be amenable to formal treatment.

2.2. Noise Resistance and Biological Organization

Even serious scientists indulge sometimes in speculations about "what is life." The goal is to formulate some abstract criteria that among the existing structures in nature, only biological (or, possibly, social) systems satisfy. It is not easy to find such criteria: self-reproduction is clearly insufficient. Answers to this type of question may influence our perspective on some of the more immediate problems of science.

Work on fault-tolerant computation led us to the conjecture that the requirement to *perform in a noisy environment* forces rather elaborate structures, with some resemblance to the ones in biological systems. Two extremal cases of the "performance" we have in mind are *information storage* and the *production of depth*.

Much of the apparent conceptual complexity of biological systems might be due to the simple requirement of reliable information storage. Here is an informal argument in favor of this (informal) thesis: without systems of biology or human society, there seems to be no method to save 300 bits of information from decay somewhere in the universe for 10^9 years. But life has preserved much more information in the DNA for longer.

A similar thesis could be stated about systems starting with zero information but producing structures of ever increasing logical depth, in a noisy environment. Again, only systems involving biology or society seem to be capable of this.

The above general observations are in contrast with most early experimental and theoretical work on cellular automata. The physical investigations were concerned with systems whose convergence to equilibrium could be taken for granted, and only the properties of these equilibria were of interest. The computational applications built structures with logical depth. Examples are algorithms involving systolic arrays, or the structures to simulate

arbitrary computations by simple rules like Conway's Game of Life [BCG] or Margolus's Billiard Ball rule [M]. However, if faults are permitted then all structures built in these computational applications turn rapidly into shallow chaos.

We are interested to find out what properties of the elementary building blocks can help counteract chaos.

2.3. On the Need for Proofs

We deal with media whose reliability can be proven mathematically. Nobody suggested yet candidates for reliable media found by experimentation. Simple reliable—or almost reliable—media without proofs probably exist: such media might underly biological systems. But elementary constructions coupled with rigorous elementary mathematical analysis yielded more information for the present research than experimentation.

One reason (suggested by the reviewer) for the lack of more experimentation with error-correcting schemes in the model we are considering is that the fault probabilities of the elementary components are generally very small, even if constant. Experiments could therefore take a long time before an interesting effect turns up.

3. A THREE-DIMENSIONAL RELIABLE MEDIUM

3.1. Toom's Results

The first questions about the possibility of storing information in arbitrary interactive particle systems were asked in the context of *ergodicity*. Without giving a formal definition here, we recall that an ergodic system converges to a unique statistical equilibrium independent of the starting configuration. Such a system therefore cannot store even *one bit* of information, and cannot increase depth too much.

It was a nontrivial result of Toom that there are infinite two-dimensional cellular media that can store reliably a bit of information. Let us assume that in the initial configuration, each cell has the same state s. We try to limit the probability of a state different from s in each cell at all later steps. It is not easy to find "voting" rules that achieve this, and even if we find one, it is not easy to prove that it works. This is what Toom did; moreover, he characterized all such *monotonic* rules.

For the rest of the paper, we choose one rule in the above family. To determine the state of a cell in the next step, this rule takes the majority of its northern, southwestern and southeastern neighbors. In what follows we will refer to this rule simply as *Toom's Rule*.

Toom's proof in [Too] uses an elaborate topological argument that we could not adapt to an efficient finite version of the theorem. A new proof technique, the technique of "k-noise," was developed in [G] and [GR]. It gave a more straightforward proof of Toom's theorem with efficient finite implications. It is also the only probabilistic tool of the present paper. Mathematical physicists point out that it belongs to the broad class of techniques known under the name *renormalization*.

3.2. Application to Computation

Let D be any one-dimensional medium (cycle) of size K working for L steps. We can simulate its work reliably using a three-dimensional medium D' on a torus of size $K \times d \times d$ where $d = \log^{1+\varepsilon}(KL)$. Each trajectory $x[t,n]$ of D (the function giving the state of cell n at time t when the medium evolves according to the rule D) is mapped into a trajectory $z[t,n,i,j]$ of D' by

$$z[t,n,i,j] = x[t,n]. \tag{3.1}$$

Thus, each symbol of D is repeated over a whole two-dimensional slice of D'. In each step, the transition rule of D' uses first Toom's Rule within the slices then rule D across the slices. Without errors, (3.1) holds for all t,n,i,j. It is shown in [GR] that with errors, the same relation will still hold with large probability. The space requirement of the reliable computation, compared to the original one, is increased by the factor d^2. The medium D' will simulate D reliably step-for-step, without any time delay.

This simulation is the simplest design for reliable computation ever proposed. Its simplicity seems to depend on some special circumstances, especially on a too high number of dimensions, and on synchrony. When these do not hold, we have to rely on much more complicated models.

No attempt was made in [GR] to maximize the error probability permissible for reliable computation. Bennett's experiments on Toffoli's Cellular Automata Machine give convincing empirical evidence that Toom's medium is nonergodic at error probabilities

below 0.05. Piotr Berman and Janos Simon, using Toom's original proof, brought error bound needed for the proof to 10^{-7} (see [BS]).

3.2.1. *Heat Production*

There are some physical reasons to look for lower dimensional error-correcting media. A three-dimensional error-correcting medium is thermodynamically unrealizable. Indeed, any error-correcting operation is inherently irreversible. Such an operation turns a certain minimum amount of free energy into heat. Therefore each cell needs a whole private "supply line" for feeding it with free energy and relieving it of the produced heat. This is possible only if the medium is at most two-dimensional.

3.2.2. *Synchrony*

Toom's Rule keeps a bit of information in a two-dimensional medium even if the cells do not all fire at the same time. But the three-dimensional computing medium defined above is heavily dependent on the synchronous operation of all cells within a cross section. All simple synchronization techniques that we are aware of interfere with error-correction.

4. HIERARCHY OF SIMULATIONS

4.1. Coding

Reliable computation always requires coding with redundancy. Without it, part of the input is lost in the first step, and part of the output in the last step. During the computation therefore, information must live in encoded form.

In general, reliability seems to require that we break up the task into manageable pieces in space and time, and, with each subtask, we go through a cycle of decoding, computation and encoding. Each cycle is repeated several times, since an error can ruin the work of the whole cycle.

In the three-dimensional case, we enjoyed the exceptional situation that the orthogonality of the computation to the repetition code permitted immediate parallel access to each bit of the code. The repetition needed to counter errors in the computation happens to

be identical to the repetition used as redundancy in coding. Due to the lack of decoding or transport, no special structures are required to support the error-correcting activity.

In the general case, information needs to be transported from storage to the place where it is processed. There, it must be decoded. After processing, it must be transported back for storage. All this needs elaborate organization devoted to the task of error correction. In the homogeneous media we are discussing now, this organization can exist only as "software," just as perishable as the rest of information, and it needs maintenance.

The program also must have some *continuity property*: small errors have small consequences. This property will be achieved by insisting that in any one phase of the program, only a small part of the information can be changed. To enforce this requirement, it will be necessary to keep track of phase and relative position by especially robust methods.

4.2. Increasing Reliability by Repeated Decoding

In a computation of size N there will be arbitrary groups of faults of size of the order of $\log N$. Reliable computation must be organized in such a way that no crucial piece of information is committed to a number of cells smaller than $\log N$. Whatever coding we use it must operate with units greater than $\log N$. The tasks of decoding, shipping, etc. with these units are large-scale computational tasks in themselves that must be organized reliably.

The greater units can be derived mathematically only from the primitive "physical" cells of our original medium M, using decoding. Decoding transforms the original probability distribution in M into a different one, in which the great units will operate much more reliably than the original cells.

We could view the greater units as a new kind of cell with a large state space (always as large as needed). It is more advantageous to view them as *colonies* built of the cells of a standard universal medium Univ. From colonies of a given size P, we will arrive at colonies of a larger size P^* and higher reliability in the following way. We choose a certain parameter Q, group the colonies in arrays of size Q by Q (called *alliances*), and apply the decoding of a certain error-correcting code to the configuration found in the resulting array of size QP by QP.

Hierarchy was used in the context of inhomogeneous one-dimensional cellular arrays in [Ts]. The transition rule of the cells was required to vary hierarchically in space and time, to organize a voting scheme, with the sole purpose of remembering one bit of information.

The paper [Kur] made a rather elaborate proposal for such a hierarchy of simulations. Several ideas of this proposal were used in [G].

5. MAINTENANCE OF THE ORGANIZATION

5.1. Structure Maintenance

Above, we referred to a hierarchy of simulations that we had to use for reliable computation in dimensions 2 and 1. On each level of the hierarchy, for some parameters P, Q, P^*, colonies of size P ("small colonies") are grouped in a Q by Q array (the alliance) to simulate a colony of size P^* (a "big colony"). All these configurations are in the universal medium Univ. Each small colony has some work period T, and the alliance has a work period TU.

5.1.1. *Health*

Each small colony will, of course, have to obey a certain program. In order for the program to have the desired effect, the configuration of the colony must be somewhat standardized. The standardization need not be very elaborate: we will just mark the boundaries of the colony at the beginning of its work period. We will say in this case that the colony is *healthy*. If health is lost then it seems that we cannot keep our view elevated from the alliance to the colony simulated by it, since the small colonies no longer work by their original program.

5.1.2. *Legality*

Even if the members of the alliance are healthy they must share some information that is of no use for the decoder but is needed for the orchestration of their work. An idea that reduces the structure maintenance problem to a manageable size is that all the

information needed for this purpose falls into one of the following categories.

- *Parameters* of the alliance: some constants that are the same for each small colony;
- Two remainders mod Q describing the position of each small colony in the alliance;
- A remainder mod U describing the current clock value in the work period of the alliance.

The last two pieces of information are called *phase variables*. The activity of the colony can be made very predictable, provided that the value of the phase variables is correct, even if the rest of the information it handles is in doubt. In particular, the colony can consult the phase variables every time it is about to write to certain places—to make sure it writes only when the program permits this. This reliance on the phase variables will guarantee the continuity property mentioned in Section 4.1. We will call a healthy small colony *legal* when its parameters and phase variables have the required values.

In some sense, the values of the parameters and phase variables are not information. Indeed, they are a simple periodic function of space and time. Legality can therefore be restored by methods more local than those needed for the rest of the information. This observation solves the problem completely for two dimensions. Healthy colonies will maintain or restore their legality by Toom's Rule.

As a result, the two-dimensional reliable medium built in the present paper has the following property:

> Legality of colonies of the lowest order will be restored over an arbitrarily large damaged area if only the frequency of faults remains small in each of their alliances. This will happen even if none of the higher order colonies is healthy.

There is no Toom Rule in one dimension. Structure maintenance is still possible by more complicated methods, as shown in [G]. That they are more complicated can be seen from the fact that the above property will no longer hold. Legality and health can be defined in one dimension similarly to the way they were defined above. If the legality of colonies of the lowest order is damaged in an area needed for simulating a colony of order k then, even without noise, it will

be restored only if outside this area, still healthy colonies of order k are simulated. Thus, in two dimensions, to restore the legality to lowest order colonies in a damaged area, it was enough if it was surrounded with an area of comparable *size* containing legal colonies. In one dimension, not only the size of the surrounding area matters: it must be of comparable size and *organized to the maximum level*, even if only the restoration of the legality of the smallest colonies is required.

5.2. Health Maintenance

Toom's Rule restores legality to healthy small colonies. It is not clear yet how will health be restored to damaged small colonies, in order for them to be able to apply Toom's Rule. However, if we assume that health is restored by "magic" then we can *program* the same kind of magic into the simulation of big colonies. Indeed, at the beginning of the simulation of a big colony by an alliance, the latter will cast the big colony into the standard initial form required by health.

Because of the initializing step, the simulation of colonies by alliances is not like ordinary simulation: periodically, a certain structure is *forced* on the simulated colonies, whatever their previous state was.

Essentially, this is the method also used in [G], but there, setting all parameters had to be part of the initialization. This type of organization resembles that of biological systems: each cell has the genetic code of the whole organization.

The health of colonies of the lowest order will be assured because these colonies will be single cells of the reliable medium M and, as such, they will be defined to have only healthy states.

At the end of the computation, whatever information is in the alliance must be cast in the form of a redundant code. Due to heightened requirements of efficiency, this problem is harder to solve here than it was in [G].

6. MINIMUM REDUNDANCY

All past and present results discussed in this chapter indicate that if the size of the computation is N then the size of the simulating computation must be at least $N \log N$, *provided that some real computation is being performed.* The last qualification is necessary

since information storage needs only a constant factor in redundancy. In the model of cellular arrays, it is possible to distinguish between the time and space requirements of the computation, and it is therefore possible to represent the redundancy as a product of the redundancies in space and time. Thus if t steps of computation of n cells of a deterministic medium are simulated reliably by t' steps of n' cells of a stochastic medium then t'/t is the time redundancy and n'/n is the space redundancy.

The results available to date on cellular arrays indicate that the product of the time and space redundancies must be logarithmic in the size of the computation. We can state this as a conjecture, but it seems a difficult one to prove, especially that the case of "no real computation" must be excepted.

The logarithmic redundancy can be shifted entirely to space, or almost entirely to time, as the following examples show.

- The three-dimensional simulation in [GR] is real-time: there is no time redundancy, but there is space redundancy of size $\log^{2+\varepsilon} N$ (reduced to $\log^2 N$ in [BS]). The ideas of [GR] applied to [G] give a real-time two-dimensional reliable simulation with space redundancy $\log^2 N$.
- The two-dimensional simulation described in the present chapter has constant space redundancy, and time redundancy $\log^{2+\varepsilon} N$.

7. THE CONCEPTS AND THE RESULT

7.1. Sites and Events

Let $\mathbf{Z}_m$ be the group of remainders mod m if m is finite, and the set $\mathbf{Z}$ of integers if m is infinite. The set $\mathbf{W}$ of *sites* of our cells will be $\mathbf{Z}_m^2$. If $m = \infty$ then $\mathbf{W}$ is the two-dimensional rectangular lattice $\mathbf{Z}^2$. Otherwise, it is a lattice over a torus of diameter m. As a notational convenience, we define, in analogy with Pascal and real analysis, the intervals

$$[a..b) = \{x \in \mathbf{Z}: a \leqslant x < b\}.$$

The following definitions are given for two dimensions but they can be generalized to any number of dimensions. For a set G and a

vector v in $\mathbf{Z}_m^2$ and integer n we define

$$v + G = \{v + u: u \in G\}.$$

A partition of the plane and the three-dimensional space into squares and cubes will be used so often that we introduce a special notation for it. For a positive integer P, we write

$$[P; i] = [iP..(i+1)P)$$

for the intervals of length P shifted by a multiple of P from the origin. Similarly, we write

$$[P; i, j]^2 = [P; i] \times [P; j]$$

for squares of size P shifted from the origin horizontally and vertically by a multiple of P. The cubes $[P; h, i, j]^3$ are correspondingly defined. The intervals of type $[P; i]$, $[P; i, j]^2$ and $[P; h, i, j]^3$ will be called P-intervals, P-squares, and P-cubes, respectively.

7.2. Configurations and Evolutions

Let S be a finite set, the set of all possible *states* of our cells. Let B be a set of sites. A *configuration* x over B is a function that orders a state $x[p] \in S$ to each element p of B. Let x, y be two configurations, over the sets of sites B and C, respectively. We will say that y is a *translation* of x, if there is a vector u such that $C = u + B$, and for all p in B we have $y[p + u] = x[p]$.

Let x be a configuration over the set B of sites, and C a subset of B. Often, we will talk about the configuration $x[C]$ that we obtain by restricting x to C. However, if C' is obtained by translation from C and x' is obtained by the same translation from x then we consider $x'[C']$ equal to $x[C]$. One dimensional example: if $C = \{-1, 0, 1\}$ then $x[t, p + C]$ is the string

$$(x[t, p - 1],\ x[t, p],\ x[t, p + 1])$$

(indexed by $0, 1, 2$).

An *evolution* with set S of states over the set $\mathbf{W}$ of sites in the time interval I is a function x from $I \times \mathbf{W}$ to S. we will write $x[t, p]$ for the state of cell p at time t in the evolution x. The function $x[t, \cdot\,]$ is a configuration for each t. Therefore $x[t, C]$ is defined according to the notation above.

7.3. Media and Their Trajectories

We will use the convenient term *medium* for an array of cellular automata. The set of sites is not part of the definition of a medium since we will consider the behavior of the same medium over various sets of sites (in particular, for various values of m).

A *medium* Med will be defined by giving a finite subset G of $\mathbf{W}$ (the *interaction neighborhood* of the site 0), a finite set $S = S_{\mathsf{Med}}$ of states, and a *transition function* Med: $S^G \to S$. Thus, the transition function Med orders a value Med(y) in S to all configurations y over G. A medium with a particular set of sites (always a torus in this paper) will be called an *iterative array*. We will often refer to the number

$$|\mathsf{Med}| = \lceil \log_2 |S_{\mathsf{Med}}| \rceil$$

as the *cell capacity* of medium Med.

Some evolutions of an iterative array are called trajectories. These are the evolutions x for which the value $x[t + 1, p]$ depends only on the values $x[t, p + p']$ for p' in the neighborhood G, in a way determined by the transition function. An evolution x is called a *trajectory* of the medium Med if for all (t, p) we have

$$x[t + 1, p] = \mathsf{Med}(x[t, p + G]). \tag{7.1}$$

The simplest example of an interaction neighborhood is $G = \{-1, 0, 1\}$ in one dimension. There, the function Med(y) can be simply viewed as an operation of three arguments in S, i.e., the right-hand side of (7.1) can be simplified to Med$(x[t, p - 1], x[t, p], x[p + 1])$.

In our context, the word "error" could denote two different concepts. To distinguish these we call them "faults" and "deviations," and reserve the word "error" to informal discussion. We say that a *fault* occurred in x at $(t + 1, p)$ (with respect to Med) if (7.1) does not hold. If y is a trajectory then the event $x[v] \neq y[v]$ is a *deviation* (of the evolution x from y). Intuitively, a medium is reliable if it can keep the number of deviations small despite the occasional occurrence of faults.

EXAMPLE 7.1 [Toom's two-dimensional error-correcting medium]. This medium can be defined for any state-space. Let us define first the majority function $\mathrm{Maj}(x,y,z)$. If two of the three arguments coincide then their common value is the value of Maj, otherwise $\mathrm{Maj}(x,y,z) = x$. The interaction neighborhood is defined by $H = \{(0,1), (-1,-1), (1,-1)\}$, and the transition function R by $R(x[H]) = \mathrm{Maj}(x[H])$. Rule R computes the majority among the states of the northern, southwestern, and southeastern neighbors. Any constant function is a trajectory of R. (There are also some nonconstant periodic trajectories.) Suppose that $x[0,p] = 0$ for all p. We have a fault in $x[t,p]$ if it is not the value obtained by voting from the triple $x[t-1,\ p+H]$. We have a deviation if $x[t,p] \neq 0$. □

Let $(\xi[t,p]\colon t \in [0\,..\,l), p \in \mathbf{W})$ be a system of random variables with a joint distribution. We say that ξ is a *ϱ-perturbation* of medium Med if for each subset B of $[0\,..\,l) \times \mathbf{W}$ the probability that for *all* $v \in B$ a fault occurs in ξ at v is less than $\varrho^{|B|}$. (This condition is satisfied if the faults occur independently with probability ϱ, but it includes some other cases important in statistical physics.) We will say that ξ is a ϱ-perturbation of a trajectory y over the same space–time set if it is a ϱ-perturbation of Med with $\xi[0,p] = y[0,p]$ for all p in $\mathbf{W}$. Our goal is to find situations in which if the probability ϱ is small then the probability of deviations is also small: in other words, faults do not accumulate.

7.4. Coding and Simulation

By reliability we mean the possibility of simulating the computation of a deterministic medium by an error-prone medium. For this, of course, the simulation must use an error-correcting, redundant code.

In this chapter, we will want to encode the information content of a square array of cells into a larger square array. Let P be a positive integer, and let C be a P-square. The set S^C is the set of all configurations over C.

Let S_0 and S_1 be two alphabets, and P_0, P_1 two integers, with C_i being the corresponding squares. A (two-dimensional block-) *code*

$$\psi = (\psi_*, \psi^*)$$

from S_1 to S_0 with *blocksizes* P_0, P_1 is characterized by *source blocksize* P_1, and *target blocksize* P_0, *encoding function*

$$\psi_*\colon S_1^{C_1} \to S_0^{C_0}$$

and *decoding* function ψ^* where $\psi^*(\psi_*(x)) = x$. A code can be *extended* to configurations larger than those over a single square: over a union of aligned squares, by encoding resp. decoding each square separately. We have a *single-letter* code if $P_1 = 1$.

Let us be given two media Med_0 and Med_1, with state sets S_0, S_1, and a code ψ with blocksizes P_0, P_1, further the natural numbers T_0, T_1. We say that ψ is a *simulation* of Med_1 by Med_0 with parameters P_0, P_1, T_0, T_1 if for any multiples m_i of P_i, any trajectories x_i of Med_i over the spaces $\mathbf{W}_i = \mathbf{Z}_{m_i}$, the relation

$$x_0[0, \mathbf{W}_0] = \psi_*(x_1[0, \mathbf{W}_1])$$

implies

$$x_0[T_0, \mathbf{W}_0] = \psi_*(x_1[T_1, \mathbf{W}_1]).$$

The parameters P_0, P_1 are still called the blocksizes. The parameter T_1 is called the *source work period*, while T_0 is called the *target work period*. Thus, a simulation sets up a correspondence between the evolutions of two media. A simulation is *single-step* if $T_1 = 1$.

A medium is *universal* if it can simulate any other medium by a total single-letter single-step simulation. There are many possible universal media in two dimensions. Some of them (the ones built like universal Turning machines) are easy to program. Others (like Conway's Game of Life, or Margolus's Billiard Ball rule) are very simple to define. For the purpose of the following theorems, let Med_0 be a fixed medium.

7.5. The Result

Our goal is to find a "reliable" simulation of our fixed arbitrary medium Med_0 by a suitable medium M, as Med_0 computes on a certain set $\mathbf{Z}_n^2$ of sites for a certain number t of steps. The simulation γ used will depend on the sizes n, t of the computation and the error probability ε permitted in the result.

The medium M and the structure of the code γ is complicated to describe, and we leave the description to the proof. However, there are two reasons why it is not possible to "cheat" and hide the whole computation in the code.

- The code is independent of the computation (except for its size), and decoding is an inverse of the encoding. To begin a computation on the output, we do not have to decode and encode again.
- The code can be computed rapidly.

Let m denote the target blocksize of the code γ. We say that the trajectory y of M on $\mathbf{Z}_m^2$ is in the *range* of γ if we have $y[0, \mathbf{Z}_m^2] = \gamma_*(u)$ for some configuration u of Med_0 on $\mathbf{Z}_n^2$.

THEOREM 7.1. (MAIN THEOREM). *There is a medium M, a fault probability bound $\varrho > 0$ and for each n, t, ε with $L = \log(n^2 t/\varepsilon)$, a simulation γ with parameters n, m, T, T', such that the following holds.*

- *for any $h < [t/T']$, the probability of the event*

$$\gamma^*(\xi[hT, \mathbf{Z}_m^2]) = \gamma^*(y[hT, \mathbf{Z}_m^2])$$

  is at least $1 - \varepsilon$ for all trajectories y of M in the range of γ, and all ϱ-perturbations ξ of y.
- *The periods are (almost) logarithmically small: we have $P, T = O(2^{(\log L)^{1.6}})$. The redundancies can be estimated by $m = O(n + P)$, $T/T' < L^{2+o(1)}$.*
- *The code γ is computable in $O(T)$ steps on a suitable deterministic medium.*

The code given in the theorem implements every computation of the ideal fault-free medium Med_0 in the "physical" medium M in such a way that the probability of deviations remains under control. The space requirement n of the original computation is increased to m in the implementation. Hence according to the statement of the theorem, the space redundancy is a constant factor, except when the space n is too small to accommodate even one (logarithmic-size) block of the simulation.

8. ERROR-CORRECTING CODES

8.1. Burst Error Correction

It is not surprising that the theory of reliable computation makes use of the theory of error-correcting codes. (What is surprising is how much more is needed besides error-correcting codes.) Indeed, if information is not stored in encoded form then one step of computation will be enough to cause irreparable damage.

A one-dimensional binary code γ (i.e., a code from strings to strings) is said to *correct t errors* if for all strings u, v, if $\gamma_*(u)$ differs from v in at most t places then $\gamma^*(v) = u$.

EXAMPLE 8.1 [Repetition code]. Let $\gamma_*(u) = uuuuu$ over strings u of length k. Let the decoding function be defined over strings v of length $5k$. The decoding goes as follows: to determine the $(i + 1)$th symbol of $\gamma^*(v)$, we look at the symbols with indexes $5i, 5i + k, \ldots, 5i + 4k$ in v and take their majority. It is easy to see that this code γ corrects two errors. Repetition coding uses too much redundancy. There are better error-correcting codes. □

The kind of error correction we need is measured rather in the number of *error bursts* of a certain size corrected than in the number of errors corrected. We say that code γ, mapping binary strings of length Kn to binary strings of length Nn, *corrects t bursts of size n* if it corrects any pattern of errors covered by at most t intervals of the form $[n; i]$.

The present section sketches the proof of the following theorem.

THEOREM 8.1. *Suppose that n has the form $2 \cdot 3^r$. Then for all integers $N \leqslant 2^n$, $t \leqslant 2^{n-1}$ there is a code from strings of length $(N - 2t)\,n$ to strings of length Nn correcting t error bursts of size n. There is a universal one-dimensional cellular array performing the decoding and encoding for these codes, for all n, N, t, in space $O(Nn)$ and time $O(Nn \log Nn)$.*

This theorem is a straightforward application of the theory of algebraic codes. The details of the proof are not original and are not needed for the rest of the chapter. We advise the average reader to just use the theorem and skip the rest of the present section.

8.2. Shortened BCH Codes

The BCH codes are treated, e.g., in the textbook [Bl]. A (shortened) BCH code correcting t error bursts of size n has a space of symbols that is a Galois field GF(2^n), and needs $2t$ symbols devoted to redundancy. We represent the field as the set of remainders with respect to an irreducible polynomial. To make things completely explicit, we use the fact, derivable from the theory of fields (see [La]), that the polynomial

$$y^{2\cdot 3^r} + y^{3^r} + 1$$

is irreducible over GF(2). We choose n to be an integer of the form $2 \cdot 3^r$, and use the above irreducible polynomial to represent the field GF(2^n).

Let α be the element of GF(2^n) represented by the polynomial y. Then the elements $1, \alpha, \alpha^2, \ldots, \alpha^{N-1}$ are all different. Our codewords are the vectors $c = (c_0, \ldots, c_{N-1})$ over GF(2^n) with the property that

$$\sum_j c_j \alpha^{ij} = 0 \ (i = 1, 2, \ldots, 2t).$$

or in other words, the polynomials $c(x)$ of degree $N-1$ over GF(2^n) with the property that $c(\alpha^i) = 0$ for $i = 1, \ldots, 2t$. Let

$$g(x) = (x - \alpha)(x - \alpha^2)\cdots(x - \alpha^{2t}).$$

Then the *codewords* are the polynomials $c(x)$ of degree $< N$ over GF(2^n) divisible by $g(x)$. Hence the *information strings* can be represented by polynomials of degree $N - 2t - 1$ over GF(2^n), and encoding is multiplication by $g(x)$.

We need efficient decoding and encoding algorithms to avoid significant time delay. Moreover, the total space used by our algorithms can be at most constant times more than the amount of information processed. To achieve this, we adapted some well-known algebraic algorithms to our iterative array.

8.3. Algebraic Operations

All needed algebraic operations can be performed in $O(n \log n)$ time and in $O(n)$ space. It is likely that time can be brought down to $O(n)$ but we do not need this fact.

Addition can obviously be done in $O(n)$ time and space.

Fast multiplication needs the fast Fourier transform (over an appropriate smaller field). The latter can be done by an algorithm $F(n)$ in $f(n) = O(n)$ steps, within a one-dimensional cellular array of length $O(n)$, as follows. In stage 1 of $F(n)$, we separate the even and odd digits into the two halves of the array: this can be done in $O(n)$ steps. Now a recursive call of $F(n/2)$ simultaneously transforms both halves in $f(n/2)$ steps. (For the sake of the Fourier transform, we can pad n to a power of 2.) Finally, the even and odd bits are restored from the two halves using $O(n)$ steps. This shows $f(n) = O(n)$. The overhead (e.g., a firing-squad-type organization for timing) does not take up more than $O(n)$ space.

The $O(n)$ Fourier transform gives $O(n)$ polynomial multiplication (convolution), and division, which give $O(n)$ multiplication over GF(n). For division over the field, the Euclidean algorithm is needed. It is known that the Euclidean algorithm can be organized in $O(n \log n)$ serial operations (see [Bo]). The same algorithm can be implemented here, even with the $O(n)$ space requirement, in $O(n \log n)$ steps.

8.4. The Complexity of Encoding and Decoding

Computing the polynomial $g(x)$ can be done in $O(N)$ field operations and linear space, using fast Fourier transform over GF(2^n). Encoding is multiplication by $g(x)$, hence it can be done with fast Fourier transform at the same time and space cost.

The first step of decoding is the computation of the syndromes $S_i = c(\alpha^i)$. Computing the values of the polynomial c at $2t$ places is a well-known operation doable on a sequential machine in $O(N \log N)$ operations. It does not cause any problem to adapt the known algorithm to the requirement of linear space on our cellular array.

The next step of decoding uses the Euclidean algorithm over GF(2^n), for finding the error-locator polynomial and error-evaluator polynomial: see Chapter 7.7 of [Bl]. Therefore it can be done in $O(nN)$ space and $O(N \log N)$ field operations, hence in $O(nN \log nN)$ operations altogether.

This completes the outline of the proof of Theorem 8.1. □

9. OUTLINE OF AN ERROR-CORRECTING STRATEGY

9.1. Correcting a Sparse Set of Faults

The assumption that faults arise randomly and independently is a natural one but not easy to deal with technically. In practice, if the fault probability is small then usually the different but similar assumption is made that faults just arise rarely. This could mean, e.g., that in not too large domains of time and space, no two faults will ever occur. Under such assumptions, error correction becomes easier. One can hope to find a mechanism to correct one fault, provided no new faults occur during the correction process. It turns out that even this problem needs an elaborate solution, outlined in Sections 10 and 12.

A little more generally, we introduce two integer parameters U and r. We will say that the set of faults is (U, r)-sparse if in each U-cube in space–time, there are at most r faults. If r is small enough with respect to U then media will be constructed that resists a (U, r)-sparse set of faults.

The present section outlines how these contributions can help in fighting probabilistic faults.

9.2. Probabilistic Noise Bounds

Let us be given some probability distribution on all possible sets in the space–time $V = \mathbf{Z} \times \mathbf{Z}_{mU}^2$. This distribution gives rise to a random set $\mathscr{E}$. For a parameter p, we say that the distribution of $\mathscr{E}$ is *p-bounded* if for all k and all finite sets $A = \{v_1, \ldots, v_k\}$ of space–time points, we have

$$\text{Prob}(A \subset \mathscr{E}) \leqslant p^k.$$

Let us define a new space–time $V^* = \mathbf{Z} \times \mathbf{Z}_m^2$ whose points are the U-cubes of V. Point $v = (h, i, j)$ of V^* corresponds to the cube $v_* = [U; h, i, j]^3$ of V. If $C = [U; h, i, j]^3$ then we will write $C^* = (h, i, j)$. For each subset E of V, we define the set $E^*(r)$ in V^* as follows: v is in $E^*(r)$ if there are more than r elements of E in the U-cube v_*. Via this mapping, the random noise $\mathscr{E}$ gives rise to the random noise $\mathscr{E}^*(r)$ in the space V^*.

LEMMA 9.1 (NOISE BOUND). *Suppose*

$$p < U^{-3(r+1)} \tag{9.1}$$

and that the noise $\mathscr{E}$ is p-bounded. Then the noise $\mathscr{E}^*(r)$ is p^r-bounded.

Thus, the derived noise is bounded with a much smaller probability than the original one.

Proof. By the definition of p-boundedness, for any set D of k space–time points, the probability of $D \subset \mathscr{E}$ is at most p^k. We can therefore increase the probabilities of all such events by assuming that individual points belong to the noise independently with probability p. Let us make this assumption. The following statement follows then immediately.

LEMMA 9.2. *For any sequence $B_0, B_1, \ldots$ of disjoint U-cubes, the events $B_i^* \in \mathscr{E}^*$ are independent.*

It follows from this lemma that it is enough to prove for a single U-cube B that the probability of $B^* \in \mathscr{E}^*$ is less than p^r.

For any sequence of $r + 1$ points in B the probability that they are all in $\mathscr{E}$ is p^{r+1}. The total number of such sequences is less than $U^{3(r+1)}$. Therefore the probability that $B^* \in \mathscr{E}^*$, i.e., that there is such a sequence in B is less than $U^{3(r+1)}pp^r$. Hence, inequality (9.1) implies the statement of the lemma. □

9.3. Increasing Reliability by Simulation

Suppose that we could solve the problem of reliable computation in the presence of a (U, r)-sparse set of faults. This could mean, in the most simple-minded sketch, that for all U, r and media Med_0 satisfying certain simple conditions, we have a simulation φ and a new medium M_1 such that the decoding φ^* maps configurations x of M_1 over U-squares $[U; i, j]^2$ in the space $\mathbf{W} = \mathbf{Z}_{mU}^2$ into states $x^*[i, j]$ of Med_0 in the space $\mathbf{W}^* = \mathbf{Z}_m^2$. (We suppressed the dependence on U, r in this notation which is local to the present subsection.) The success in eliminating a (U, r)-sparse set of faults from the evolution x would mean that the decoded evolution $x^*[h, i, j]$ is a trajectory. In Sections 10–12, we will indeed solve the problem of reliable computation in the presence of a (U, r)-sparse set of faults, though the results will have a somewhat more complicated form.

If the set E of faults in x is not (U,r)-sparse then there will probably be faults in the evolution of x^* at the points of the derived set $E^*(r)$. *If the faults were confined to the set* $E^*(r)$ then the problem of error correction would be essentially solved. Indeed, according to the above lemma, the random set $\mathscr{E}^*(r)$ is p^r-bounded. We recreated therefore the original situation of a medium Med_0 to be implemented, and probabilistic faults, with the only difference that the probability bound is now p^r instead of p. We could call the simulation φ a "reliability amplifier," or in short, an *amplifier*. Concatenating several amplifiers, we can get probability bounds p^{r_1}, $p^{r_1 r_2}$, etc. Soon the probability of faults becomes negligibly small.

9.3.1. *Too Big Cells*

Before being carried away with this plan let us take a closer look. Each time we apply an additional amplification the cell capacity (we defined it as the logarithm of the number of states) of the simulating medium could increase. The number of amplifiers depends on the need to decrease the fault probability, and this depends on the size of the original computation. But this means that the cell capacity of the simulating medium depends on the size of the computation to be carried out, which is not what we want.

Due to this problem, Sections 13 and 14 modify the construction of Section 12, replacing each cell of the simulating medium by a block of some universal medium Univ. These blocks will be called *colonies*, and an array of colonies will be called an *alliance*. In the simulation thus modified, alliances in Univ will simulate blocks in the simulated medium Med_0. The working time T of colonies will be different from their size P. In case the simulated medium is also Univ the simulated blocks will also be called colonies. The simulated colonies are called *big*, the simulating colonies are called *small*.

9.3.2. *How to Restore Colony Structure*

Once we introduced colonies, a new problem arises, as mentioned in Section 5.2: the simulating colonies will need some minimal *structure* to work according to their program: how will this structure be restored after the faults? We will postulate this restoration *axiomatically*. Evolutions satisfying these axioms will be called *self-correcting evolutions*.

9.3.3. *How to Obtain Self-Correcting Evolutions*

We are willing to restrict ourselves to self-correcting evolutions if, in case the simulated medium is Univ again, the following holds:

> All evolutions obtained by decoding from evolutions that are self-correcting with respect to small colonies will be again self-correcting, with respect to big colonies.

The simulation, φ, will be modified in Section 16 to have this property. This will complete the construction of amplifiers.

10. A PERIODICALLY VARYING MEDIUM RESISTING SPARSE NOISE

10.1. The Noise Condition

In the present section, we start the investigation of a special kind of error correction. The concepts developed here have some interest in their own right. However, they are justified in the present chapter as a building block of the proof of the main theorem.

The noise condition used in the present section depends on a *time period* U and a *fault bound* r. Let us say that an evolution x is a (U,r)-*trajectory* if in every U-*cube*, x has at most r faults. We say that a (U,r)-*trajectory* is a (U,r)-perturbation of a trajectory y if it coincides with y at time 0. Our goal here is, for an arbitrary deterministic medium Med_0, to find a simulation φ of Med_0 by a suitable medium M_0 that withstands (U,r)-perturbation. The simulation has source- and target-blocksizes Q' and Q and source- and target-workperiods U' and U. It is simulating the work of Med_0 on $\mathbf{Z}^2_{nQ'}$.

The medium M_0 and the simulation φ used will not depend on the size n. They *do* depend on the medium Med_0 and the constants Q, U, Q', U', r. Thus, the error correction is block-for-block, using the fact that there are only r faults by U-cube.

The target blocks of our simulation will be called *alliances*. Each alliance has the form $[Q; i, j]^2$, consisting thus of $Q \times Q$ cells that, at least under fault-free conditions, form a cooperating unit.

10.2. Periodically Varying Media

Let us temporarily relax the requirement of homogeneity for the medium M_0. We permit the transition rule M_0 to be *inhomogeneous*:

it can change periodically in space (in both directions) with period Q and in time with period U. In other words, M_0 at space–time point t, i, j will depend on $t \bmod U$, $i \bmod Q$, and $j \bmod Q$: in the equation of a trajectory instead of 7.1 at a point (t, u) with $u = (i, j)$ we have

$$x[t+1, u] = M_0(x[t, u+G],\ t \bmod U,\ i \bmod Q,\ j \bmod Q).$$

We imagine such a transition rule as a computer program telling each cell of the alliances in each step of the working period specifically, what local action to perform.

Let $m = nQ$, denote the output blocklength of the code φ, and let $\mathbf{W} = \mathbf{Z}_m^2$.

Before stating the theorem let us note that the parameters Q, U, Q', U' are not completely arbitrary. The size Q must be large enough for $18r$ errors to be correctable. We require $Q > 3Q'$ to represent easily nine neighbor Q'-squares within one alliance. This requirement could be eased considerably since we can choose a larger cell capacity for the medium M_0. The time period U must clearly be long enough to carry out the error-correcting simulation. This will involve some decoding, coding, and repetition. A constant factor could again be hidden here by choosing a larger cell capacity in the simulating medium. For convenience, we also require the time period to be divisible by the blocklength. This leads to the following assumption.

CONDITION 10.1 (SIZE).

$$Q \geqslant \max(3Q', Q' + 22r),$$

$$U \geqslant \lceil U'/Q' \rceil Q^2 \log Q,$$

$$Q \mid U.$$

There are many ways to satisfy these conditions. Given an arbitrary r and Q', U', we can choose Q greater than the right-hand side of the first inequality. Then we can choose U to be any multiple of Q greater than the right-hand side of the second inequality.

THEOREM 10.1. *Suppose that the Size Condition 10.1 holds. Then there is a periodically varying medium M_0 with periods U, Q and capacity $|M_0| = O(\log(Q+U))$, and a simulation φ with parameters*

Q, Q', U, U' such that the following holds. For all n, all trajectories y of M_0 in the range of φ over $\mathbf{Z}_{nQ}^2$, all (U, r)-perturbations x of y, and all nonnegative integers h we have

$$\varphi^*(x[hU, \mathbf{W}]) = \varphi^*(y(hU, \mathbf{W}]).$$

The rest of the present section is devoted to the proof this theorem.

10.3. The Construction

Error-correcting devices have a crucial feature: *continuity*. The property of continuity means that every elementary event of the computation (happening at one space–time point) influences only a small part of the result. The continuity property will be achieved as follows. We subdivide the alliance into a certain number of columns and the working period into the same number of stages. We establish that in stage i, only column i can change its information content. In this way, stage i influences only column i.

10.3.1. Variables

As we often do in the analysis of Turing machines, the state-set $S = S_{M_0}$ of the medium M_0 will be the Cartesian product of several sets: $S = S_1 \times \cdots \times S_k$. Thus, the value $x[h, i, j]$ is the collection of several values $Z_1[h, i, j], \ldots, Z_k[h, i, j]$. For a Turing machine, we would say that we divided the tape into k individual "tracks." We call $\log|S_i|$ the *width* of track i.

Borrowing the terminology of computer programming, we will refer to the value

$$Z_1[h, i, j] = Z_1(x)[h, i, j]$$

as the *value of variable* Z_1 at site (i, j) at time t (in the evolution x). The transition rule M_0 will therefore say how the individual variables at site (i, j) depend on those in the neighbor sites at time h.

The notion of tracks (variables) is another tool to achieve continuity. We agree that the information used by the decoding function φ^* is in the variable *InpMem*, i.e., the value $\varphi^*(x[t, A])$ for the alliance A depends only on $InpMem[t, A]$. Now the program can

move information across any cells (i, j): as long as it does not require to change the variable $InpMem[i, j]$ and no fault happens at (i, j), the variable will not be changed.

Actually, the continuity requirement applies only to the variables in *InpMem* and the ones used immediately for their updating, called *OutMem*. We extend the notion of *deviation* to tracks, i.e., variables. When we compare the evolution x with the trajectory y of the same medium, we will say that there is a deviation at space–time point (t, i, j) on track Z_1 if $Z_1(x)[t, i, j]$ differs from $Z_1(y)[t, i, j]$. Of course, if x deviates from y on any one track at the space–time point (t, i, j) then there is a deviation in the absolute sense at this point, i.e., $x[t, i, j]$ and $y[t, i, j]$ are different. But in general, we will be more interested in the deviations on particular tracks than in deviations at all.

Let us outline the major operations that the inhomogeneous medium M_0 performs in the U steps of the work period in the alliance and the neighbor alliances.

10.3.2. *Decoding*

Let b be the first integer of form $2 \cdot 3^k$ greater than both the cell capacity $|\mathsf{Med}_0|$ and $\log Q$. The Size Condition and Theorem 8.1 implies the existence of a code Algeb from binary strings of length bQ' to binary strings of length bQ that corrects $11r$ error bursts of length b. The simulated Q'-square of Med_0 will be encoded row-by-row by the code Algeb.

Now we can define the code φ. It takes a Q' by Q' configuration of Med_0. It encodes each row into a binary string of length bQ'. Then it applies the code Algeb to each row, and writes the result onto the *ImpMem* track of a row of length Q, writing b bits into one cell. (Thus, the *ImpMem* track must be at least b bits wide.) The other tracks are set to an arbitrary initial state. There will be $Q - Q'$ unused rows in the target square: we can ignore them. Decoding is the inverse. We take the *ImpMem* track, and apply the decoding function Algeb to each of its rows.

The first task of the program is *decoding*. Indeed, we do not know how to manipulate the information in encoded form.

The decoding process, as well as all other operations mentioned later, uses tracks different from *InpMem* or *OutMem*. We will not give names to all these other tracks: they have the collective name Workspace. The result of the decoding in each alliance is a

configuration of Med_0 over a $Q' \times Q'$ square. The original content of the *ImpMem* variables is not changed. The result of the decoding is stored on a Workspace track.

10.3.3. *Input*

Due to the Size Condition, we can store the result of decoding in a subsquare of size $Q/3$ of the alliance, in a part of the Workspace called the Simulator track. In this way, the Simulator track of an alliance could store the decoded information not only from the alliance itself but also from its eight neighbors, arranged in their original geometrical relation, in nine subsquares. This is necessary, since the state of the alliance after U steps will also depend on the neighbor alliances.

10.3.4. *Computation*

On the decoded contents of the original blocks of Med_0, the work of Med_0 is simulated step-for-step for Q' steps, in the Simulator track. This procedure *Compute*(t) is simple: the Workspace track of the cells of M_0 is programmed to behave like Med_0 for t steps.

10.3.5. *Encoding, Output, Repetition*

The square of size $Q/3$ on the Simulator track is encoded again.

Now, we resist the temptation to write back the encoded result into the *OutMem* track of the colony. While decoded, the information was vulnerable to even a single fault. The computation performed on it could spread the errors even farther. Therefore we cannot completely trust the result. We will use only a single column of it, column s, therefore this part of the program can be called the procedure *Output*(s). In this procedure, we write column s of the result back into column s of a new track called *OutMem*. The rest of the result can be discarded. The reason we have to use a new track is that the old value of *InpMem* is still needed. Only the last step of the program replaces *InpMem* with *OutMem*.

The part of the program defined until now can be summarized in the following procedure.

```
procedure CompColumn(s, t);
begin
  Decode;
  Input;
  Compute(t);
  Encode;
  Output(s);
end;
```

One more level of repetition is needed. The parameter t above will always be chosen smaller than Q'. Indeed, more steps of the computation would depend on alliances farther away than the immediate neighbors. Therefore the program part defined until now must be repeated U'/Q' times.

The program below also has some idle steps at the beginning. These are not important for other than technical convenience in the later proofs.

The whole program can now be written as follows. Let

$$N = \lfloor U'/Q' \rfloor.$$

```
idle 2Q steps;
for l := 1 to N + 1 do begin
  if l ⩽ N tben t := Q' else t := U' − NQ'.
  for s := 1 to Q do
    CompColumn(s, t);
  ImpMem := OutMem;
end
```

Since the rule M_0 is allowed to be space–time dependent, the implementation of the above program in the form of a transition table does not cause any principal difficulty. We do not do it because it is tedious. It is clear from Theorem 8.1 that all five parts of *CompColumn* take $O(Q \log Q)$ steps, hence the whole program takes at most $O(Q^2(U'/Q')\log Q)$ steps. A constant factor in the running time can be eliminated by the unusual speedup trick, increasing the cell capacity and combining several steps into one.

10.4. Proof of Theorem 10.1

Let y be a trajectory of M_0 over $\mathbf{W}$ in the range of φ. Let x be a (U, r)-perturbation of y. We have to prove the relation

$$\varphi^*(x[hU, \mathbf{W}]) = \varphi^*(y[hU, \mathbf{W}])$$

for all h. We will actually prove a little more:

LEMMA 10.1. *Let C be an alliance. Then for all h, in each row of C, on the InpMem track there are at most $10r$ deviations of x from y.*

The theorem will follow since the decoding φ^*, which is essentially the decoding Algeb*, corrects $18r$ errors. □

Proof of Lemma 10.1. The lemma certainly holds for $h = 0$, since x and y coincide there. We will assume that it holds for h and prove it for $h + 1$. Let C be an alliance. From the inductive assumption, it follows that at time hU, there are at most $10r$ deviations on the *ImpMem* track in any row of any neighbor alliance of C at time hU.

The events happening in the evolution x during the time interval $[U; h]$ that can have any effect on the configuration in C at time $(h + 1)U$ can obviously be covered by nine U-cubes. According to the assumption that x is a (U, r)-trajectory, in these nine cubes there are at most $9r$ faults.

The inner part of the program given above consists of Q calls to the procedure *CompColumn*(s, t), and a last step copying *OutMem* to *InpMem*. We are going to show that only those columns s of *InpMem* will have deviations on the *InpMem* track at time $(h + 1)U$ for which either there was a fault during the sth call or a fault in column s at some other time. The number of these columns is at most $9r + r = 10r$.

It is enough to show that in the calls of the procedure *CompColumn*(s, t) when no faults happen, no deviation is created in the sth column of the *OutMem* track. Certainly no deviations will be created during these calls on the *ImpMem* track, since nothing will be written there. Therefore columns containing deviations on the *InpMem* track in any of the neighbor alliances of C come from two sources. $10r$ of these columns were inherited from the beginning hU of the work period. r others are created by faults. Therefore

at the beginning of a call of *CompColumn*(s, t), there are at most $11r$ columns per alliance containing deviations on the *InpMem* track. Since the code Algeb can correct $11r$ errors per alliance, the effect of these deviations is eliminated by the procedure *Decode* during a fault-free call of *CompColumn*(s, t). □

11. TOOM'S RULE

11.1. Shrinking Deviations

Let us review the way a finite amount of information can be remembered in a two-dimensional homogeneous medium. For a finite set S of states, Toom's Rule R is defined in Section 7.3. Why is Toom's Rule R likely to preserve a nearly constant initial configuration? Let us define the linear functions

$$l_1(\alpha, \beta) = -2\alpha + \beta, \qquad l_2(\alpha, \beta) = 2\alpha + \beta, \qquad l_3(\alpha, \beta) = -\beta.$$

For an arbitrary subset F of $\mathbf{Z}^2$ we define

$$m_j(F) = \sup_{v \in F} l_j(v).$$

We call $m_j(F)$ the *measurements of* F. For any real numbers a_1, a_2, a_3, let us define the triangle

$$I = L(a_1, a_2, a_3) = \{(x, y): l_j(x, y) \leqslant a_j \qquad \text{for } j = 1, 2, 3\}.$$

The numbers a_j are the measurements $m_j(I)$.

The following assertion is easy to verify. Let c be some constant, and let y be the constant evolution, i.e., for which $y[t, u] = c$ for all times t and sites u.

LEMMA 11.1. *Suppose that x is a trajectory of R in which at time t, the set of deviations from the constant evolution y is enclosed into the triangle $L(a, b, c)$. Then at time $t + 1$, the same set is enclosed in $L(a - 1, b - 1, c - 1)$.*

It is this speed of shrinking, *independent of size* of the set of deviations that distinguishes Toom's Rule.

11.2. Triangles

11.2.1. Size and Separation

For later use in proofs concerning Toom's Rule, let us introduce some more geometrical concepts. We call

$$|I| = (a + b)/2 + c$$

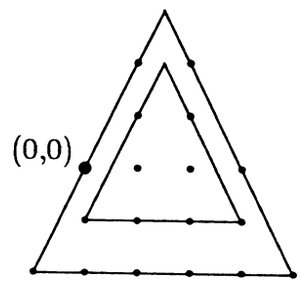
the size of triangle $L(a, b, c)$. [This expression must be chosen since the relation $(l_1 + l_2)/2 + l_3 = 0$ will then make sure that the size of a point is 0.] If the size is negative then the triangle is empty. For finite diameter m of the torus, the above definition does not work since inequality is not defined in $\mathbf{Z}_m$. But the definition can be easily extended as long as the size is less than m. Let us add this requirement to the definition of triangles.

For a set $\mathcal{J}$ of sets we denote by $\cup \mathcal{J}$ their union and by $|\mathcal{J}|$ the sum $\sum_{J \in \mathcal{J}} |J|$. We say that $\mathcal{J}$ *covers* a set if its union does so. We denote by $\# \mathcal{J}$ the number of elements of $\mathcal{J}$.

It is easy to verify the following: If the triangles I and J have nonempty intersection and $|I| + |J| < m$ (where m is the diameter of the torus) then the size of the smallest triangle containing their union is smaller than $|I| + |J|$. We can transform a finite set $\mathcal{J}$ of triangles with $|\mathcal{J}| < m$ into a set $\mathcal{J}'$ of disjoint triangles in the following way: we successively replace any pair of intersecting triangles in the set with the smallest triangle containing their union (this process will be called *merging*), as long as we find intersecting triangles. We have

$$|\mathcal{J}'| \leqslant |\mathcal{J}|.$$

Figure 1. Triangles $L(0, 6, 2)$ and $L(-1, 5, 1)$, of sizes 5 and 3, respectively.



THEOREM 11.1. *Assume $m \geqslant 16$. Let $y[t,u]$ be a trajectory of R^+ over $\mathbf{Z}_m^2$. Then G_y ($\langle$cons$\rangle m$) is either empty or is the whole torus.*

The estimate $\langle$cons$\rangle m$ is certainly too high, but the present chapter cannot attempt to find the best coefficient of m in this theorem.

11.3.1. *Geometrical Definitions*

For the present subsection, triangles will be defined with the help of the linear functions

$$l_1'(\alpha,\beta) = -\alpha, \qquad l_2'(\alpha,\beta) = -\beta, \qquad l_3'(\alpha,\beta) = \alpha+\beta.$$

These are the triangles naturally associated with rule R' as the old triangles were with rule R. Now R' shrinks a triangle $L'(a,b,c)$ to $L'(a,b,c-1)$. The *size* of a triangle $L'(a,b,c)$ is given by $a+b+c$. Thus, the rule decreases the size of each triangle by 1. We also introduce a new neighborhood relation on the two-dimensional lattice $\mathbf{Z}_m^2$. The new neighbors of a cell 0 are all vectors (i,j) with

$$\max(|i|,|j|,|i+j|) \leqslant 1.$$

(These are the old neighbors with the exception of the northeastern and southwestern ones.) We will view the set of sites as a graph where the edges connect the neighbors.

The mapping

$$(i,j) \to (i \bmod m, j \bmod m)$$

will be called the *wraparound*. For any cycle $u_1, u_2, \ldots, u_n$ in our graph we can compute the sum

$$(u_2-u_1)+(u_3-u_2)+\cdots+(u_1-u_n)$$

not taken mod m but as the sum of integer two-dimensional vectors in the plane (not the torus). This sum always has the form (mi, mj) for some i, j. When $i=j=0$ we say that the cycle is *contractable*. A connected subset of the torus is called contractable if all cycles in it are contractable. A noncontractable cycle is called a *belt*. The following lemma is elementary.

LEMMA 11.3. *A connected subset C of $\mathbf{Z}_m^2$ contains no belt if and only if there is a set C' in $\mathbf{Z}^2$ whose wraparound is C, such that the wraparound has an inverse on C that maps neighbors to neighbors. The set C' is unique up to translation.*

For a connected contractable set C we call the C' of the above lemma the *lift* of C. The *size* of a contractable connected component is the size of the smallest triangle containing its lift.

11.3.2. *Global Behaviour of Toom's Rule*

It is sometimes easier to think of the rule R' as applied not to a configuration x over $\mathbf{Z}_m^2$ but to the set G_x. Thus, for any transition rule D with a state-space $\{0,1\}$ and any set $E \subset \mathbf{Z}_m^2$, we say $D(E) = E'$ if D has a trajectory y such that $E = G_y(0)$, $E' = G_y(1)$.

LEMMA 11.4. *Let C be a set of sites.*

(*a*) *Suppose that $R'(C)$ is nonempty. Then C is connected if and only if $R'(C)$ is connected.*

(*b*) *The set $R'(C)$ contains a belt if and only if C does. Suppose that C is connected, contractable with size $n>0$. Then the size of $R'(C)$ is $n-1$.*

(*c*) *If $C = C_1 \cup \cdots \cup C_n$ is the breakup of C into disjoint connected components then $R'(C) = R'(C_1) \cup \cdots \cup R'(C_n)$ is the breakup of $R'(C)$ into disjoint connected components (some of whom may be empty).*

The verification of this lemma is not difficult but requires the somewhat tedious examination of a few special cases, so we do not give it here.

THEOREM 11.2. *Let $y[t,u]$ be a trajectory of R' over $\mathbf{Z}_m^2$. We have $G_y(t) = 0$ for a large enough t if and only if $G_y(0)$ contains no belt.*

Proof. It follows from Lemma 11.4 (*b*) that if there is a belt in $G_y(0)$ then $G_y(t)$ never becomes empty. Suppose there are no belts $G_y(0)$. Then it follows from (*c*) and (*b*) of Lemma 11.4 that its components disappear after a finite number of steps. □

11.3.3. Combining Inflation with Toom's Rule

The following lemma is easy to check.

LEMMA 11.5. *Let $C = C_1 \cup \cdots \cup C_n$ be the breakup of C into disjoint connected components. Then each of the connected components of Inflate(C) is the union of sets of the form Inflate(C_i).*

The following inequality for an arbitrary set of sites follows from the monotonicity of the rule R'.

$$Inflate(R'(C)) \subset R'(Inflate(C)). \tag{11.3}$$

The following lemma follows easily from Lemmas 11.4 and 11.5.

LEMMA 11.6. *(a) Let $C = C \cup \cdots \cup C_n$ be the breakup of C into disjoint connected components. Then each of the connected components of $R^+(C)$ is the union of sets of the form $R^+(C_i)$.*
(b) If C is connected and $R^+(C)$ is contractable with size n then C is contractable with size $n+1$.

Let y be a trajectory of R^+. A connected component C' of $G_y(t+1)$ is said to be an *immediate successor* of a component C of $G_y(t)$ if $R^+(C)$ is nonempty and is contained in C'. A component D of $G_y(t+k)$ is a successor of a component C of $G_y(t)$ if there is a sequence $C_0 = C, C_1, \ldots, C_k = D$ such that C_i is a component $G_y(t+i)$ and is an immediate successor of C_{i-1}. The above lemma says that each component has at most one successor at all times.

LEMMA 11.7. *Let y be a trajectory of the rule R^+, let $n < m$. Let C be a connected contractable component of $G_y(t)$ which has only a single predecessor C_i at time i for all i in $(t-2n\,..\,t)$. Then C_{t-n} contains a triangle of size n.*

Proof. Since C is nonempty the predecessor at time $t-1$ has size at least 2. It follows from Lemma 11.6 that if C_{t-2n+1} is contractable then its size is at least $2n$. Let us denote temporarily $t-2n+1 = s$. We have, using (11.3):

$$\begin{aligned} C_{t-n+1} &= (R^+)^n(C_s) = (Inflate \circ (R')^2)^n(C_s) \\ &\supset Inflate^n((R')^{2n}(C_s)). \end{aligned} \tag{11.4}$$

Since C_s either contains a belt or has a size at least $2n$, Lemma 11.4 implies that the set $(R')^{2n}(C_s)$ is not empty. Therefore the right-hand side of (11.4) contains a triangle of size n. □

Proof of Theorem 11.1. Let y be a trajectory of R^+. Let us assume that $G_y(\langle cons\rangle m)$ is not empty. Let $t_0 = \langle cons\rangle m$. For $i = 0, 1, \ldots$, let us define

$$\delta_i = \lceil \sqrt{48}(0.6)^{i/2} m \rceil, \qquad t_{i+1} = t_i - \delta_i.$$

First we show $t_i > 0$ for all $i \leqslant 4 \log m$. Indeed, we have

$$\begin{aligned} t_i &= t_0 - (\delta_0 + \cdots + \delta_{i-1}) \\ &\geqslant t_0 - i - \frac{\sqrt{48} m}{1 - \sqrt{0.6}} \\ &> \langle cons\rangle m - i - 31m \geqslant m - 4 \log m \geqslant 0. \end{aligned}$$

In the last two inequalities we used the definition of $\langle cons\rangle$ and $m \geqslant 16$.

If some $G_y(t_i)$ with $i > 0$, $t_i \geqslant 0$ contains a belt then $G_y(t_1)$ contains a belt. In the $\delta_0 \geqslant \sqrt{48} m$ steps until t_0 the component of this belt will be inflated over the whole space. Suppose therefore that $G_y(t_i)$ with $i > 0$, $t_i \geqslant 0$ has no belts.

Let n_i be the number of components in $G_y(t_i)$. Let us call a connected component in $G_y(t_i)$ *persistent* if it has exactly one predecessor in $G_y(t)$ for all t in $[t_{i+1}\,..\,t_i)$. Let j be the first i such that at least $n_i/6$ of the connected components in $G_y(t_i)$ are persistent.

First we show $j < 4 \log m$. For any $i < j$, at least $5n_i/6$ of the connected components are not persistent. Each of these components has at least two predecessors in $G_y(t_{i+1})$. We have therefore $n_{i+1} \geqslant 5n_i/3$ for all $i < j$. It follows that $n_i \geqslant (5/3)^i > 2^{i/2}$. For $i \geqslant 4 \log m$ this would give $n_i > m^2$, i.e., the number of components would exceed the size of the whole space.

Let C be a persistent connected component of $G_y(t_j)$. It follows from Lemma 11.7 that the predecessor of C in $G_y(t_j - \lceil \delta_j/2 \rceil)$ contains a triangle of size $\lceil \delta_j/2 \rceil$, i.e. it contains at least $\delta_i^2/8$ points. Altogether, the set $G_y(t_j - \lceil \delta_j/2 \rceil)$ contains thus at least as many points as

$$\frac{n_j}{6} \frac{\delta_j^2}{8} \geqslant \left(\frac{5}{3}\right)^j \frac{\delta_j^2}{48}.$$

Using the definition of δ_j, this is greater than number m^2 of points in the torus. □

12. A HOMOGENEOUS MEDIUM RESISTING SPARSE NOISE

12.1. Eliminating Space–Time Dependency

In this section, we construct a medium M_1 that does everything that M_0 does in Theorem 10.1, but it will be *homogeneous*, i.e., a medium in the original sense of our definitions.

For the sake of the present section, let us denote

$$\langle \text{sick} \rangle = 18$$

$$\langle \text{dev} \rangle = \langle \text{sick} \rangle + 18$$

$$\langle \text{corr} \rangle = 10\langle \text{sick} \rangle + \langle \text{dev} \rangle.$$

CONDITION 12.1 (SIZE).

$$Q \geqslant \max(3Q', Q' + 2\langle \text{corr} \rangle r),$$

$$U \geqslant ([U'/Q'])Q^2 \log Q,$$

$$Q \mid U.$$

There are many ways to satisfy these conditions, as shown in the remark after Condition 10.1.

THEOREM 12.1. *Suppose that the Size Condition 12.1 is satisfied. Then there is a medium M_1 and a simulation φ satisfying the assertions of Theorem 10.1.*

The present section is devoted to the proof of this theorem.

We can eliminate the inhomogeneity from M_0 *formally*, by introducing an extra restriction on the evolution. We add three new tracks, i.e., three new variables denoted by τ, π_1, π_2 called the *phase variables*. The variable τ takes its value from the set $[0..U)$. The variables π_1, π_2 take their values from $[0..Q)$. The new local state space is that of M_0, multiplied by the ranges of the phase variables.

We could restrict our attention to evolutions x over the new state-space with the property that at all space-time points (t, i, j) we have

$$\tau(x)[t, i, j] = t \bmod U,$$

$$\pi_1(x)[t, i, j] = i \bmod Q,$$

$$\pi_2(x)[t, i, j] = j \bmod Q. \tag{12.1}$$

This requirement means that even the faults cannot change the values of the phase variables. Now we can modify the transition rule of M_0 in such a way that instead of depending directly on time and space, it will depend on the phase variables. This new medium M_0' is homogeneous, and obviously satisfies Theorem 10.1, if we restrict ourselves to evolutions satisfying (12.1). In what follows we show how to eliminate the requirement (12.1).

12.1.1. *Toom's Rule for Periodic Stable States*

We will use Toom's Rule in a slightly generalized form. What we need to preserve are the periodic evolutions of the variables τ, π_1, π_2 rather than all-constant evolutions. (The generalized formulation of Toom's Rule is used in [Too] first.)

The rule for τ is (arithmetic operations are mod U):

$$\tau[h+1, i, j] = \text{Maj}(\tau[h, i, j+1], \tau[h, i-1, j-1], \tau[h, i+1, j-1]) + 1.$$

The rule for π_1 is (arithmetic operations are mod Q)

$$\pi_1[h+1, i, j] = \text{Maj}(\pi_1[h, i, j+1], \pi_1[h, i-1, j-1] + 1, \pi_1[h, i+1, j-1] - 1).$$

The rule for π_2 is analogous.

12.1.2. *The Medium M_1 and the Code φ*

Let us define the medium M_1 as follows. It works as the medium M_0' on the variables different from the phase variables. To the phase variables, it applies a generalized Toom's Rule as defined above.

The medium M_1 is homogeneous. We will prove that it satisfies Theorem 10.1, even if the property (12.1) is not required.

The code φ is defined similarly to its definition in the previous section. Decoding is exactly the same. In encoding, the phase variables must be set correctly. The phase variable τ will be set to 0. In this way, the result of the encoding is always an alliance at the beginning of its work period.

12.2. Legal Cells

Let x be an evolution. We will say that the cell at site (i, j) is *legal* at time t if the equations (12.1) are satisfied. Toom's Rule has the property that, in the absence of faults, it decreases the set of illegal cells. This property can be spelled out in a lemma similar to Lemma 11.1. In words, if the set of illegal cells is enclosed in a triangle then in the next moment, it will be enclosed in a smaller triangle. We generalize to sets of triangles. If there are several enclosing triangles then they will be able to contract onto the illegal cells only if each of them is surrounded by legal cells. For technical convenience, we will express this in the following form: there is a set of *disjoint* triangles whose *deflation* covers the illegal cells. Thus, we will use the following property of Toom's Rule.

LEMMA 12.1. *Let y be a trajectory of M_1 (not necessarily in the range of φ). Suppose that at time t all illegal cells are enclosed into $D(\mathscr{I}, 1)$, where $\mathscr{I}$ is a disjoint set of triangles. Then at time $t+1$, all illegal cells are enclosed in $D(\mathscr{I}, 2)$.*

Let $m = nQ$, let y be a trajectory of M_1 over $\mathbf{W} = \mathbf{Z}_m^2$ in the range of φ. Let x be a (U, r)-perturbation of y.

Let E be an alliance. The set of sites at time $t - U$ that can have any effect on the state of E at time t is $\Gamma(E, U)$. Since squares of the order of magnitude U occur this way, we will try to use U-squares as much as possible. Condition 12.1 required U to be divisible by Q. Therefore each U-square is the union of some alliances. Let us call the U-squares therefore *alliance clusters*.

We will say that an alliance cluster C is *locally healthy* at time t if there is a set $\mathscr{I}$ of disjoint triangles with $|\mathscr{I}| \leqslant \langle \text{sick} \rangle r$ such that the deflation $D(\mathscr{I}, 1)$ covers the set of illegal cells in $\Gamma(C, U)$.

LEMMA 12.2. *Assume that the conditions of Theorem 12.1 hold. Let C be an alliance cluster. Then at all times hU, the following statements hold:*

(*a*) *In each alliance E of C, in each row of E, on the InpMem track there are at most $\langle \text{dev} \rangle r$ deviations of x from y.*
(*b*) *C is locally healthy.*

Of course, this lemma implies the theorem. □

To prove the lemma, we will use induction on h. It holds by definition for $h = 0$. Let us assume that it holds for h, we prove it for $h + 1$.

12.3. Singularity

First we will prove statement (*b*) of Lemma 12.2. Let us call a space–time point (t, i, j) *singular* if either a fault occurs in the evolution x at this point or (i, j) is illegal at time t. Otherwise, the point is called *regular*. At a regular space–time point (t, i, j), the medium M_1 computes the value $x[t+1, i, j]$ just like the inhomogeneous medium M_0 did. Therefore our immediate goal is to obtain a bound on the set of singular points. The next lemma bounds their time projection by a small set of short intervals, and their space projection by a small set of triangles. First, some remarks and definitions.

Let C be an alliance cluster. The set of sites at the time $q \in [U; h]$ that can have any effect on the state of C at time $(h+1)U$ is

$$C_q = \Gamma[C, (h+1)U - q].$$

Let us define

$$\langle \text{kill} \rangle = (4.5\langle \text{sick} \rangle + 18)r + 1.$$

Notice that the first inequality in Size Condition 12.1 implies

$$10\langle \text{kill} \rangle < Q. \tag{12.2}$$

LEMMA 12.3 (SINGULARITY LOCALIZATION). *There is a set Y of times in $[U; h]$ consisting of $9r$ intervals of length $\langle \text{kill} \rangle$, and sets $\mathscr{K}$, $\mathscr{L}$ of triangles with*

$$|\mathscr{K}| \leqslant 9\langle \text{sick} \rangle r, \qquad |\mathscr{L}| \leqslant \langle \text{sick} \rangle r$$

such that the following holds. For any singular space–time points (q, i, j) *during* $[U; h]$ *with* $(i, j) \in C_q$ *we have*

$$q \in Y \cup [hU \,..\, hU + \langle \text{kill} \rangle), \qquad (i, j) \in \bigcup D((\mathscr{K} \cup \mathscr{L})', 1).$$

For $q > hU + \langle \text{kill} \rangle$, *the site* (i, j) *is covered even by the smaller set* $D(\mathscr{L}, 1)$.

This lemma implies statement (*b*) of Lemma 12.2. Indeed, inequality (12.2) implies $(h + 1)U > hU + \langle \text{kill} \rangle$. Therefore the set of singular sites, and thus certainly the set of illegal cells at time $(h + 1)U$ is covered by $D(\mathscr{L}, 1)$.

12.4. Triangle Development

12.4.1. *Noise*

The set C_{hU} consists of nine U-squares. The domain of space–time involved is covered by the nine cubes above these squares. Since x is a (U, r)-trajectory, there are at most $9r$ faults in these cubes. For a number q in $[U; h]$, let $\mathscr{F}_q$ be the set of projections of the faults in this domain happening at time q. The number q will be called *singular* if the set $\mathscr{F}_q$ is nonempty. Otherwise, it is called *regular*.

Let the set X consist of all singular numbers q. We define

$$Y = \bigcup_{q \in X} [q \,..\, q + \langle \text{kill} \rangle).$$

By definition, the set Y can be covered by $9r$ intervals of length $\langle \text{kill} \rangle$.

It is convenient to take the set $\mathscr{F}_q$ into account via the following set of triangles:

$$\mathscr{L}_q = D(\mathscr{F}_q, -1)'.$$

Here, each point of $\mathscr{F}_q$ was enclosed into a triangle of size 0 (itself), and then this triangle was blown up by 1. The blowup provides for the possibility of a later deflation by the same amount. The size of each element of $\mathscr{L}_q$ is 2. Therefore the sum of their sizes is at most $18r$.

12.4.2. *The Triangle Sets* $\mathscr{I}_q$

Let us define for all q in $[U; h]$ a set $\mathscr{I}_q$ of disjoint triangles such that the following proposition holds.

LEMMA 12.4. *At time* q, *the system* $D(\mathscr{I}_q, 1)$ *covers the illegal cells in* C_q.

The set $\mathscr{I}_q$ will be defined inductively. Since we assumed that C_{hU} is locally healthy at time hU, for each of the nine alliance clusters in C_{hU} there is a set $\mathscr{K}_i$ of triangles of size less than $\langle \text{sick} \rangle r$ for which $D(\mathscr{K}_i, 1)$ covers the sick cells at time hU. We define

$$\mathscr{I}_{hU} = \mathscr{K} = \left(\bigcup_i \mathscr{K}_i \right)'.$$

We proceed to the definition of $\mathscr{I}_q$ for $q > hU$. Let us assume that $\mathscr{I}_q$ is defined

$$\mathscr{I}_{q+1} = \begin{cases} (\mathscr{I}_q \cup \mathscr{L}_q)' & \text{if } q \text{ is singular,} \\ D(\mathscr{I}_q, 1) & \text{otherwise.} \end{cases}$$

With this definition, the proof of Lemma 12.4 is easy by repeated use of Lemma 12.1. □

12.4.3. *Vanishing Triangles*

The next lemma states that by the time $q = \langle \text{kill} \rangle$ the deflations eliminate the effect of the set $\mathscr{I}_{hU}$.

LEMMA 12.5. *For all* $s \in [hU \,..\, (h + 1)U - \langle \text{kill} \rangle)$ *there is a* q *in* $[s \,..\, s + \langle \text{kill} \rangle)$ *for which* $\mathscr{I}_q$ *is empty. Consequently, for all* $q \geq hU + \langle \text{kill} \rangle$ *the triangles in* $\mathscr{I}_q$ *are covered by* $D(\mathscr{L}, 1)$ *with*

$$\mathscr{L} = \left(\bigcup_q \mathscr{L}_q \right)'.$$

The second statement follows because we built $\mathscr{I}_q$ by consecutive merging of elements of $\mathscr{L}_q$ to $\mathscr{I}_{hU}$.

Proof. Suppose that $\mathscr{I}_q$ never vanishes between s and $s + \langle\text{kill}\rangle$. Let us calculate the decrease of the size of $\mathscr{I}_q$ during this interval, combined with the sum of all increases since time hU. At singular points q, there is a possible increase by the size of $\mathscr{L}_q$. The sum of these increases is at most $18r$. There are at most $9r$ such points. For at least $\langle\text{kill}\rangle - 9r$ points in $[s\,..\,s + \langle\text{kill}\rangle)$, the decrease is at least 2. The total decrease from the size of $\mathscr{I}_{hU}$ thus is at least

$$2(\langle\text{kill}\rangle - 9r) - 18r = 2\langle\text{kill}\rangle - 36r > 9\langle\text{sick}\rangle r.$$

by the definition of $\langle\text{kill}\rangle$. But $9\langle\text{sick}\rangle r$ is the upper bound on the size of $\mathscr{I}_{hU}$. The size of $\mathscr{I}_q$ would thus have decreased below 0. The contradiction proves the lemma. □

Proof of Lemma 12.3. Let the space–time point (q, i, j) be singular. It follows from Lemma 12.4 and the definition of $\mathscr{I}_q$ that (i, j) is in a deflation of an element of $\mathscr{I}_q$, thus it is in $D((\mathscr{K} \cup \mathscr{L})', 1)$ since $(\mathscr{K} \cup \mathscr{L})'$ covers all sets $\mathscr{I}_q$. It follows from Lemma 12.5 that q is in $Y \cup [hU\,..\,hU + \langle\text{kill}\rangle)$, and that if $q > hU + \langle\text{kill}\rangle$ then (i, j) is in $D(\mathscr{L}, 1)$. □

12.5. Computation

To end the proof of the first statement of Lemma 12.2, let us remember that we assumed its assertion for h, and are proving it for $h + 1$.

Let us estimate the deviations at time $(h + 1)U$. Since we obtained an upper bound on the set of singular points during the space–time period of the computation, we can follow the proof of Theorem 10.1, i.e., the proof of Lemma 10.1. The program still consists of Q calls to the procedure *CompColumn*(s, t), and a last step copying *OutMem* to *InpMem*.

LEMMA 12.6. *If site (i, j) in column s has deviations on the InpMem track at time $(h + 1)U$ then either the duration of call s intersects the set Y or there was a singular event at (i, j) some later time.*

Proof. Only a singular event can create new deviations on the *InpMem* track. According to the Singularity Localization Lemma, the sites of the singular events are all covered by a deflation of the

set $(\mathscr{K} \cup \mathscr{L})'$. Therefore each row contains at most $|\mathscr{K}| + |\mathscr{L}| \leqslant 10\langle\text{sick}\rangle r$ new deviations on the *ImpMem* track. Each row contains at most $\langle\text{dev}\rangle r$ old deviations, making the total $(10\langle\text{sick}\rangle + \langle\text{dev}\rangle)r = \langle\text{corr}\rangle r$. Under the Size Condition 12.1, we can construct a code Algeb just as in Section 10.3, to correct this many deviations (error bursts, in the original terminology, but each "burst" is confined here to one cell). Therefore if a call s has no singular event its computation starts with the correct input and it will write the correct value on the *OutMem* track.

Lemma 12.3 says that the times of the singular events are covered by $Y \cup [hU\,..\,(hU + \langle\text{kill}\rangle))$. After the first idling steps of the program, the first call $s = 0$ begins at time $hU + 2Q$, which is, according to inequality (12.2), greater than $hU + \langle\text{kill}\rangle$. Therefore if a call s intersects with the time of a singular event then it intersects with the set Y. □

To finish the proof of the first statement of Lemma 12.2, we estimate the number of columns s for which either the duration of the sth call intersects the set Y or there was a singular event at some later time in the column.

The set Y consists of at most $9r$ intervals of length $\langle\text{kill}\rangle$. According to inequality (12.2) we have $\langle\text{kill}\rangle < Q$. Therefore each of these intervals can intersect the duration of at most two calls. This is at most $18r$ calls. The number of columns s in any row in which a singular event happened after iteration s can be estimated by $|\mathscr{L}| \leqslant \langle\text{sick}\rangle r$. The total number of deviations in any row at time $(h + 1)U$ is thus at most $18r + \langle\text{sick}\rangle r = \langle\text{dev}\rangle r$. □

12.6. Reaching Consensus in the Presence of Noise

This section applies the technique developed in the present section to the model introduced in Section 11.3. The notion of triangles, the constant $\langle\text{cons}\rangle$ etc. used here are therefore those used in Section 11.3.

THEOREM 12.2. *Let $x[t, u]$ be an evolution over $\mathbf{Z}^+ \times \mathbf{Z}_m^2$, and r an integer less than m. Suppose that x has at most r faults with respect to the rule R^+. Then there is an integer $b = 0$ or 1 such that for all $t > \langle\text{cons}\rangle(r + 1)m$ there is a set of triangles of total size less than $4r$ covering the set of sites u with $x[t, u] \neq b$. If at time 0, there is a set of triangles of total size less than $4r$ covering the set of sites u with $x[0, u] \neq b$ then $b = b_.$.*

The bound $\langle \text{cons} \rangle (r+1)m$ is probably too high. We conjecture that if $r = o(m)$ then $O(m)$ can be written in its place.

Notice that about m^2 processors are used here to fight r faults, where r is approximately m. This is analogous to the results in [Ly] where, in case of r *permanent* faults, approximately an r by r array of processors is needed to achieve consensus. (By permanent faults, we mean cells that can act arbitrarily.) The rule R^+ is sensitive to even a small number of permanent faults. Two permanent faults can keep an arbitrarily large triangle forever from shrinking. It is, an interesting open question whether, in case the consensus must be achieved by a homogeneous array of automata, an r by r array is necessary to achieve consensus in the presence of r transient faults. The rule R^+ can certainly be fooled by as many faults as the size of the torus, whether they are placed at one time, or at a constant number of places but for a long time. Is this true of all rules?

Sketch of proof. The proof of the second statement of the theorem is similar to (only much simpler than) the proof of the Singularity Localization Lemma 12.3. We build sets $\mathcal{I}_q$ of triangles and estimate their size increases and decreases.

To prove the first statement of the theorem, note that there is a time interval of length $\langle \text{cons} \rangle m$ between times 0 and $\langle \text{cons} \rangle (r+1)m$ when no faults occur. To this time interval, we can apply Theorem 11.1. □

13. COLONIES

In Theorem 12.1, we found a medium M_1 that, under certain noise conditions, reliably simulates a given medium Med_0. There is no universal medium for this sort of simulation, even if we fix the medium Med_0. Indeed, the cell capacity of the medium M_1 crucially depended on the pair (U, r). Nevertheless, we will bring all the different simulated and simulating media to a "common denominator." To standardize the simulated media, let us first choose some universal Turing machine Turing. For any medium Med_0, let us encode the alphabet S_{Med_0} into binary strings of fixed length $|\mathsf{Med}_0|$. This binary encoding is called *expanding*: we expand a symbol s into the string $\mathsf{bin}(s)$. Now let Prog_0 be some program (e.g., first one) that computes on Turing the transition function Med_0 from its nine arguments when each state s is represented by $\mathsf{bin}(s)$. The program Prog_0 describes the parameter $|\mathsf{Med}_0|$ as well.

The common denominator will be more interesting to find for the *simulating* media M_1 and different noise conditions (U, r). We define a new *universal medium* Univ. In the previous section, we simulated each Q'-square of Med_0 by a Q-square of M_1. Now for some integers P, T where T is a multiple of P, we subdivide the plane into squares of size P that will simulate, in a working period of size T, the cells of medium M_1. Just as we used a name for the Q-squares of medium M_1 calling them "alliances," we will use a name for the P-squares of medium Univ, calling them *colonies*.

Actually, we will dispose of the intermediate medium M_1 altogether. Instead of saying that we simulate M_1 by Univ and combine this with the simulation of Med_0 by M_1, we will just say that we simulate the Q'-squares of Med_0, using QP-squares of Univ called *alliances*. Here the medium Univ does not depend on Med_0 or any of the parameters.

13.1. Cells with Nonunit Size and Worktime

Later, we will consider evolutions of Univ that were obtained by decoding from some other evolution of another medium. In such cases, it is useful to measure the size and worktime of cells by the cost in space and time in the simulating medium. For this later goal, we generalize slightly the model considered so far. We introduce two, not necessarily integer, parameters, the *cell size* $\alpha \geq 1$ and *cell worktime* $\beta \geq 1$. If there are m cells across the torus then the space $\mathbf{W}$ will consist now not only of sites with integer coordinates but of all points with coordinates (i, j) where (i, j) are real numbers in $[0, m\alpha)$. Each cell of the space $\mathbf{W}$ occupies a square of size α. Similarly, each transition of the medium Univ will take β time units. Of course, the notions of evolution and trajectory are modified accordingly: evolution $x[t, u]$ is defined only for instants t that are multiples of β. Let us introduce the integers

$$P' = P/\alpha, \qquad T' = T/\beta.$$

During T units of time, only T' actual state transitions of the medium Univ occur, and the number of cells across a colony is only P'. It is reasonable to require that T' be significantly larger than P', in order for the colony to have time to receive information from the neighbor colonies and some time to process it. Let us require

$$T' \geq 9P'. \tag{13.1}$$

The numerical parameters determining the model are thus

$$P', T', Q, U, Q', U', P, T, r.$$

13.2. Noise

To the assumption that the medium M_1 has at most r faults in every U-cube, we make a corresponding assumption under the new conditions. The correspondence cannot be perfect since sites of M_1 correspond to squares of size P and working period T. Since we want to look at the colonies only at times $h = qT$, and since the effect of faults can be expected to spread during each work period, we account for faults by the abstract notion of *noise*. The noise is a given union $\mathcal{N}$ of some T-cubes.

Let us generalize the probabilistic noise bounds of Section 9. Let us be given some probability distribution on all unions of T-cubes. This distribution gives rise to a random union $\mathscr{E}$ of T-cubes. For a parameter p, we say that the distribution of $\mathscr{E}$ is *p-bounded* if for all k, all finite sets $A = C_1 \cup \cdots \cup C_k$ of disjoint T-cubes, we have

$$\mathrm{Prob}(A \subset \mathscr{E}) \leqslant p^k.$$

The noise in a UT-cube C will be called (U, r, T)-*sparse* if at most r of the T-cubes of C belong to it. Since in the present section, U, r, T are fixed, we will not indicate the dependence on them. The noise is sparse over a union of UT-cubes if it is sparse on each of them. It is sparse over some other space–time set E if it is sparse over a union of cubes covering E.

Faults in M_1 were *transient*. After the fault happened, a cell of M_1 obeyed again the transition rule. Why would a fault in a colony have only a transient effect? The simulation that the colonies perform most probably depends on some structure within the colony that will be destroyed by the fault. Our present solution to this problem will be to *define it away*. We will make further *restrictions on the permissible evolutions*. The particular form of the restrictions is dictated by our needs to be able to *prove* them under certain circumstances.

Since T-squares $[T; i, j]^2$ will appear frequently below, we give them a name: we call them *clusters*. For a cluster E, we call the set $\Gamma(E, T)$ its *neighborhood*. It is the union of nine clusters.

13.3. Parameters and Phase Variables

Before we present the structure restoration condition we must say something about the notion of structure.

To find the boundaries of the colony at the beginning of the work period, let us mark the left-most column and the top row by a special symbol.

To give us some freedom for later tuning, let us add one more parameter: a number modif of yet unspecified role that will later be used to modify the action of Univ. Let us set aside the row second from the top for the parameters P', Prog_0, modif, r abbreviated as $P', \ldots, r$. This area is called the *parameter field.*

In the construction of the medium M_1 in the preceding section, three variables called the *phase variables* played a special role. The role of these variables will here also be a distinguished one. We set aside a fixed field for them in the colony called the *phase variable field:* the row below the parameter field.

Now we are ready to define a standard initial state for the colony. Let us assume that a special symbol set

$$S_{\mathsf{init}} \subset S_{\mathsf{Univ}}$$

and an element

$$\mathit{topleft} \in S_{\mathsf{Univ}} \setminus S_{\mathsf{init}}$$

are given. A colony $C = [P; i, j]^2$ is called *healthy* at time qT (with respect to the evolution x) if

- $x[qT, u] = \mathit{topleft}$ for all sites u that are either in the top row or in the leftmost column of C.
- $x[qT, u] \in S_{\mathsf{init}}$ for all sites in C not in the top row or left-most column.
- The first two integers in the parameter field are P' and T'.

The notion of health thus depends on the parameters P', T'. This is not essential, but convenient.

A healthy colony is $[P; i, j]^2$ is *legal* with respect to the parameters $P', \ldots, r$ at time qT if the parameter field contains these parameters, and the phase variable contains the integers $q \bmod U$, $i \bmod Q$, and $j \bmod Q$.

The medium Univ *supports colonies* if the following condition holds, for all P', T' with $P' \mid T'$, for all trajectories y with unit cell sizes and worktimes and for all colonies C healthy at time 0.

- C is healthy at time T.
- The configuration of C at time T depends only on the configuration of its nine neighbors (including itself) at time 0.
- Assume that in addition, two of the northern, southeastern, and southwestern neighbor colonies are legal at time 0 with respect to some parameters $P', \ldots, r$. Then C is legal at time T.

We will be interested only in media Univ supporting colonies. Colony support will be easy to add to the program of a medium that does not have it.

Let φ be a simulation whose code maps Med_0-configurations $z[B]$ over a square B of size Q' into Univ-configurations $x[C] = \varphi_*(z[B])$ over a square C of size QP. We say that the simulation φ is *standard* if for each configuration $x[C]$ of the form $\varphi_*(z[B])$, all colonies in the alliance $x[C]$, are legal with the value 0 in the phase variable τ.

13.4. Damage, Quarantines

We introduce a way to keep track of the bad parts of a configuration $x[qT]$. For all q, we introduce a set $\mathsf{Damage}(qT) \subset \mathbf{W}$. This set consists not only of points that are sites (i.e., whose coordinates are multiples of α) but may contain also other points of the torus $\mathbf{W}$. For all clusters C the set $C \cap \mathsf{Damage}(qT)$ will be the union of a finite number of convex polygons. The set $\mathsf{Damage}(0)$ will be assumed given, and the set $\mathsf{Damage}(qT)$ will be defined inductively for $q > 0$.

It is often convenient to take the set $\mathsf{Damage}(qT)$ into account by a set of disjoint triangles containing it. Let C be a union of clusters. We will say that a set $\mathscr{I}$ of disjoint triangles is a *quarantine* at time qT on C if we have

$$C \cap \mathsf{Damage}(qT) \subset \bigcup D(\mathscr{I}, T).$$

The deflation $D(\mathscr{I}, T)$ is used in the above definition because the quarantine must cover the damage by a well-separated system of triangles.

For all C and qT there is a *minimal quarantine*. Indeed, the set $C \cap \mathsf{Damage}(qT)$ is the union of a finite set $\mathscr{F}$ of convex polygons. The set $D(\mathscr{F}, -T)'$ (see Section 11.2) is then the minimal quarantine.

Suppose that $\mathsf{Damage}(qT)$ is defined. Let C be a cluster. Let $\mathscr{I}$ be the minimal quarantine for $C' = \Gamma(C, T)$. We define

$$C \cap \mathsf{Damage}((q+1)T) = \begin{cases} C \cap \bigcup D(\mathscr{I}, T+P) & \text{if } \mathscr{N} \cap ([T;q] \times C') = \varnothing, \\ C & \text{otherwise.} \end{cases}$$

Thus, the damage shrinks in the absence of noise, and it maximally grows in the presence of noise. We will be interested only in evolutions that satisfy the following condition.

CONDITION 13.1 (RESTORATION). *All colonies disjoint from* $\mathsf{Damage}(qT)$ *are legal at time* qT.;

This condition says, implicitly, that if, in the neighborhood of a cluster C, the illegal colonies are confined to well-separated triangles then, in the absence of noise, these triangles shrink.

We call a cluster $C = [T; i, j]^2$ *regular* at time qT if the following two conditions are satisfied.

- The neighborhood $C' = \Gamma(C, T)$ of C does not intersect with the noise during the interval $[T; q]$.
- $C' \cap \mathsf{Damage}(qT) = 0$.

Otherwise, we will call C *singular* at time qT. We will also call the triple (q, i, j) singular.

The following condition requires that the evolution of Univ we are considering should behave like a trajectory on regular clusters.

CONDITION 13.2 (COMPUTATION). *Let the cluster* C *be regular at time* qT. *Then the configuration* $x[(q+1)T, C]$ *is what it would be if* x *was a trajectory of* Univ *starting from the same configuration* $x[qT, C']$.

13.5. Summary of the Primitive Notions

We can divide the ingredients of the model into two parts. The first group contains those ingredients chosen by us, the machine

designers. The second group relates to the constraints imposed on the evolution that is otherwise chosen by nature.

Our choice:

- The parameters $P', T', Q, U, Q', U', P, T, \mathsf{Prog}_0, \mathsf{modif}, r$. (The notions of health and legality are now defined.)
- The colony-supporting medium Univ.
- The standard simulation φ.

Nature's choice:

- The evolution $x[t, i, j]$;
- The noise $\mathcal{N}$;
- The set Damage(0).

The triple $(x, \mathcal{N}, \mathsf{Damage}(0))$ will be called a *self-correcting evolution* if it satisfies the Restoration Condition and the Computation Condition. Thus, when we say that the evolution x is self-correcting, we assume that the other two ingredients of the triple are also given. The notion of a self-correcting evolution depends on the parameters $P', \ldots, r$ and the medium Univ.

A *self-correcting perturbation* of a trajectory y in the range of the simulation φ is a self-correcting evolution $(x, \mathcal{N}, \emptyset)$ such that x coincides with y at time 0. A random self-correcting perturbation of a trajectory y is a *self-correcting p-perturbation* if the random noise $\mathcal{N}$ is p-bounded. The problem of how to find random self-correcting perturbations with a small noise probability will be partly addressed in Section 13.7.

13.6. Size Conditions and Simulation Theorem

The theorem stated in the present section is analogous to the theorem of the previous section, using the Restoration Condition in place of Lemma 12.1. To emphasize the analogy and save notation, we will use some of the names and notation used in the preceding section with slightly different but analogous meaning. For the present section, let us define the constants

$$\begin{aligned} \langle \text{sick} \rangle &= 72, \\ \langle \text{dev} \rangle &= 180, \\ \langle \text{corr} \rangle &= \langle \text{dev} \rangle + 27(\langle \text{sick} \rangle + 6). \end{aligned} \tag{13.2}$$

In the new Size Condition below the lower bound on QP' is needed for the existence of a code correcting enough error bursts and enough simulation space. The lower bound on P' makes sure that numbers smaller than U, Q and T as well as the program of the transition rule Med_0 are representable within a colony. The bound for UT' is similar to the one found in Size Condition 12.1, with UT' replacing U and $(QP/T)QT'$ replacing Q^2. The multiplier QP/T is the number of repetitions in the program. The expression QT' measures the time taken by Q colony work periods, i.e., the time to get any information across the alliance. The lower bound on T' is the same as (13.1). The divisibility assumptions serve only convenience. The constant Step_0 measures the time needed by the Turing machine Turing with program Prog_0 to compute the transition function Med_0. We assume that the size $|\mathsf{Prog}_0|$ of the program also measures the space needed for the computation.

CONDITION 13.3 (SIZE).

$$QP' > \max\left(3Q'|\mathsf{Med}_0|, Q'|\mathsf{Med}_0| + \frac{9\langle \text{corr} \rangle rP'T}{P}\right), \tag{13.3}$$

$$P' > \max(\log U, \log T', |\mathsf{Prog}_0|, |\mathsf{modif}|), \tag{13.4}$$

$$UT' > \frac{U'}{Q'}\frac{QP}{T}QT' \log(QP')\mathsf{Step}_0, \tag{13.5}$$

$$T' > 9P', \tag{13.6}$$

$$P \mid T, \qquad \frac{T}{P} \Big| Q, \qquad Q \mid U.$$

Let us show how to satisfy all these conditions but (13.4). This remaining condition can be easily satisfied if we do not choose our parameters to be exponentially large compared to P'. Let Q', U', Prog_0 and modif be arbitrary. Let P' be large enough such that $|\mathsf{Prog}_0| < P'$. Let $P > P'$. Let $T > T'$ be a multiple of P large enough for (13.6). Let Q be a multiple of T/P large enough for (13.3). Let U be a multiple of Q large enough to satisfy (13.5). In this way, all Size Conditions will be satisfi

THEOREM 13.1. *There is a medium* Univ *supporting colonies such that for all* $P', \ldots, r$ *satisfying the Size Condition* 13.3 *with* modif $= 0$, *there is a standard simulation* φ *of* Med_0 *by* Univ *such that the following holds.*

Let y *be a trajectory of* Univ *in the range of* φ *over the space* $\mathbf{W} = \mathbf{Z}^2_{nQP}$, *and let* $(x, \mathcal{N}, \emptyset)$ *be a self-correcting perturbation of* y. *If the noise* $\mathcal{N}$ *is* (U, r, T)*-sparse then for all nonnegative integers* h *we have*

$$\varphi^*(x[hUT, \mathbf{W}]) = \varphi^*(y[hUT, \mathbf{W}]).$$

This theorem will be proved in the next section. The rest of the present section discusses the ways in which we will find self-correcting perturbations.

13.7. Implementations

We can view the medium Univ as the ideal, programmable medium to be used for computation. In order to achieve reliability in the real world, we have to find some physical "implementation" of Univ, in such a way that arbitrary evolutions of M are decoded into self-correcting evolutions of Univ.

The main ingredient of the implementation is a simulation ψ. Let the medium Univ support colonies. Let P', T', P, T be given satisfying

$$T' \geqslant 9P', \qquad P' \leqslant P, \qquad T' \leqslant T, \qquad P \mid T. \tag{13.7}$$

Let M be a medium and let ψ be a simulation with parameters P, P, T, T, that maps from the space $\mathbf{W}^*$ of Univ into the space $\mathbf{W}$ of M. (With the introduction of nonunit cell sizes, it is not restrictive to require that the simulating colonies in M have the same size and worktime as the simulated ones in Univ.) Let z be an evolution of M. We define the evolution $x = \psi^*(z)$ for values $t = qT$ as $x[qT] = \psi^*(z[qT])$, i.e., we obtain x by decoding from z. We define $x[t, u]$ for values t that are not a multiple of T in an arbitrary way.

Besides the decoding ψ^*, we must say how noise and the damage are found in the space $\mathbf{W}^*$. Let

$$\mathbf{\Psi} = (\psi, \mathcal{N}_\Psi, \mathsf{Damage}_\Psi)$$

be a triple where ψ is the code assumed above. To each evolution, z of M, the mapping $\mathcal{N}_\Psi(z)$ orders a union of T-cubes in the space–time over $\mathbf{W}^*$. The mapping $\mathsf{Damage}_\Psi(z)(0)$ orders a subset of $\mathbf{W}^*$ to each evolution z.

Let $0 < p_1, p_2 < 1$. We call the triple $\mathbf{\Psi}$ an *implementation* with parameters

$$P', T', P, T, p_1, p_2$$

If the following properties hold.

- Suppose that the parameters $Q, \ldots, r$ supplement P', T', P, T in a way satisfying the Size Conditions. Then for all evolutions z of M, the triple

$$(\psi^*(z), \mathcal{N}_\Psi(z), \mathsf{Damage}_\Psi(z)(0))$$

  is a self-correcting evolution.
- Suppose further that for some trajectory y of Univ, the random evolution ζ is a p_1-perturbation of $\psi_*(y)$ (in the sense of Section 7). Then the triple $[\psi^*(\zeta), \mathcal{N}_\Psi(\zeta), \mathsf{Damage}_\Psi(\zeta)]$ is a self-correcting p_2-perturbation of y.

Of course, the notion of implementation depends on the choice of Univ. Our goal is to find implementations with constant p_1 and small p_2. This will be achieved in the following way. First, we find a trivial implementation in the paragraph below. Then, in Section 16, we show how to turn an implementation into one with a smaller p_2, using a special self-simulation of Univ that we will call an *amplifier*.

Let us give a trivial but important example $\mathbf{\Psi}_0$ of an implementation.

LEMMA 13.1. *Let* P', T', P, T *be parameters satisfying* (13.7), *and* $0 < \varrho < 1$. *There is an implementation with the parameters* $P', T', P, T, \varrho, \varrho$.

Proof. We introduce a new medium M over the state set $S_M = S^{P'^2}_{\mathsf{Univ}}$ whose states are the healthy configurations over clusters $[T; i, j]^2$. The transition rule computes from the states of nine neighbor cells in one step what would have been computed in T'

steps by Univ from the corresponding clusters. The encoding ψ_{0*} of the code ψ_0 are equal to the identity function, defined on the healthy cluster configuration. In the space $\mathbf{W}$ of the medium M, we introduce cell size and cell workperiod T.

For an evolution z of M, a cube $[T; q, i, j]^3$ in $\mathbf{W}^*$ belongs to the noise $\mathcal{N}_{\Psi_0}(z)$ if the corresponding "cube" $[T; q, i, j]^3$ in the evolution z contains a fault. The set $\mathsf{Damage}_{\Psi_0}(z)(0)$ is the union of illegal clusters at time 0.

The Computation Condition will be satisfied automatically for an evolution x defined this way. Health is not a problem, since in this evolution, all colonies are healthy at all times, by definition. The Restoration Condition can be proved by induction on q since the medium Univ supports colonies. □

14. UNIVERSAL ROBUST SIMULATION WORKS

14.1. The Form of the Universal Simulation

The program of the simulation of an arbitrary medium Med_0 by colonies of Univ will be almost like the one for cells of M_1. In particular, the period-for-period application of the Toom Rule results in the colony-supporting property of Univ. Let us still point out some small differences.

14.1.1. *The Error-Correcting Code*

We represent each row of a Q'-square as a binary string of length $Q'|\mathsf{Med}_0|$. Let n be the greatest integer $\leqslant P'T/P$ of the form $2 \cdot 3^s$. We subdivide each row into segments of length n, using possibly a partial segment at the end: there are K segments, with

$$K = \lceil Q'|\mathsf{Med}_0|/n \rceil.$$

We use Theorem 8.1 to find an error-correcting code Algeb encoding these strings into strings of length

$$Nn = (K + 8\langle \mathrm{corr} \rangle r)n \leqslant Q'|\mathsf{Med}_0| + (8\langle \mathrm{corr} \rangle r + 1)P'T/P$$

and correcting $4\langle \mathrm{corr} \rangle r$ bursts of errors of length n. By Size Condition 13.3, these codewords are smaller than QP', therefore

will fit into a row of the alliance. The n-cell segments can reach across colony boundaries.

Now the code φ is defined much as in Sections 10 and 12. Decoding is the same. The encoding creates legal colonies that are about to start the working period of the alliance.

14.1.2. *Variables into Fields*

Some *variables* will now occupy several cells. This will happen, e.g., to the phase variables. Since the size of their field depends on Q, U, they cannot be part of the state of a single cell. We assigned to them the phase variable field.

14.1 3. *Toom Phase Protection*

To make colony support complete, we must apply Toom's Rule to the parameters and phase variables in every work period of every small colony. As long as a colony is healthy it does not cause any problem to do this. The parameters and phase variables of the northern, southeastern, and southwestern neighbor small colony are read in. After the application of the Toom's Rule to the parameters, the result is written into the parameter field. After this, using Q and U, the generalized Toom's Rule is applied to the phase variables. This operation is repeated in every T-interval, e.g., in parallel (using separate tracks) with everything else done by the colony.

14.1.4. *Computation*

Even after decoding by the code Algeb, the cell states of Med_0 are represented by binary strings of length $|\mathsf{Med}_0|$. The simulation part *Compute* of the program will be less direct now than it was when one cell of the simulated medium was represented in one cell of the simulating medium. The simulation of one step of Med_0 will be carried out for each cell of Med_0 with the help of the program Prog_0, simultaneously on all these strings.

14.1.5. *Output Columns*

Another slight difference between the new program and the old one is that in the old program, in the procedure *Output*(s, t), only

a single column was written. Now this procedure will write in a column of width T at once, i.e., in the part of the alliance whose projection is the interval $[T; s]$. This is the reason for the factor QP/T in the Size Conditions.

14.2. Local Health

Just as clusters (T-squares) are often more convenient than colonies, UT-squares are often more convenient than alliances. They will be called *alliance clusters.*

We will say that an alliance cluster C is *locally healthy* at time qT if it has a quarantine $\mathscr{I}$ with

$$|\mathscr{I}| < \langle \text{sick} \rangle rT.$$

For q in $[U; h]$, let

$$C_q = \Gamma(C, ((h+1)U - q)T). \tag{14.1}$$

This is the set of those sites whose state at time qT can have some effect on the state of the UT-square C at time $(h+1)U$. The next lemma is analogous to Lemma 12.2.

LEMMA 14.1. *Assume that the conditions of Theorem 13.1 hold, and the medium* Univ *and the code φ are defined as above. Let C be an alliance cluster. Suppose that at time hUT, the following statements hold:*

- *In each row of each alliance of C_{hU}, the deviations of the InpMem track of evolution x from the trajectory y are covered by $\langle \text{dev} \rangle r$ T-intervals.*
- *C_{hU} is locally healthy.*

If the noise is sparse over $C_{hU} \times [UT; h]$ then the same statements hold at time $(h+1)UT$.

The two statements of the lemma hold, of course, for $h = 0$. Therefore the lemma implies that they hold for all h, which will prove the statement of Theorem 13.1. □

14.3. Triangle Development

We begin with the lemma analogous to Lemma 12.3. Let

$$\langle \text{kill} \rangle = 11 \langle \text{sick} \rangle rT/P.$$

The relation

$$10 \langle \text{kill} \rangle < Q \tag{14.2}$$

follows immediately from the Size Conditions 13.3.

For any set $\mathscr{I}$ of triangles, let us denote

$$\mathscr{I}^0 = D(\mathscr{I}, T).$$

LEMMA 14.2 (SINGULARITY LOCALIZATION). *There is a set Y of times in $[U; h]$ consisting of at most $9r$ intervals of length $\langle \text{kill} \rangle$, and sets $\mathscr{K}$, $\mathscr{L}$ of triangles with*

$$|\mathscr{L}| \leqslant \langle \text{sick} \rangle rT, \qquad |\mathscr{K}| \leqslant 9 \langle \text{sick} \rangle rT \tag{14.3}$$

with the following properties. Let $\mathscr{J} = (\mathscr{K} \cup \mathscr{L})'$. At all times qT with q in $[hU \,..\, (h+1)U]$, the set $\mathscr{J}$ is a quarantine for C_q. For $q > hU + \langle \text{kill} \rangle$, the same is true for $\mathscr{L}$.

Suppose that cluster E in C_q is singular at time qT for $q < (h+1)U$. Then q is in $Y \cup [hU \,..\, hU + \langle \text{kill} \rangle)$, and $\Gamma(E, T)$ is intersected by $\mathscr{J}^0$. If $q \geqslant hU + \langle \text{kill} \rangle$ then $\Gamma(E, T)$ is intersected by the smaller set $\mathscr{L}^0$.

This lemma implies the second statement of Lemma 14.1. Indeed, it says that the set $\mathscr{L}$, of size $\leqslant \langle \text{sick} \rangle rT$, is a quarantine for C at time $(h+1)U$.

14.3.1. Noise

The set $C_{hU} = \Gamma(C, UT)$ consists of nine alliance clusters. The domain of space–time involved is covered by the nine cubes above these squares. Since the noise is sparse, at most $9r$ of the T-cubes in this domain belong to the noise. For a number q in $[U; h]$, let $\mathscr{F}_q$ be the set of projections of the T-cubes belonging to the noise in this

domain at time qT. The number q will be called *singular* if the set $\mathscr{F}_q$ is nonempty. Otherwise, it is called *regular*. Let Y be the union of intervals $[q\,..\,q + \langle\text{kill}\rangle)$ for all singular numbers q.

To each T-square F in $\mathscr{F}_q$, we order a triangle $L = D(\Gamma(F, T), -T)$ with size $|L| = 8T$. Performing this operation for each element F of $\mathscr{F}_q$ we arrive at the set $\mathscr{L}_q$ of triangles. Let us define $\mathscr{L} = (\bigcup_q \mathscr{L}_q)'$. We have

$$|\mathscr{L}| \leqslant 9 \cdot 8rT = \langle\text{sick}\rangle rT.$$

14.3.2. The Triangle Sets $\mathscr{I}_q$

Let us define for all q a set $\mathscr{I}_q$ of disjoint triangles such that the following proposition holds.

LEMMA 14.3. *At time qT, the set $\mathscr{I}_q$ is a quarantine for C_q. If E is a cluster in C_q singular at time qT then $D(\mathscr{I}_q \cup \mathscr{I}_{q+1}, T)$ intersects $\Gamma(E, T)$.*

The set $\mathscr{I}_q$ will be defined inductively. Since we assumed that C_{hU} is locally healthy at time hUT, for each of the nine alliance clusters in it there is a quarantine $\mathscr{K}_i$ of size less than $\langle\text{sick}\rangle rT$. We define

$$\mathscr{I}_{hU} = \mathscr{K} = \left(\bigcup_{i=1}^{9} \mathscr{K}_i\right)'.$$

Then $\mathscr{K}$ is a quarantine for C_{hU}. We proceed to the definition of $\mathscr{I}_q$ for $q > hU$. Let us assume that $\mathscr{I}_q$ is defined. We define

$$\mathscr{I}_{q+1} = \begin{cases} (D(\mathscr{I}_q, P) \cup \mathscr{L}_q)' & \text{if } q \text{ is singular,} \\ D(\mathscr{I}_q, P) & \text{otherwise.} \end{cases}$$

We see that all sets $\mathscr{I}_q$ are covered by $\mathscr{J} = (\mathscr{K} \cup \mathscr{L})'$.

Proof of Lemma 14.3. This lemma is essentially an immediate consequence of the definitions.

Let us prove first that $\mathscr{I}_q$ is a quarantine for all q. We use induction on q. For $q = hU$, the statement follows from the definition of $\mathscr{I}_q$. Suppose it is true for q. If q is regular, it follows by the definition of $C_{q+1} \cap \mathsf{Damage}((q + 1)T)$ given in 13.4.

If q is singular, it will also follow by this definition, applied to a smaller area B that is unaffected by the noise. We define

$$B = C_{q+1} \setminus \bigcup \Gamma(\mathscr{F}_q, T).$$

The set $\Gamma(B, T)$ is disjoint from the noise during the interval $[T; q]$. It follows from the damage definition that $D(\mathscr{I}_q, P)$ is a quarantine for B at time $(q + 1)T$. We have now

$$\mathsf{Damage}((q + 1)T) \subset D(\mathscr{I}_q, P + T) \cup \Gamma(\mathscr{F}_q, T).$$

By the definition of $\mathscr{L}_q$, the second term on the right-hand side is contained in $D(\mathscr{L}_q, T)$. Using the definition of $\mathscr{I}_{q+1}$, this proves that it is a quarantine.

Suppose now that the cluster E in C_q is singular at time qT. Then one of the two conditions of regularity is not satisfied. If this is the first one then the set $\mathscr{I}_{q+1}$ contains $\mathscr{L}_q$ whose deflation contains E. If the second one is not satisfied then by the Restoration Condition the set $D(\mathscr{I}_q, T)$, as the deflation of a quarantine, intersects $\Gamma(E, T)$. □

14.3.3. Vanishing Triangles

Lemma 14.2 will follow from Lemma 14.3, the fact that $\mathscr{J}$ covers $\bigcup_q \mathscr{I}_q$, and the fact, to be proved in the present paragraph, that $\mathscr{L}$ covers $\bigcup_{q > \langle\text{kill}\rangle} \mathscr{I}_q$. □

LEMMA 14.4. *For all $s \in [hU\,..\,(h + 1)U - \langle\text{kill}\rangle)$ there is a q in $[s\,..\,s + \langle\text{kill}\rangle)$ for which $\mathscr{I}_q$ is empty. Consequently, the triangles in $\bigcup_{q > \langle\text{kill}\rangle} \mathscr{I}_q$ are covered by $\mathscr{L}$.*

Proof. Suppose that $\mathscr{I}_q$ never vanishes between s and $s + \langle\text{kill}\rangle$. Let us calculate the decrease of the size of $\mathscr{I}_q$ during this interval, combined with the sum of all increases since time hU. At singular points q, there is a possible increase by the size of $\mathscr{L}_q$. The sum of these increases is at most $|\mathscr{L}|$.

There are at least $\langle\text{kill}\rangle - 9r$ regular numbers in the interval $[s\,..\,s + \langle\text{kill}\rangle)$. At each of these times, there is a deflation by P, i.e., a size decrease by $2P$. The total size decrease at these times is at least $2(\langle\text{kill}\rangle - 9r)P$. Combining with the possible increase of $|\mathscr{L}|$

and the original size $|\mathscr{K}|$, this gives

$$|\mathscr{I}_{s+\langle \text{kill} \rangle}| \leqslant |\mathscr{K}| + |\mathscr{L}| + 18rP - 2\langle \text{kill} \rangle P$$
$$< (10\langle \text{sick} \rangle + 18)rT - 2\langle \text{kill} \rangle P \leqslant 0.$$

By the definition of $\langle \text{kill} \rangle$, the size would thus have decreased below 0. This contradiction proves the lemma. □

14.4. Computation

Now we prove the first statement of Lemma 14.1 for $h + 1$. Let A be an alliance in the cluster C. Let us estimate the colonies with deviations at time $(h + 1)U$ in a row of A. Since we obtained an upper bound on the set of singular points during the space–time period of the computation, we can follow the proof of Theorem 10.1, i.e., the proof of Lemma 10.1. The program still consists of calls to the procedure *CompColumn*(s, t), for $s = 1, \ldots, QP/T$, and a last step copying *OutMem* to *InpMem*.

The following simple geometrical lemma is useful.

LEMMA 14.5. *Let $\mathscr{I}$ be a set of triangles. The number of clusters E in a horizontal row with the property that $\Gamma(E, T)$ intersects $\mathscr{I}^0$ is at most $3|\mathscr{I}|/T$.*

Proof. Let us estimate the number of elements of $\mathscr{I}$. We can assume that each element J of $\mathscr{I}$ has a size of at least $2T$, since otherwise, $D(J, T)$ would be empty. Therefore their number is at most $|\mathscr{I}|/(2T)$. It is easy to see that for a triangle J, the number of clusters E in a row for which $\Gamma(E, T)$ intersects J is at most $|J|/T + 4$. Therefore the total number of intersected T-intervals in a row is at most

$$\frac{|\mathscr{I}|}{T} + 4\frac{|\mathscr{I}|}{2T} = 3\frac{|\mathscr{I}|}{T}.$$ □

LEMMA 14.6. *If site (i, j) in column s will have deviation on the InpMem track at time $(h + 1)UT$, then either the duration of call s intersects the set Y or there was a singular event at (i, j) at some later time.*

Proof. Before time $(h + 1)UT$, only a singular event can create new deviations on the *InpMem* track. According to Lemma 14.2,

if cluster E is the site of a singular event then its neighborhood $\Gamma(E, T)$ intersects $\mathscr{I}^0$. According to Lemma 14.5, the number of such clusters in a row is at most $3|\mathscr{I}|$. Each row contains at most $\langle \text{dev} \rangle r$ T-intervals with old deviations, making the total

$$\langle \text{dev} \rangle r + 3|J|/T \leqslant (\langle \text{dev} \rangle + 27(\langle \text{sick} \rangle + 8))r = \langle \text{corr} \rangle r.$$

These T-intervals intersect at most four times this many n-intervals, according to the definition of the number n in the Section 14.1. The code **Algeb** was constructed to correct a pattern of $4\langle \text{corr} \rangle r$ bursts of errors of length n. Therefore if a call s has no singular event it will write the correct value on the *OutMem* track.

It follows from Lemma 14.2 that the singular numbers $q > hU + \langle \text{kill} \rangle$ are covered by the set Y. Since the call $s = 1$ begins only after $2QT'$ idling steps in the program and since we have inequality (14.2), the times of singular events during calls *CompColumn*(s) are covered by Y. □

Let us apply the above lemma to estimate the number of deviations at time $(h + 1)UT$. The set Y consists of $9r$ intervals of length $\langle \text{kill} \rangle$. By inequality (14.2), each of these intervals can intersect the duration of at most two calls. This is at most $18r$ calls. The number of intervals $[T; s]$ in any row in which a singular event happened after iteration s can be estimated similarly to the proof above by $3|\mathscr{L}|/T$. The total number of deviations in any row at time $(h + 1)U$ is thus at most.

$$3|\mathscr{L}|/T + 18r < 180r = \langle \text{dev} \rangle r.$$

This proves the first part of Lemma 14.1. The second part was proved after Lemma 14.4. □

14.5. The Spreading of Local Health

The second statement of Lemma 14.1 says that, under the conditions of Theorem 13.1, alliances always stay locally healthy. Let us strengthen this statement. A set $\mathscr{I}$ of triangles is a *local quarantine* for the alliance cluster C if there is a set $\mathscr{H}$ of triangles of size $\langle \text{sick} \rangle rT$ such that $(D(\mathscr{I}, UT) \cup \mathscr{H})'$ is a quarantine for C. We say that $\mathscr{I}$ is a local quarantine for a union of alliance clusters if it is a local quarantine for each of them.

The deflation by UT corresponds to the earlier deflation by T. Let us therefore define the abbreviation

$$\mathscr{I}^1 = D(\mathscr{I}, UT).$$

LEMMA 14.7. *Suppose that the medium* Univ *supports colonies and the parameters* $P', \ldots, r$ *satisfy the Size Conditions. Let* $(x, \mathscr{N}, \mathsf{Damage}(0))$ *be a self-correcting evolution. Let* C *be an alliance cluster,* h *a natural number, and* $\mathscr{I}$ *a local quarantine for* C_{hU} *at time* hUT. *Suppose that the noise* $\mathscr{N}$ *is sparse over the set* $C_{hU} \times [UT; h]$. *Then* $D(\mathscr{I}, QP)$ *is a local quarantine for* C *at time* $(h+1)UT$.

This lemma is a generalization of the second statement of Lemma 14.1 speaking about the preservation of local health. That statement becomes a special case with an empty set $\mathscr{I}$. The lemma is a step in our plan to *prove* the Restoration Condition for "higher order" colonies, simulated by alliances.

We will prove the above lemma by proving the following, somewhat stronger but less transparent, lemma which also contains a generalization for some parts of Lemma 14.2.

LEMMA 14.8. *Assume that the conditions of Lemma 14.7 hold. Let us define, for* $q \geqslant \langle \text{kill} \rangle$,

$$\mathscr{H}_q = D(\mathscr{I}^1, (q - \langle \text{kill} \rangle)P).$$

Assume that the noise is sparse over C_{hU} *during the time interval* $[UT; h]$. *Let the sets* $\mathscr{K}$, $\mathscr{L}$, Y *be defined as above. Then the set* $(\mathscr{I}^1 \cup \mathscr{K} \cup \mathscr{L})'$ *is a quarantine for* C_q *at time* qT *for all* q *in* $[hU \,..\, (h+1)U]$. *For* $q \geqslant \langle \text{kill} \rangle$, *the same is true for the smaller set* $(\mathscr{H}_q \cup \mathscr{L})'$.

Suppose that cluster E *in* C_q *is singular at time* qT. *Then* $\Gamma(E, T)$ *is intersected by*

$$D((\mathscr{I}^1 \cup \mathscr{K} \cup \mathscr{L})', T).$$

If $q \geqslant \langle \text{kill} \rangle$ *and* $\Gamma(E, T)$ *is not intersected by* $\mathscr{H}_q^0$ *then it is intersected by* $\mathscr{L}^0$ *and* q *is in* Y.

Lemma 14.7 follows immediately from Lemma 14.8. Indeed, the latter lemma implies that for all q, the set $D(\mathscr{I}, (q - \langle \text{kill} \rangle)P)$ is a local quarantine for C_q. Therefore for $q \geqslant \langle \text{kill} \rangle + Q$ the set $D(\mathscr{I}, PQ)$ is a local quarantine for C_q. It follows from (14.2) that $\langle \text{kill} \rangle + Q < 2Q < U$. □

Proof of Lemma 14.8. We build a new sequence of quarantines $\mathscr{I}_q$ just like in Section 14.3, satisfying Lemma 14.3. The recursive definition step is the same, but the starting quarantine $\mathscr{I}_{hU}$ is different. The set C_{hU} is the union of nine alliance clusters B_i for $i = 1, \ldots, 9$. Let $\mathscr{K}_i$ be the set of triangles with size $\langle \text{sick} \rangle rT$ such that $(\mathscr{I}^1 \cup \mathscr{K}_i)'$ is a quarantine for B_i. According to the assumption of the theorem, there is such a triangle set for all i. We define

$$\mathscr{K} = \left(\bigcup_i \mathscr{K}_i \right)', \qquad \mathscr{I}_{hU} = (\mathscr{I}^1 \cup \mathscr{K})'.$$

The latter set is obviously a quarantine for C_{hU} at time hU.

The process of creating $\mathscr{I}_q$ involves three kinds of steps. Sometimes we added a new triangle. Sometimes we deflated an old triangle. And sometimes we merged two triangles into the smallest one containing both. For each element of $\mathscr{I}_q$ we can define the set of their *ancestors*. New triangles are their own ancestors. Deflating a triangle does not change its ancestors. Merging two triangles unites their ancestry. Those triangles that have some ancestry in $\mathscr{I}$ are called *big*, the rest are called *small*.

LEMMA 14.9. *Each big triangle has exactly one big ancestor.*

Proof. We have to show that in the process of constructing $\mathscr{I}_q$, two big triangles will never be merged. Let H and I be two such triangles. They are disjoint, so according to Lemma 11.2, there is a j such that their j-separation $d_j(H, I)$ is positive. The construction begins by deflating each big triangle by the amount UT. This increases the j-separation by $2UT$. From that time on, the big triangles suffer only merging with small triangles and further deflation. Each deflation increases the separation. The merging decreases the separation at most by the size of the small triangle merged with the big one.

The small triangles come from elements of $\mathscr{K}$ or $\mathscr{L}_q$ for some q, using merging and deflation. Hence the small triangles are covered by $\mathscr{J}$ whose size is bounded by $10\langle\text{sick}\rangle rT$. The Size Conditions imply that this is smaller than $2UT$. □

Let $q > \langle\text{kill}\rangle$. We find just as in the proof of Lemma 14.4 that there is an s between hU and $\langle\text{kill}\rangle$ for which the small triangles disappear from $\mathscr{J}_s$. From this time on, the small triangles are covered by $\mathscr{L}$.

For each big triangle I in $\mathscr{J}$, let us denote by $\Lambda_q(I)$ the big triangle in $\mathscr{J}_q$ whose ancestor it is. Then we have

$$\Lambda_q(I) \subset D(I^1, P(q - \langle\text{kill}\rangle)). \tag{14.4}$$

Indeed, just as in the proof of Lemma 14.4, we find that the total extent of deflation of $\Lambda_q(I)$ is

$$\begin{aligned} &qP - 9rP - |\mathscr{J}| \\ &\quad \geqslant (q - \langle\text{kill}\rangle)P + \langle\text{kill}\rangle P - 10\langle\text{sick}\rangle rT - 9rP \\ &\quad \geqslant (q - \langle\text{kill}\rangle)P. \end{aligned}$$

Of course, we use Lemma 14.9 here.

This completes the proof of Lemma 14.8. Indeed, we showed that the big triangles and $\mathscr{L}$ together form a quarantine, and that the big triangles are deflated from $\mathscr{J}^1$ by at least $(q - \langle\text{kill}\rangle)P$. □

15. FORCING CODED OUTPUT

Section 14.5 indicated that the set of illegal colonies will shrink in the absence of noise. Thus, legality is restored automatically. Another important condition (used in the proof of Lemma 14.6) of successful computation in the presence of noise is that the input is close to a codeword of the error-correcting code Algeb. This condition will not be restored automatically, even if the colonies of the alliance are healthy, and even though the output of each fault-free computation is a codeword. Indeed, suppose that some input words have a larger number of deviations than the one correctable by Algeb. Then faults can change the input in such a way that the outputs of fault-free repetitions will be different from each other.

We keep only a small part of each repetition in *OutMem*. The result will therefore be a mixture of different codewords, which is in general not a codeword.

If the *InpMem* track contains words deviating only little from a codeword of Algeb then the result of the computation will be the same in all fault-free repetitions. In our program this implies that the result (even if it is otherwise wrong) will be close to a codeword.

The present section shows that a procedure can be added to the end of the program of the medium Univ that brings all words on the horizontal rows of the *InpMem* track of an alliance close to a codeword of Algeb, without significantly changing words that are already near codewords. This property will be important for the self-simulation in the next section.

For row $k = 0, \ldots, QP - 1$ of alliance E, let $u_k(q, E)$ denote the word on the *InpMem* track in the kth row of E at time qT. Let CodeDiff(u) denote the number of T-intervals in which the word u differs from $\text{Algeb}_*(\text{Algeb}^*(u))$. With an eye on the later application, we will only consider values $k > 2$, and will not try to change the top three rows of the alliance.

For the following lemma, two Size Conditions need some change with respect to Condition 13.3. Let

$$\langle\text{dev}\rangle' = 2\langle\text{corr}\rangle - \langle\text{dev}\rangle + 6\langle\text{sick}\rangle + 360.$$

We obtain $\langle\text{corr}\rangle'$ by replacing $\langle\text{dev}\rangle$ in its definition (13.2) with $\langle\text{dev}\rangle'$:

$$\langle\text{corr}\rangle' = \langle\text{dev}\rangle' + 27(\langle\text{sick}\rangle + 6).$$

The change in the Size Conditions is the following. In the first one, $\langle\text{corr}\rangle$ is replaced with $\langle\text{corr}\rangle'$. In the second one, QP/T is now squared.

CONDITION 15.1 (SIZE).

$$QP' > \max\left(3Q'|\text{Med}_0|, Q'|\text{Med}_0| + \frac{9\langle\text{corr}\rangle' rP'T}{P}\right),$$

$$UT' > \frac{U'}{Q'}\left(\frac{QP}{T}\right)^2 QT' \log(QP')\text{Step}_0,$$

$$P' > \max(\log U, \log T', |\mathsf{Prog}_0|, |\mathsf{modif}|),$$

$$T' > 9P',$$

$$P \mid T, \qquad \frac{T}{P} \Big| Q, \qquad Q \mid U.$$

THEOREM 15.1. *There is a procedure ForceCode, with the following property. Suppose that the definition of the medium* Univ *and the code φ differs from the one given in Section 14 only in that ForceCode is attached to the end, and* Algeb *must be correct now $\langle \mathrm{dev} \rangle' r$ errors. Assume that the conditions of Theorem 13.1 hold, with the modifications in the Size Conditions indicated above.*

Let C be an alliance cluster. Suppose that at time hUT, the set C_{hU} is locally healthy. Let f_0 denote the beginning of the procedure ForceCode. Then in each alliance E of C in each row $k > 2$ of E, we have

$$\mathsf{CodeDiff}(u_k((h+1)U, E)) < \langle \mathrm{dev} \rangle r.$$

If we had $\mathsf{CodeDiff}(u_k(f_0, E)) < \langle \mathrm{dev} \rangle r$ *for all $k > 2$ then we have*

$$\mathsf{Algeb}^*(u_k(f_0, E)) = \mathsf{Algeb}^*(u_k((h+1)U, E))$$

for all $k > 2$. Thus, if all input words are close to a codeword the input will not be changed by ForceCode.

The rest of this section is devoted to the proof of the above theorem. The procedure *ForceCode* will attempt to decide whether each row of *InpMem* differs only slightly from a codeword of Algeb. If this is not the case then each row will be changed to a standard fixed codeword w_0.

The procedure works within each alliance independently. We will therefore omit the notation of dependence on E from $u_k(q, E)$. Whenever the time qT refers obviously to the current time we also omit the dependence on q.

In all of the following lemmas we assume that the number of T-cubes belonging to the noise in the neighborhood of the cluster of E in the given time period is at most $9r$, i.e., that the noise is (U, r)-sparse.

15.1. Computing the Votes

The crucial part of the procedure *ForceCode* writes a "vote" $Vote[i, s] = 0$ or 1 in each cluster (i, s). The following properties will hold, even in the presence of (U, r)-sparse noise.

- There is a number b and a set $\mathcal{I}$ of triangles of size $O(rT)$ such that $Vote[i, s] = b$ for all clusters (i, s) outside $\mathcal{I}$.
- If, at time f_0, for some $k > 2$, we have $\mathsf{CodeDiff}(u_k) > (2\langle \mathrm{corr} \rangle - \langle \mathrm{dev} \rangle)r$ then $b = 0$.
- If, at time f_0, for all $k > 2$, we have $\mathsf{CodeDiff}(u_k) \leqslant \langle \mathrm{dev} \rangle r$ then $b = 1$.

The first part of *ForceCode* is an assessment of the situation. For the present section, let

$$n = PQ/T.$$

The

procedure *CheckCode*(i, s)

for $i, s = 0, \ldots, n-1$ does the following. After Q idling steps, it checks, simultaneously in all rows $k > 2$ of the alliance, whether

$$\mathsf{CodeDiff}(u_k) \leqslant \langle \mathrm{corr} \rangle r.$$

Then it computes the conjunction of these tests and stores the result in a bit $Vote[i, s]$ in the cluster (i, s) of the alliance. Thus, only the place where the result is stored depends on i and s. The first part of *ForceCode* now looks like this.

```
procedure FindVotes
begin
  for i = 0 to n − 1 do
    for s = 0 to n − 1 do
      CheckCode(i, s);
end
```

The n^2 repetitions are performed as a guard against noise. This is the most time-consuming part of the whole program of Univ.

We assumed that C_{hU} is locally healthy. Let G_1 be the union of all clusters F for which $\Gamma(F, T)$ intersects a triangle in $\mathscr{L}^0$. They include all clusters in which a singular event occurred during the original program after the first Q idling steps. (By that time q, as shown in Lemma 14.8, the big triangles of the quarantine $\mathscr{I}$ of the definition of local health disappear in C_q.) Let G_2 be the union of all clusters (i, s) for which a singular event occurred during the nonidling part of the computation of *CheckCode* (i, s). The following lemma is obtained by an analysis of the above iteration similar to the proof of Lemma 14.6.

LEMMA 15.1. (*a*) *Suppose that at the beginning of the program* $\mathsf{CodeDiff}(u_k) \leqslant \langle \mathrm{dev} \rangle r$ *holds for all* $k > 2$. *Then at the end of FindVotes, in each cluster* (i, s) *not belonging to* $G_1 \cup G_2$ *we have* $Vote[i, s] = 1$.

(*b*) *Suppose that at the beginning of the program we have* $\mathsf{CodeDiff}(u_k) > \langle \mathrm{dev} \rangle r + 2(\langle \mathrm{corr} \rangle - \langle \mathrm{dev} \rangle)r$ *for at least one* $k > 2$. *Then at the end of FindVotes, in each cluster* (i, s) *not belonging to* $G_1 \cup G_2$ *we have* $Vote[i, s] = 0$.

15.2. The Consensus Problem

The real technical difficulty of the present section is that of making a decision based on the contents of the array $Vote[i, s]$. We cannot fight faults made during the decision as we did until now, by mixing the results from different repetitions. In borderline cases, the result could then namely be the mixture of some codeword and w_0. This mixture may not be a codeword, and we want a codeword result in all cases. We can view the values of $Vote[i, s]$ as the different opinions of some population. We want to achieve consensus in this population, without reversing a near consensus. This problem was essentially solved in Section 12.6.

To apply those results, the sites of a small torus (used there to achieve consensus) will be simulated by the clusters of the alliance. Now, a square is not a torus but a torus can be "folded over" to a square. To be more formal, let us introduce the mapping $i \to \tilde{\imath}$ from $\mathbf{Z}_{2n}$ to $[0 \,..\, n)$ by

$$\tilde{\imath} = \min(i, 2n - i - 1).$$

All points of the interval $[0 \,..\, n)$ have two inverse images under this mapping. We define a mapping

$$(i, s) \to (\tilde{\imath}, \tilde{s})$$

of the torus $\mathbf{Z}_{2n}^2$ to a lattice square. This mapping maps points that are neighbors in the torus to points that are neighbors in the square. Let us temporarily denote by $(i_1, s_1), \ldots, (i_4, s_4)$ the four inverse images of (i, s).

We will use a new array $Vote'[i, s]$ over the torus $\mathbf{Z}_{2n}^2$. The four bits $Vote'[i_1, s_1], \ldots, Vote'[i_4, s_4]$ will be kept in the cluster $[T; i, s]^2$. Initially, the array $Vote$ will be quadrupled into the array $Vote'$, by repeating each bit of a cluster four times. Of course, since one cluster represents four sites of the imaginary torus $\mathbf{Z}_{2n}^2$, one fault will affect all four of these sites. Now the second part of the procedure *ForceCode* can be written as follows.

procedure *Consensus*
begin
  for $i, s = 0$ **to** $2n - 1$ **parallelly do** $Vote'[i, s] := Vote[\tilde{\imath}, \tilde{s}]$;
  for $i = 1$ **to** $2(\langle \mathrm{cons} \rangle + 1)n(r + 1)$ **do**
    $Vote' := R^+(Vote')$;
  for $i, s = 0$ **to** $n - 1$ **parallelly do**
    $Vote[i, s] := Vote'[i_1, s_1]$;
end

In the last step, we just chose arbitrarily the first one of the four votes in square $[T; i, s]^2$.

15.3. Analysis of the Consensus Algorithm

To avoid the awkward conversions between measurements, let us give a new size T to each cell of the torus $\mathbf{Z}_{2n}^2$, converting it into the real torus $[0, 2nT)^2 = [0, 2PQ)^2$ where the cells occupy the points whose coordinates are divisible by T (see Section 13.1). Without loss of generality, let us assume that the alliance E under consideration is $[PQ; 0, 0]^2$, i.e., its lower left corner is at the origin. The mapping $\tilde{a}$ now becomes $\min(a, 2PQ - a)$. In the torus $[0, 2PQ)^2$ we will also speak of T-squares. Each such T-square holds one vote.

In the reasoning below, we use both notions of triangles: the usual one, as well as the one used in connection with the modified Toom's Rule. Let us call the latter *special triangles*. The ordinary triangles are defined on the alliance cluster and its eight neighbors. The special triangles are defined on the torus $[0, 2PQ)^2$. Let us say that a special triangle *weakly covers* a cluster C if it covers the bottom of C.

LEMMA 15.2. *There is a number $b = 0$ or 1 such that at the end of the procedure Consensus, there is a set $\mathscr{M}$ of special triangles with*

$$|\mathscr{M}| \leqslant (6\langle\text{sick}\rangle + 360)rT$$

weakly covering each cluster $[T; i, s]^2$ of the torus with $Vote[i, s] \neq b$. In the cases treated by Lemma 15.1, the value of b will be what is stated there.

Proof. Let us first show that there is always a number b and a time t_0 during *Consensus* such that all votes outside $G_1 \cup G_2$ are equal to b. In the cases of Lemma 15.1 we can take t_0 to be the starting time of *Consensus*. In the rest of cases, let us locate a sequence of $2(\langle\text{cons}\rangle + 1)n$ noise-free iterations of the rule R^+. The first $2n$ iterations will be enough to eliminate the set $\mathscr{I}_q$, and thus make all clusters regular. By Theorem 11.1, the remaining $2\langle\text{cons}\rangle n$ noise-free iterations achieve consensus. Thus we can set t_0 equal to the time after these iterations.

The remainder of the proof will show that the consensus achieved by time t_0 will not be overturned. Assume $b = 0$. The case $b = 1$ can be treated similarly but somewhat simpler. We construct a set $\mathscr{M}$ of special triangles on the torus with the property that for all iterations of R^+ after t_0, they weakly cover every T-square in which the vote differs from 0.

For a triangle I, let us consider the four inverse images $I_1, \ldots, I_4$ of I under the above mapping $(i, s) \to (\tilde{\imath}, \tilde{s})$. Let $\bar{I}_k$ be the smallest special triangle containing I_k. For any set $\mathscr{I}$ of ordinary triangles let

$$\bar{\mathscr{I}} = \left(\bigcup_{I \in \mathscr{I}} \{\bar{I}_1, \ldots, \bar{I}_4\}\right)'.$$

Thus, we merge all special triangles obtained for all elements of $\mathscr{I}$ until we get a set $\bar{\mathscr{I}}$ of disjoint special triangles. We have $|\bar{I}_k| = 3/2|I|$, hence

$$|\bar{\mathscr{I}}| \leqslant 6|\mathscr{I}|.$$

Let us start from the set $\mathscr{L}$ of triangles. The number of triangles in $\mathscr{L}$ is at most $9r$. These have the property that for every cluster F that ever becomes singular during the computation considered, $\Gamma(F, T)$ intersects $\mathscr{L}$. The same will be true for $\bar{\mathscr{L}}$ and all inverse images of singular clusters. We have $\bar{\mathscr{L}} \leqslant 6|\mathscr{L}| \leqslant 6\langle\text{sick}\rangle rT$. Let us obtain the set $\mathscr{M}_1$ of special triangles by replacing each triangle $L(a, b, c)$ of $\bar{\mathscr{L}}$ with $L(a + 2T, b + 2T, c + 4T)$, and then merging. Then all inverse images of singular clusters are (completely) covered by $\mathscr{M}_1$. We have

$$\begin{aligned} |\mathscr{M}_1| &\leqslant |\bar{\mathscr{L}}| + 8T\#\bar{\mathscr{L}} \leqslant 6\langle\text{sick}\rangle rT + 8 \cdot 4 \cdot 9rT \\ &= (6\langle\text{sick}\rangle + 288)rT. \end{aligned}$$

If case (a) of Lemma 15.1 holds then at time t_0 set $G_1 \cup G_2$ can contain clusters (i, s) with $Vote[i, s] = 1$. We covered the inverse image of G_1 by $\mathscr{M}_1$. Let $\mathscr{M}_2$ be the set of special triangles obtained by covering the inverse images of the clusters of G_2 and merging. One cluster can be covered by a triangle of size $2T$. We have thus $|\mathscr{M}_2| \leqslant 4 \cdot 9r \cdot 2T = 72rT$. Let $\mathscr{M} = (\mathscr{M}_1 \cup \mathscr{M}_2)'$. The above estimates give $|\mathscr{M}| \leqslant (6\langle\text{sick}\rangle + 360)rT$.

Let us first show that if before an application of R', the set of clusters (i, s) on the torus with $Vote'[i, s] = 1$ is weakly covered by $\mathscr{M}$ then this is the case after the application, too. Suppose the cluster (i, s) (the T-square $[T; i, s]^2$) is not weakly covered by $\mathscr{M}$. Then it is regular. It is not possible that both clusters $[T; i, s + 1]^2$ and $[T; i + 1, s]^2$ are weakly covered by $\mathscr{M}$. Indeed, $\mathscr{M}$ consists of disjoint triangles, hence these two clusters would be weakly covered by the same triangle, which would then also cover the cluster $[T; i, s]^2$. Suppose, e.g., that $[T; i, s + 1]^2$ is not covered. Then the neighbor clusters $[T; i, s]^2$ and $[T; i, s + 1]^2$ are regular and contain a 0 vote throughout the computation of R'. Therefore the computation of R' will not change $Vote'[i, s]$.

Suppose that after an *Inflate*, at time $(q + 1)T$, a cluster $[T; i, s]^2$ gets vote 1. We will prove that it is weakly covered by $\mathscr{M}$. Suppose

it is not. Then it is regular. Suppose that one of the clusters $[T; i-1, s]^2$, $[T; i, s-1]^2$ is singular. Then it is completely (not only weakly) covered by a triangle of $\mathcal{M}$. It follows that $[T; i, s]^2$ is weakly covered by the same triangle. Suppose now that all three of these clusters are regular.

If *Inflate* brought $Vote'[i, s] = 1$ by time $(q+1)T$ then at time qT, we had either $Vote'[i-1, s] = 1$ or $Vote'[i, s-1] = 1$. Without loss of generality, let us suppose that we had $Vote'[i-1, s] = 1$. Since this is a regular cluster, this value of $Vote'$ at time qT is obtained by R' from the votes in clusters $[T; i-1, s]^2$, $[T; i, s]^2$, $[T; i-1, s+1]^2$ at time $(q-1)T$. Therefore two of these three clusters must be weakly covered by $\mathcal{M}$. It follows by simple geometric inspection that $[T; i, s]^2$ is also weakly covered. □

15.4. The New Codewords

For a word u and a number s in $[0..n)$ let us denote by $u[T; s]$ the part of u on the T-interval $[T; s]$. The next part

procedure *Enforce*

of *ForceCode* works simultaneously in all clusters of the alliance E. In cluster $[T; i, s]^2$, if $Vote[i, s] = 0$ then for all rows $k > 2$ in $[iT..(i+1)T)$, it changes $u_k[T; s]$ to $w_0[T; s]$.

LEMMA 15.3. *Let q denote the time at the end of the procedure Enforce. Under the conditions of Theorem 15.1, we have* $\mathsf{CodeDiff}[u_k(q, E)] \leqslant \langle \mathrm{dev} \rangle' r$ *for all $k > 2$. In case (a) of Lemma 15.1, we also have*

$$\mathsf{Algeb}^*(u_k(f_0, E)) = \mathsf{Algeb}^*(u_k(q, E)),$$

for all $k > 2$, i.e., the original codewords are not changed.

Proof. Suppose $\mathsf{CodeDiff}(u_k(hU, E))$ is at most $\langle \mathrm{dev} \rangle r$ for all $k \geqslant 2$. Then, by Lemma 15.1, we have the case $b = 1$ in Lemma 15.2. Due to the last lemma the clusters where the codeword is changed will be weakly covered by $\mathcal{M}$. Therefore the number of such clusters is bounded in each row by $|\mathcal{M}|/T$. This is the bound on the increase in the number of deviations from a codeword.

Suppose now that $\mathsf{CodeDiff}(u_k(hU, E))$ is greater than $\langle \mathrm{dev} \rangle r + 2(\langle \mathrm{corr} \rangle - \langle \mathrm{dev} \rangle)r$ for some $k > 2$. Then by Lemma 15.1 we have the case $b = 0$ in Lemma 15.2, and we get $u_k = w_0$ in all clusters except for the ones weakly covered by $\mathcal{M}$. Therefore we get $\mathsf{CodeDiff}(u_k) \leqslant |\mathcal{M}|/T$.

Suppose finally that $\mathsf{CodeDiff}[u_k(hU, E)]$ is at most $\langle \mathrm{dev} \rangle r + 2(\langle \mathrm{corr} \rangle - \langle \mathrm{dev} \rangle)r$ for all $k > 2$. Lemma 15.2 still holds but we do not know now the value of b. If $b = 0$ then the $\mathsf{CodeDiff}(u_k)$ will again be bounded by $|\mathcal{M}|/T$ for all $k > 2$, while if $b = 1$ then the number of deviations will *increase* by $|\mathcal{M}|/T$, bringing it to at most

$$\begin{aligned} &\langle \mathrm{dev} \rangle r + 2(\langle \mathrm{corr} \rangle - \langle \mathrm{dev} \rangle)r + |\mathcal{M}|/T \\ &\quad = (2\langle \mathrm{corr} \rangle - \langle \mathrm{dev} \rangle + 6\langle \mathrm{sick} \rangle + 360)r \\ &\quad = \langle \mathrm{dev} \rangle' r. \end{aligned}$$

□

The final part of the procedure *ForceCode* is a procedure that will decrease $\mathsf{CodeDiff}(u_k)$ for each $k > 2$ from $\langle \mathrm{dev} \rangle' r$ back to $\langle \mathrm{dev} \rangle r$. It will simply try to decrease the difference to 0. To fight faults it is repeated for each output column of width T just as *CompColumn*(s, t).

```
procedure Refresh
begin
  for s = 0 to n − 1 do
    for k := 0 to PQ − 1 parallelly do
      u_k[T; s] := (Algeb_*(Algeb*(u_k)))[T; s]
end
```

End of proof of Theorem 15.1. The proof of Lemma 14.6 applied to the procedure *Refresh*, together with the fact that the code Algeb corrects $\langle \mathrm{dev} \rangle' r$ bursts of errors, gives the desired conclusion. □

Let us summarize the procedure *ForceCode*.

```
procedure ForceCode
begin
  FindVotes;
  Consensus;
  Enforce;
  Refresh;
end.
```

16. AMPLIFIERS

Theorem 13.1 and Lemma 14.8 look similar to the Computation Condition and Restoration Condition for alliances instead of colonies. This section extends the similarity to equivalence. We will show that by an appropriate choice of the code φ, a new self-correcting evolution can be defined by decoding. With the tools of the present section, we learn how to turn an implementation into a better one, as desired in Section 13.7.

In the previous section, the medium Med_0 was chosen arbitrarily. Let us choose it now to be the medium Univ. There is no circularity in this choice: the medium Univ was constructed to work for *all* media Med_0, including Univ itself. Let us refer to the spaces in which the simulating and simulated iterative arrays of the medium Univ operate by $\mathbf{W}^*$ and $\mathbf{W}$, respectively:

$$\mathbf{W}^* = \mathbf{Z}^2_{nQ'}, \qquad \mathbf{W} = \mathbf{Z}^2_{nPQ}.$$

Let us assume that the parameters $P', \ldots, r$ satisfy the Size Condition 13.3. Let φ be a code satisfying the conditions of Theorem 13.1.

The decoding φ^* orders an evolution x^* of $\mathsf{Med}_0 = \mathsf{Univ}$ in $\mathbf{W}^*$ to the evolution x of Univ in $\mathbf{W}$. Let us define

$$y = \varphi_*(x^*). \tag{16.1}$$

Note that the evolution y is now not necessarily a trajectory. The target blocks of the code φ are the PQ-squares called alliances. The source blocks are Q'-squares, which we will call *big colonies*. We define

$$T^* = TU, \qquad P^* = PQ, \qquad \alpha^* = TU/U', \qquad \beta^* = PQ/Q'$$

Then we have $P^* \mid T^*$. We assign nonunit cell size α^* and cell worktime β^* to the simulated space. With this definition, the size and work time of the big colony is equal to the size and work time of the alliance simulating it. The total sizes of the spaces $\mathbf{W}$ and $\mathbf{W}^*$ become identical. Both spaces are subdivided into the same number of squares of size $P^* = PQ$. However, the size of the individual cells is generally larger in $\mathbf{W}^*$ than in $\mathbf{W}$.

Under the simulation φ, the evolution x^* on the big colony $E = [P^*; i, j]^2$ during time interval $[T^*; h]$ corresponds to the evolution

x on alliance $E_* = [QP; i, j]^2$ in the time interval $[UT; h]$. Just as T-squares were called clusters, the T^*-squares will be called *big clusters*. The above correspondence will map each big cluster A into an alliance cluster A_*. Therefore each T^*-cube $B = [T^*; h] \times A$ is mapped into a UT-cube $B_* = [UT; h] \times A_*$.

16.1. Noise, Health, Damage

We will say that the T^*-cube B belongs to the noise $\mathcal{N}^*$ if the noise $\mathcal{N}$ is *not sparse* on the corresponding UT-cube B_*. Notice that the noise $\mathcal{N}^*$ is defined in terms of the noise $\mathcal{N}$ and the sizes T, U, r, independently of all other primitive notions.

Let us choose a Q^* and $U^* > Q^*$ such that $\log U^* < Q'$. Now that we have P^*, T^*, Q^*, U^*, the notion of health and legality for big colonies will be defined just as it was for small colonies. The set $\mathsf{Damage}^*(0)$ is the union of all big clusters C for which *not* all of following conditions hold at time 0:

- The alliance cluster C_* is locally healthy.
- In each row of each alliance of C_*, the deviations of the *InpMem* track of evolution x from the evolution y are covered by $\langle \mathrm{dev} \rangle r$ T-intervals.
- All big colonies in C are legal.

The first two conditions concerns the simulating evolution x. They are identical, for $h = 0$, with the statements in Lemma 14.1. The second condition can also be stated, using the terminology of Section 15 as $\mathsf{CodeDiff}(u_k(0, C_*)) \leqslant \langle \mathrm{dev} \rangle r$ for all rows k of C_*. The last condition concerns the simulated evolution x^*.

16.2. Simulating a Self-Correcting Evolution

THEOREM 16.1. *There is a universal medium* Univ *supporting colonies with the following properties. Assume for parameters* $P', \ldots, r$, *the Size Condition 15.1.*

- *If* $\mathsf{modif} = 0$ *then* Univ *satisfies Theorem 13.1.*
- *If* $\mathsf{modif} = 1$ *then there is a standard simulation* φ *of* Univ *by itself such that for all self-correcting evolutions* $(x, \mathcal{N}, \mathsf{Damage}(0))$ *the triple* $(x^*, \mathcal{N}^*, \mathsf{Damage}^*(0))$ *is a self-correcting evolution.*

This theorem forms the backbone of the proof of the main result, Theorem 7.1. Its case with modif $= 0$ is simply Theorem 13.1. The case with modif $= 1$ gives the desired *amplifier* simulation that turns an implementation into a better one, with smaller probability bound. It implies namely the following theorem.

THEOREM 16.2. *Suppose that the parameters $P', \dots, r$ satisfy the conditions of Theorem 16.1 and the medium* Univ *satisfies its conclusion. Suppose that we have an implementation Ψ with the parameters P', T', P, T, ϱ, p, satisfying* (9.1). *Then the construction*

$$\Phi\colon (x, \mathcal{N}, \mathsf{Damage}(0)) \to (x^*, \mathcal{N}^*, \mathsf{Damage}^*(0))$$

gives a new implementation $\Psi \circ \Phi$ with the parameters $Q', U', P^*, T^*, \varrho, p^r$.

The rest of the section is devoted to the proof of Theorem 16.1, i.e., to the changes to the program of Univ to achieve the desired effects for modif $= 1$.

First we prove an auxiliary statement.

LEMMA 16.1. *Suppose that the medium* Univ *supports colonies and the parameters $P', \dots, r$ satisfy the Size Conditions 15.1. Let $(x, \mathcal{N}, \mathsf{Damage}(0))$ be a self-correcting evolution. Let C be a big cluster in W^*, and h a natural number. Then any quarantine at time hT^* for C with respect to the triple $(x^*, \mathcal{N}^*, \mathsf{Damage}^*(0))$ is a local quarantine for C_* with respect to the triple $(x, \mathcal{N}, \mathsf{Damage}(0))$.*

Proof. This statement can be proven by induction on h. It is true for $h = 0$ by the definition of Damage*(0). If it is assumed true for h then it follows immediately for $h + 1$ using the inductive definition of $C \cap \mathsf{Damage}((q + 1)T)$ given in Section 13.4, the definition of $\mathcal{N}^*$ and Lemma 14.7. □

To make the triple $(x^*, \mathcal{N}^*, \mathsf{Damage}^*(0))$ a self-correcting evolution we have to make it satisfy the Restoration Condition and the Computation Condition.

The Restoration Condition says now that at all times hT^*, all big colonies disjoint from Damage*(hT) are legal. We will prove the following lemma.

LEMMA 16.2. *There is a universal medium* Univ *supporting colonies with the following property. Assume for parameters $P', \dots, r$, the Size Condition 13.3. Let h be a natural number. Assume that at time hT^*, for all big colonies E not intersecting with* Damage*(hT), *the following conditions are satisfied.*

(*a*) *In each row of the alliance E_*, the deviations of the InpMem track of evolution x from the evolution y [as defined in* (16.1)] *are covered by $\langle \mathrm{dev} \rangle r$ T-intervals.*

(*b*) *E is legal.*

Then the same is true at time $(h + 1)T^*$ for all big colonies not intersecting with Damage*$((h + 1)T^*)$.

For $h = 0$, the assertions $(a - b)$ in the lemma will be satisfied, due to the definition of Damage*(0) in 16.1 above. Therefore this lemma will imply that these conditions are satisfied for all h, i.e., the Restoration Condition holds.

Let C be a big cluster in W^* containing the big colony E. If the noise $\mathcal{N}^*$ is not empty during the interval $[T^*; h]$ in $C_{hU} = \Gamma(C, T^*)$ then C belongs to Damage*$((h + 1)T^*)$, and the conclusion of Lemma 16.2 is automatically satisfied. Suppose now that the noise $\mathcal{N}^*$ is empty in this region and $\mathcal{I}$ is a minimal quarantine for C at time hT^*. Then by the definition of Damage(qT) in Section 13.4, we have

$$C \cap \mathsf{Damage}^*((h + 1)T^*) = C \cap \bigcup D(\mathcal{I}, P^* + T^*).$$

We must design Univ therefore in such a way that if E does not intersect $D(\mathcal{I}, P^* + T^*)$ then properties $(a - b)$ are satisfied in E at time $(h + 1)T^*$.

16.3. Initialization

Our original definition of the medium Univ will not guarantee that the big colony E becomes legal if it was not. Indeed, even in the absence of noise, the small colonies just simulate the evolution of whatever they find in the big colony and its neighbor. The simulated medium is now Univ again. Though we made Univ to support colonies, this property is usable only for a big colony that is already healthy.

To make the simulated big colony healthy "against its own will," a simple but crucial new procedure, called *Init*, will be inserted between the procedures *Input* and *Compute*, at the beginning of each big work period. After the procedure *Input*, the configurations of these colonies are in nine subsquares of size $QP/3$ of the alliance, on the Simulator track. Let us agree that whenever we speak about a big colony we mean one of these nine subsquares of the Simulator track of our alliance.

The procedure *Init* puts the symbol *topleft* into the left-most column and the top row of the simulated big colony and checks that all other cells of this colony are in a state belonging to S_{init}. If a cell is in some other state then its state is changed arbitrarily to one in S_{int}. Finally, *Init* writes the integers Q', U' in place of the first two parameters in the parameter field. (They play the role of P', T' for the big colony).

Now the new procedure *CompColumn* has the following form.

procedure *CompColumn*(s, t, l);
begin
  Decode;
  Input;
  if modif $=$ 1 **and** $l = 1$ **then** *Init*;
  Compute(t);
  Encode;
  Output(s);
end;

It is crucial that in the procedures *ForceCode* and *Init*, the small colonies intervene in the simulation of big colonies. The code φ thus defined is therefore no more a pure simulation, since health is *forced* on the big colonies. Since, however, φ is not a simulation, we do not want to use this program always, and the parameter modif allows us to make the choice.

Sections 15 and 16.3 engineered the "magic" recovery needed for the big colonies to recover from loss of structure. The rest of the present section proves that the magic works.

16.4. Legalization Works

Let E be a big colony disjoint from $D(\mathscr{I}, P^* + T^*)$. We have to show at time $(h+1)T^*$, conditions $(a-b)$ of Lemma 16.2 hold. The following geometrical lemma is easy to verify.

LEMMA 16.3. *Let $\mathscr{H}$ be a set of disjoint triangles. Let E be a P^*-square disjoint from $D(\mathscr{H}, P^*)$. Let the P^*-squares E_1, E_2, E_3 be the northern, southeastern, and southwestern neighbors of E. At least two of these three squares are disjoint from $\mathscr{H}$.*

We apply this lemma to $\mathscr{H} = \mathscr{I}^1$ as defined above and the big colony E that we assumed at the beginning of the previous paragraph to be disjoint from $D(\mathscr{I}^1, P^*)$. It follows that two of the three neighbors E_i are disjoint from $\mathscr{I}_1$. Without loss of generality, we can assume that E_1 and E_2 are disjoint from $\mathscr{I}^1$.

We assumed that $\mathscr{I}$ is a quarantine at time hT^* for the evolution x^*. Hence by Lemma 16.1, $\mathscr{I}$ is a local quarantine at that time, and E_1, E_2 are legal big colonies at the same time.

We will now follow the reasoning of Section 14.4. Before time $(h+1)U$, only a singular event can create new deviations on the *InpMem* track. According to Lemma 14.8, if cluster F is the site of a singular event then its neighborhood $\Gamma(F, T)$ intersects

$$D((\mathscr{I}^1 \cup \mathscr{K} \cup \mathscr{L})', T).$$

The alliances E_{1*}, E_{2*} are disjoint from $\mathscr{I}^1$ at time hUT. Just as in the proof of Lemma 14.6, we can conclude the following lemma.

LEMMA 16.4. *During the time interval $[hUT..(h+1)UT)$, in each row of E_{1*} and E_{2*}, the number of T-intervals with deviations on the InpMem track is bounded by $\langle \text{corr} \rangle r$.*

The following lemma says that after time $(hU + 2Q)T$, in the alliances E_*, E_{1*}, and E_{2*}, the singular events are restricted as in the proof of Lemma 14.6.

LEMMA 16.5. *If cluster F in $E_* \cup E_{1*} \cup E_{2*}$ is singular at time qT for q in $[(hU + 2Q)T..(h+1)U)$ then $\Gamma(F, T)$ intersects $\mathscr{L}^0$ and q is in Y.*

Proof. Remember $P^* = PQ$. According to Lemma 14.8, the neighborhood $\Gamma(F, T)$ is intersected by $\mathscr{H}_q^0 \cup \mathscr{L}^0$. Here, $\mathscr{H}_q = D(\mathscr{I}^1, (q - \langle \text{kill} \rangle)P)$. Let us show that for $q > hU + 2Q$ the set $\mathscr{H}_q^0$ is disjoint from $\Gamma(F, T)$. This will imply that the latter intersects $\mathscr{L}^0$. Lemma 14.8 says that in this case, q is in Y.

We have

$$(q - \langle \text{kill} \rangle)P > (2Q - \langle \text{kill} \rangle)P$$
$$= PQ + (Q - \langle \text{kill} \rangle)P \geqslant PQ + 2T.$$

Here we used $(Q - \langle \text{kill} \rangle)P > 2T$, which follows from the Size Conditions. It follows that $H_q^0 \subset D(\mathscr{I}^1, PQ + 3T)$. We assumed that $D(\mathscr{I}^1, PQ)$ is disjoint from E_*, E_{1*}, and E_{2*}, hence it is disjoint from F. Therefore by (11.2), the neighborhood $\Gamma(F, T)$ is disjoint from $D(\mathscr{I}^1, PQ + 3T)$. □

Proof of Lemma 16.2. The program begins with $2Q$ idling steps. After these, according to Lemma 16.5, the singular events are restricted in E_*, E_{1*}, and E_{2*} just as stated by Lemma 14.2.

We assumed $\mathsf{modif} = 1$. Hence for $l = 1$, in all regular calls s of the procedure *CompColumn*(s, t, l), the procedure *Init* makes the simulated colony healthy. The simulated medium Univ, by assumption, supports colonies. Therefore since the big colonies E_1, E_2 were legal, by the end of the procedure *Compute*(t), the result represents a legal colony.

The code φ is such that the top three rows of the alliance code the top three rows of the corresponding big colony. Therefore the top three rows of the alliance deviate in at most $\langle \text{dev} \rangle r$ T-intervals from the code of the top three-rows of a legal big colony.

Finally, the procedure *ForceCode* decreases $\mathsf{CodeDiff}(u_k)$ to $\langle \text{dev} \rangle r$, for all $k > 2$. □

End of the proof of Theorem 16.1. To make the evolution $(x^*, \mathscr{N}^*, \mathsf{Damage}^*(0))$ self-correcting, it remains to prove the Computation Condition. This condition says now the following: Let the big cluster C be regular at time hT^*. Then the configuration $x^*[(h+1)T^*, C]$ is what it would be if x^* was a trajectory of Univ starting from the same configuration $x^*[hT^*, C_{hU}]$.

The regularity of the big cluster C at time hT^* means a condition on the noise, and one for the damage. For the noise, it says that the noise $\mathscr{N}$ is sparse over the set $[UT; h] \times C_{hU}$ in W. For the damage, it says that $\mathsf{Damage}^*(hT^*)$ does not intersect C_{hU}. By the definition of $\mathsf{Damage}^*(0)$ and Lemma 16.2, this implies that each row of each alliance of C_{hU} differs from a codeword by at most $\langle \text{dev} \rangle r$ T-intervals.

Now Lemma 14.2 has essentially these conditions and the needed conclusions. □

17. CONSTANT SPACE REDUNDANCY

In the present section, we sharpen Theorem 13.1, by weakening one of the Size Conditions 13.3. This condition has the form

$$QP' > \max(3Q'|\mathsf{Med}_0|, Q'|\mathsf{Med}_0| + 9\langle \text{corr} \rangle rP'T/P).$$

We want to eliminate the factor $3|\mathsf{Med}_0|$. This economy is necessary only when our goal is constant space redundancy, as promised in the main theorem. In that case, the factor $3|\mathsf{Med}_0|$ cannot be tolerated, since the final simulation is a concatenation of several simulations of this kind, multiplying the cell size (which is the measure of the space redundancy) every time by $3|\mathsf{Med}_0|$.

As long as we design the simulation it is convenient to use unit cell sizes and cell worktimes in the simulated colonies $[Q'; i, j]^2$. The new cell size α^* and cell worktime β^* can then be computed at the end, as done in Section 16. These parameters do not enter the Size Conditions anyway.

17.1. Economical Trajectories

If we want to get rid of the factor $|\mathsf{Med}_0|$ then most cells of the Q'-square should not carry much information so that an encoding into binary strings of length $|\mathsf{Med}_0|$ is unnecessary. Therefore the largest part of the Q'-square will be destined to the passive storage of information, i.e., it will be "memory." We add more ingredients to our model. We assume that there is a four-element subset $S_{\mathsf{Med}_0, \text{mem}}$ of the states of the medium Med_0, called the set of *memory states* that is *invariant* with respect to the rule Med_0, i.e., the rule Med_0 never changes a memory state into a nonmemory state.

We also assume that a parameter $\langle \text{wksp} \rangle' \geqslant 1$ is given. A Q'-square of Med_0 will be called *economical* if its cells in all but the bottom $Q'/\langle \text{wksp} \rangle'$ rows have memory states. These bottom rows will be called the Workspace field of the Q'-square, the rest the Memory field. Notice that the notion of an economical square depends on the parameter $\langle \text{wksp} \rangle'$ and the set $S_{\mathsf{Med}_0, \text{mem}}$.

In Theorem 13.1, we simulated all possible trajectories of Med_0. Now we confine ourselves to trajectories that are *economical*, i.e., such in which all colonies are economical. Due to the invariance of the set of memory states, if a trajectory starts from a configuration of economical colonies, then it is economical.

Though we make our task easier by confining ourselves to the simulation of economical trajectories, we pay for this by making the medium Univ and its simulating evolutions also economical. The set S_{mem} of memory states of Univ will be part of the set S_{init} of initial states introduced in Section 13.3. There will also be a parameter $\langle \text{wksp} \rangle$ for the definition of economical small colonies. The small colonies will be required to be both legal and economical.

The set of parameters of our model is now

$$P', T', Q, U, Q', U', P, Q, \langle \text{wksp} \rangle, \langle \text{wksp} \rangle', \mathsf{Prog}_0, \mathsf{modif}, r.$$

We replace the Size Conditions 15.1 with the following.

CONDITION 17.1 (SIZE).

$$QP'\left(1 - \frac{1}{\langle \text{wksp} \rangle}\right) > Q'\left(1 + \frac{|\mathsf{Med}_0|}{\langle \text{wksp} \rangle'}\right) + 9\langle \text{corr} \rangle' r P' T / P,$$

$$P' > \max(\log U, \log T', |\mathsf{Prog}_0|),$$

$$UT' > (\langle \text{wksp} \rangle |\mathsf{Med}_0|)^2 \times \frac{U'}{Q'}\left(\frac{QP}{T}\right)^2 QT' \log(QP')\mathsf{Step}_0,$$

$$T' \geqslant 9P',$$

$$P \mid T, \qquad \frac{T}{P} \,\Big|\, Q, \qquad Q \mid U.$$

Only the bounds on QP' and UT' have changed. The bound on QP' is smaller: it will be asymptotically equal to Q'. On the other hand, we pay with a factor $(\langle \text{wksp} \rangle |\mathsf{Med}_0|)^2$ in the bound on the time UT', for the fact that only a fraction $1/\langle \text{wksp} \rangle$ of each colony is devoted to workspace.

Now we can formulate the optimized version of Theorem 13.1.

THEOREM 17.1. *There is an economical medium* Univ *supporting colonies such that the following holds. Assume the Size Condition 17.1 for some economical* Prog_0, *and numbers* $P', \ldots, r$. *Then there is a standard simulation* φ *of* Med_0 *by* Univ *mapping economical trajectories into economical trajectories, such that the conclusion of Theorem 13.1 holds.*

The rest of the present section is devoted to the proof of the above theorem. It describes all necessary modifications to the medium Univ and the code φ. If the reader is not interested in this optimization then this section can be skipped.

17.2. Drying

Ordinary self-simulation would proceed by expanding the content of each cell to be simulated, as done at the beginning of Section 13 and storing the result in the Memory of the simulating colony. It is possible to avoid expansion by giving different treatment to the Memory and the Workspace. Due to the small number of memory states, only the Workspace must be expanded. The operation can be viewed as *drying* a flower by pressing it between the leaves of a book. The process will not change the parts that are already dry.

Formally, one can define a code

$$\mathsf{Dry} = (\mathsf{Dry}_*, \mathsf{Dry}^*).$$

It is applied to a Q'-square and encodes it into an array of memory cells. The code *leaves the Memory unchanged* and expands each symbol of the Workspace into a column $\mathsf{bin}(s)$ of height $|\mathsf{Med}_0|$. Thus, each row of the Workspace is expanded into $|\mathsf{Med}_0|$ rows. The vertical size of the Q'-square increases therefore to

$$Q'(1 - 1/\langle \text{wksp} \rangle') + |\mathsf{Med}_0| Q' / \langle \text{wksp} \rangle'.$$

The crucial difference between drying and the error-correcting code Algeb is that since drying is symbol-for-symbol, the simulating alliance can manipulate the dry information *without ever reconstituting it*. The parameters of the alliance contain $\langle \text{wksp} \rangle'$, hence the small colonies of the alliance will know which simulated cells are expanded and which ones are not.

17.3. The Code φ

The Memory field of a Univ-colony is divided into a vertical strip of width $P/\langle \text{wksp} \rangle$ on the right side, called the Channel field, and the rest, called the Data field. The Channel field is needed only for communication between the Workspaces of the colony and its northern neighbor. Only the Data field, which still occupies most of the colony, will be used for actual storage.

The union of the Memory fields of the member small colonies will be called the *Memory field of the alliance*. The Data, Workspace, and Channel fields are defined similarly. Thus, the Workspace and Channel fields of the alliance form a grid: a union of horizontal and vertical strips of width $P/\langle\text{wksp}\rangle$. The Data field is the complement: it consists of Q^2 squares of size $P(1 - 1/\langle\text{wksp}\rangle)$.

Let us define the code φ. The encoding φ_* is defined as follows. First we apply the code $\mathsf{Algeb}_* \circ \mathsf{Dry}_*$ to the Q'-square. Then we distribute the result in the Data fields of the small colonies of the alliance. Finally, for all i, j, we make each of the small colonies legal, initialized to the beginning of a work period of the alliance.

The decoding φ^* is defined as follows. For each row of the alliance intersecting with the Data, we take the information from the Data parts of the row. We apply the decoding $\mathsf{Dry}^* \circ \mathsf{Algeb}^*$ to the rectangle obtained from all rows this way. This means reconstituting the Workspace rows of the Q'-square from their expanded form.

17.4. Memory in the Universal Medium

How will we retrieve information from the Memory field of Univ-colonies? For $n = 0, 1, 2, 3$, if a cell has the memory state s_n then we say that this cell stores the pair of bits n_0, n_1 in n. Memory cells with shift their left bits up and their right bits down between each other. Thus, if x is a trajectory of Univ and

$$x[t, i, j] = s_{m_1 m_0}, \qquad x[t, i+1, j] = s_{n_1 n_0}$$

for bits m_0, m_1, n_0, n_1 then we will have

$$x[t+1, i, j] = s_{n_1 u}, \qquad x[t+1, i+1, j] = s_{v m_1}$$

for some bits u, v. If the upper neighbor is not a memory cell then the cell reads some fixed function of the state of the upper neighbor into its right bit: the effect of this is that a workspace cell can "write" into a memory cell. (Of course, it can also read.)

During almost the whole computation, the workspace cells at the upper and lower edges of the Memory have states that keep the information rotating in each column: flowing down in the right bits and turned back at the lower edge to flow back up in the left bits. An appropriate times, the workspace cells can write something into the Memory or read from it.

17.5. Ranks and Slots

We must give up the luxury of separate *InpMem* and *OutMem* tracks in Univ. Moreover, since the Workspace is only a small part of the whole colony, we cannot store all nine coded neighbor big colonies in the alliance for simulation. We solve these problems in the present subsection by *trading time for space*.

Let

$$R = \left\lfloor \frac{QP'}{7|\mathsf{Med}_0|\langle\text{wksp}\rangle} \right\rfloor.$$

Let us divide the Q'-square into horizontal strips called *ranks*. The width of each rank is at least R but smaller than $2R$. Each rank is either completely in the Memory or completely in the Workspace. These are the only requirements for ranks, and it is easy to find a subdivision with these properties. With the Memory, e.g., we can begin to divide it into strips of width R, leaving at the bottom a somewhat wider last strip.

The computation will update the content of the simulated Q'-square rank-by-rank. An *extended rank* is a rank extended by its left and right neighbor ranks from the corresponding neighbor colonies. After drying, a rank of width n will be represented by n or $|\mathsf{Med}_0|n$ rows depending on whether it comes from the Memory or the Workspace. The number R is determined in such a way that even when they have $2R|\mathsf{Med}_0|$ rows after drying, three neighboring ranks will fit comfortably into the Workspace of the alliance. To fit there three neighboring extended ranks, we just store three symbols per cell of the simulating medium Univ. (Each symbol consists of at most two bits.)

Let

$$N = \lfloor U'/R \rfloor.$$

The computation will proceed in *stages* $l = 1, \ldots, N+1$. Each but the last stage computes the state of the Q'-square after R more time units. Some care must be taken not to update information that will be needed in the next stage of the serial simulation. One solution, taken from Toffoli's Cellular Automaton Machine (see [Tof]), is to distinguish between the physical and logical positions of a rank. The physical position of a rank will be called its *slot*. It will vary

from stage to stage. Each slot is a strip consisting of a sufficient number of rows in the simulating alliance, to store a certain rank. The number

$$m = \lfloor QP'(1 - 1/\langle \text{wksp} \rangle) \rfloor$$

of rows in the Memory of the simulating alliance is somewhat more than needed to accommodate all ranks. Indeed, the number of all rows in the dried Q'-square is less than $Q'(1 + |\text{Med}_0|/\langle \text{wksp} \rangle')$, which, according to the Size Conditions, is still less than m.

We will use the m available rows for slots as a circular store. At the beginning, $slot(i, 0)$ holds the initial content of rank i. These slots are consecutive strips starting from the top row of the alliance. Each stage l of the simulation consists of *substages* i, corresponding to the ranks of the alliance. In stage l, at the beginning of substage i, the new content of rank j for $j \leqslant i - 1$ is found in $slot(j, l)$. The old content of ranks $j \geqslant i - 1$ is found in the slots $slot(j, l - 1)$. In the cycle of available rows mod m, the slots $slot(j, l - 1)$ for $j \geqslant i - 1$ are followed by $slot(j, l)$ for $j \leqslant i - 1$. The union of all these slots forms a segment mod m.

In substage i, we load the old content of the extended ranks $i - 1$, i, and $i + 1$ into the Workspace of the alliance from the corresponding slots. The simulation is performed (with the repetitions s as earlier), and the new value of rank i is stored in a new slot $slot(i, l)$ at the end of the segment. At the same time, $slot(i - 1, l - 1)$ is released, since it is no longer needed.

17.6. Procedures

The above outline results in the following procedures. Let $\langle \text{ranks} \rangle$ denote the number of ranks.

procedure *Input*(i, l)

for $i = 2, \ldots, \langle \text{ranks} \rangle - 1$ and $l = 1, \ldots, N + 1$ loads the Data of the extended slots

$$slot(i - 1, l - 1),\ slot(i, l - 1),\ slot(i + 1, l - 1)$$

(representing the corresponding extended ranks) and spreads it in the Workspace of the alliance. For $i = 1, \langle \text{ranks} \rangle$, the information

of one of the three ranks must be taken from the appropriate slot of the upper resp. lower neighbor alliance. For extended rank i, for a row n in one of the above three slots, its destination is a row

$$Map(i, n)$$

in the destination alliance. The function $Map(i, n)$, as well as the boundaries of all slots in all stages, could be described by explicit formulas. The Workspace occupies only a portion $1/\langle \text{wksp} \rangle$ of each small colony. Therefore at its destination given by the function $Map(i, n)$, the information of a rank will occupy $\langle \text{wksp} \rangle$ times more colony rows than in its slot.

procedure *Output*(i, s, l)

takes the information in rows $Map(i, n)$ of the colonies in column s of clusters and writes it back to the rows n of column s of clusters in $slot(i, l)$.

After stage $l = N + 1$,

procedure *Backrotate*

rotates the alliance vertically back to the initial state when rank i is stored in $slot(i, 0)$. This procedure is just a series of copying operations. It cannot carry deviations from one column to another.

procedure *Decode*

decodes the Data loaded into all rows $Map(i, n)$ using the code Algeb described in Section 8. Procedure *Encode* performs the corresponding encoding. After encoding as well as decoding, the information is left in the same row.

procedure *Init*(i)

depends now on i, since the top row of the simulated Q'-square is located in a slot dependent on i.

procedure *Compute*(t, i)

simulates t steps (less than the width of the rank after decoding) of the work of the three decoded extended ranks. Even after decoding,

the information is still in dry form. Since the code Dry does not treat all ranks the same way (the higher numbered ranks belong to the Workspace of the Q'-square), procedure *Compute* depends on i.

procedure *ForceCode*

remains much as it was defined in Section 15, only it will also have to work rank-by-rank, due to space shortage.

17.7. The Program

Small colony support can be added to the program below just as easily as in Section 16.3.

```
begin
  for l = 1 to N + 1 do begin
    if l ⩽ N tben t := R else t := U' − NR;
    for i = 1 to ⟨ranks⟩ do begin
      for s = 1 to QP/T do begin
        Input (i, l);
        Decode;
        if modif = 1 and l = 1 tben Init (i);
        Compute (t, i);
        Encode;
        Output (i, s, l);
      end;
      ForceCode;
    end;
  end;
  Backrotate;
end.
```

It is easy to see that the procedures of this program can be written now in such a way as to satisfy the new Size Conditions. Essentially, the only new Size Condition to check is the bound on UT'. The main difference of the present program from the earlier one is that what was earlier an U'/Q'-fold iteration, is now replaced with an iteration U'/R-fold, combined with an iteration ⟨ranks⟩-fold. The extra factor is therefore

$$\langle \text{ranks} \rangle Q'/R \leqslant (Q'/R)^2 \leqslant (|\text{Med}_0| \langle \text{wksp} \rangle)^2.$$

The proof of Theorem 13.1 can now be almost literally carried over to the proof of Theorem 17.1 when the program there is replaced with the present one. □

If we restrict ourselves to economical trajectories of Univ then Theorem 16.1 will also remain true with the new Size Conditions, and the proof carries over virtually without change.

18. PROOF OF THE MAIN THEOREM

18.1. A Series of Implementations

For $k = 0, 1, \ldots$, let us generate a sequence of parameter sets $P', \ldots, r$ each of which satisfies the Size Conditions 17.1. For the reader who skipped Section 17: a sequence satisfying the earlier Size Conditions 15.1 is even easier to construct. The only difference is that P'_k/P'_{k-1} will be about 3|Univ| instead of converging to 1.

The noise bound r_k and the probability p_k are defined as

$$r_k = 2^k, \qquad p_k = \varrho^{r_1 \cdots r_k}.$$

Let w be an integer parameter to be chosen conveniently large later. Let us define the basic colony size P_0 and, for $k \geqslant 0$, the auxiliary parameter g_k as follows.

$$\begin{aligned} P_0 &= g_0 = w, \\ g_k &= (wk)^{11} r_k^2 g_{k-1} \qquad \text{for } k > 0 \end{aligned} \tag{18.1}$$

The absolute and relative colony sizes and work periods P_k, T_k, Q_k, U_k are defined as follows.

$$\begin{aligned} T_k &= P_k g_k \\ P_k &= (wk)^2 r_k T_{k-1}, && \text{for } k > 0, \\ Q_k &= P_k/P_{k-1} = (wk)^2 r_k g_{k-1} && \text{for } k > 0 \\ U_k &= T_k/T_{k-1} = (wk)^2 r_k g_k && \text{for } k > 0. \end{aligned}$$

Finally, the represented colony sizes and work periods are defined as follows.

$$P'_0 = w, \qquad P'_1 = \lfloor Q_1/2 \rfloor,$$

$$P'_k/P'_{k-1} = (1 - k^{-2})Q_k \qquad \text{for } k > 1,$$

$$T'_k = wP'_k.$$

THEOREM 18.1. *For large enough w and small enough ρ, there is a series of implementations Ψ_k for $k > 0$, with parameters P'_k, $T'_k, P_k, T_k, \varrho, p_k$.*

Proof. Let us fix k. We define the parameters

$$P' = P'_{k-1},\ T' = T'_{k-1},\ P = P_{k-1},\ T = T_{k-1},$$

$$Q = Q_k,\ U = U_k,\ Q' = P'_k,\ U' = T'_k,$$

$$\langle \text{wksp} \rangle = wk^2,\ \langle \text{wksp} \rangle' = w(k+1)^2,\ r = r_k.$$

We prove that if w is chosen large enough and ϱ small enough then the above choices satisfy the Size Conditions 17.1 with $\mathsf{Med}_0 = \mathsf{Univ}$, and for $k > 1$ the inequality

$$p_{k-1} \leqslant U_k^{-3(r_k+1)}. \tag{18.2}$$

[This inequality (18.2) corresponds to (9.1).] Then, the application of Lemma 13.1, and Theorem 16.2 completes the proof.

Proof. The first Size Condition can be written as

$$QP(1 - 1/\langle \text{wksp} \rangle) > Q'(1 + |\mathsf{Univ}|/\langle \text{wksp} \rangle')\alpha + 9\langle \text{corr} \rangle' rT. \tag{18.3}$$

We have $Q'\alpha = P'_k P_{k-1}/P'_{k-1} = (1 - k^{-2})P_k$. Therefore the right-hand side of (18.3) can be written as

$$P_k(1 - k^{-2})\left(1 + \frac{|\mathsf{Univ}|}{w(k+1)^2}\right) + 9\langle \text{corr} \rangle' P_k(sk)^{-2}$$

$$< P_k(1 - k^{-2}(1 - |\mathsf{Univ}|/w - 9\langle \text{corr} \rangle'/w^2)).$$

Suppose that w is large enough to have

$$|\mathsf{Univ}|/w + 9\langle \text{corr} \rangle'/w^2 < 0.5.$$

Then the right-hand side of (18.3) is less than

$$QP(1 - 1/\langle \text{wksp} \rangle) = P_k\left(1 - \frac{1}{wk^2}\right).$$

The second Size Condition says, with the substitutions made:

$$P'_{k-1} > \max(\log U_k, \log T'_{k-1}, |\mathsf{Prog}_0|).$$

Here, Prog_0 is the program of the transition function Univ on the Turing machine Turing. The condition $|\mathsf{Prog}_0| < P'_{k-1}$ follows if we set

$$w > |\mathsf{Prog}_0|.$$

We have $\log T'_{k-1} = \log P'_{k-1} + \log w < P'_{k-1}$ since $P'_{k-1} \geqslant w$.

Before going further, let us find some explicit formulas and estimates from the above recursive definitions. We have, with functions $0 < \lambda(k, i) < 1$,

$$g_k = w^{11k}(k!)^{11}2^{2\binom{k+1}{2}} \tag{18.4}$$

$$= \exp_2\left\{2\binom{k+1}{2} + 11k(\log(wk) - 2\lambda(k,1))\right\}, \tag{18.5}$$

$$Q_k = (wk)^{-9}g_k,$$

$$P_k = w^{-9k}(k!)^{-9}g_1 \cdots g_k$$

$$= \exp_2\left\{2\binom{k+2}{3} + 11\binom{k+1}{2}\log(wk)\lambda(k,2)\right\}.$$

The relations

$$\log U_k < 0.5Q_{k-1},$$

$$\log P_k < wk^3 \tag{18.6}$$

are now easy to check. The first one implies

$$\log U_k < 0.5Q_{k-1} < P'_{k-1},$$

which finishes the proof of the second Size Condition. The third Size Condition reads:

$$U > (\langle \text{wksp} \rangle |\text{Univ}|)^2 (U'/Q')(QP/T)^2 Q \log(QP')\,\text{Step}_0. \tag{18.7}$$

We have

$$\begin{aligned} U'/Q' &= T'_k/P'_k = w, \\ (QP/T)^2 &= (P_k/T_{k-1})^2 = (wk)^4 r_k^2, \\ QP' &= Q_k P'_{k-1} < P_k = Q_k P_{k-1}, \\ UT' &= U_k T'_{k-1} = wU_k P'_{k-1} > U_k P_{k-1}. \end{aligned}$$

Hence the right-hand side of (18.7) can be estimated, using (18.6), as

$$\begin{aligned} &(wk^2|\text{Univ}|)^2 w(wk)^4 r_k^2 Q_k \log P_k \,\text{Step}_0 \\ &\quad < |\text{Univ}|^2 w^7 k^8 r_k^2 Q_k wk^3 \,\text{Step}_0 \\ &\quad < (wk)^{11} r_k^2 Q_k = U_k. \end{aligned}$$

This proves the third Size Condition. The remaining two Size Conditions are satisfied by definition.

To prove (18.2), we prove the following stronger inequality, with $R = -\log \varrho$.

$$\log U_k \leqslant r_1 \cdots r_{k-1} R/4r_k = R2^{\binom{k-1}{2}-3}.$$

We have, from (18.4):

$$\log U_k = \log g_k + k + 2\log(wk) < 2\binom{k+1}{2} + 12k \log(wk).$$

This is clearly smaller than the right-hand side of the previous equation, if only R is not too small relative to w. □

18.2. Divisibility Considerations

Let us return to the formulation of the Main Theorem. Given the values n, t, ε, let K be the smallest k with

$$(n + P_k)^2 t \varrho^{r_1 \cdots r_k} < \varepsilon.$$

It is convenient to assume that n is divisible by P'_K. Let us show that if this is not the case then a simple extra coding η of Med_0 into a slightly different medium Med_1 will achieve this. No subtlety is involved, and we mention the details only for completeness' sake. We have seen in Section 15 that a medium on a torus can be treated as it were folded to a medium working on a square, using the mapping $(i, j) \to (\tilde{i}, \tilde{j})$. If we surround the square by "blanks" then the boundaries of the square are automatically marked. And there is no difficulty in simulating a medium on a square by a medium on a larger square. Let

$$n' = P'_K \lceil n/P'_K \rceil \tag{18.8}$$

be the smallest multiple of P'_K greater than n. Thus, using a step-for-step simulation η we order to each configuration of Med_0 over $\mathbf{Z}_n^2$ a configuration of a new medium Med_1 over $\mathbf{Z}_{n'}^2$.

18.3. Deviation Probability

The simulation γ of the theorem will be given as

$$\gamma = \psi_{K-1} \circ \varphi \circ \eta.$$

Here, η is the simulation described in Section 18.2. The simulation ψ_{K-1} is provided by Theorem 18.1. The simulation φ with parameters P'_K, P_K, T'_K, T_K of the medium Med_1 by Univ is given by Theorem 13.1, with the parameters as at the beginning of the proof of Theorem 18.1, with $k = K$. The target space of code γ is $\mathbf{Z}_m^2$ with

$$m = \frac{n'}{P'_K} \frac{P_K}{T_0}. \tag{18.9}$$

We divide by T_0 since in the implementation $\mathbf{\Psi}_0$, the size of a single cell of medium M was chosen T_0.

The number t of steps of the original computation will be followed for $\lceil t/T'_K \rceil$ work periods of the simulation.

Now let z be a trajectory of Med_1, then $y = \varphi_*(z)$ is a trajectory of Univ. Let ξ be a ϱ-perturbation of $\psi_{(K-1)*}(y)$. Then by the definition of implementations, the random evolution $\psi^*_{K-1}(\xi)$ is a self-correcting p_{K-1}-perturbation of y. In each T_K-cube, the probability that the noise belonging to this p_{K-1}-perturbation is not (U_K, r_k, T_{k-1})-sparse, is at most $p'^{r_k}_{K-1} = p_k$. The number of T_K-cubes in the space–time area in question is smaller than $n'^2 t$. It follows that the probability that over this area the noise is not sparse is at most $n'^2 t p_K < \varepsilon$. With probability $1 - \varepsilon$, the noise is sparse. Using Theorem 13.1 we conclude that with probability $1 - \varepsilon$, we have $\varphi^*(\psi^*_{K-1}(\xi)) = z$.

18.4. Redundancy

First we prove

$$P_K, T_K = O(2^{(\log L)^{1.6}}). \tag{18.10}$$

From (18.1), it follows that the quantities g_k, U_k grow approximately as 2^{k^2}:

$$g_k = 2^{k^2 + O(k \log k)},$$

$$U_k = 2^{k^2 + O(k \log k)}.$$

Therefore the quantities P_k, T_k grow approximately as $2^{2k^3/3}$:

$$\begin{aligned} T_k &= 2^{2k^3/3 + O(k^2 \log k)}, \\ P_k &= 2^{2k^3/3 + O(k^2 \log k)}. \end{aligned} \tag{18.11}$$

Above, K was defined as the smallest k with

$$2 \log_\varrho [(n + P_k)^2 t/\varepsilon] \leqslant r_1 \cdots r_k.$$

If $n > P_K$ we get from here, with $L = \log(n^2 t/\varepsilon)$ as defined in the Theorem:

$$L = 2^{K^2/2 + O(K \log K) + O(1)}.$$

The constant in $O(1)$ here depends on ϱ. If $n \leqslant P_K$ the same estimate is obtained. From this, and (18.11), we obtain (18.10).

Now we prove $m = O(n + P_K)$. It follows from (18.8) that $n' = O(n + P'_K)$. Now it follows from (18.9) that $m = O(n + P_K)(P_K/P'_K)$. Finally, it follows from the definitions (18.1) that

$$\frac{P_k}{P'_k} = O\left(\prod_{k>1} \frac{1}{1 - 1/k^2} \right) = O(1).$$

The definitions (18.1) show that the time redundancy is proportional to g_k. By the above equation, this is

$$2^{O(K \log K)} L^2 = L^{2 + o(1)}.$$

The third statement of the theorem concerns the complexity of computation of the code. It was proved in Section 8. □

19. CONCLUSION AND OPEN PROBLEMS

19.1. Simplification

Simplification can be understood in conceptual and in quantitiative sense. In order to lend the two-dimensional medium physical credibility, reducing the number of states (while retaining nearest-neighbor interaction) is crucial. The features required at the cell level include

- computational universality;
- parallel transport with periodic copying;
- application of a generalized Toom's Rule to some local variables (the phase variables);
- periodic majority vote among certain local variables.

These features can probably be achieved by a relatively simple local rule. The elaboration of details would be very interesting.

19.2. Continuous Time

The hierarchical organization used for error correction can also be used for synchronization. Each block of cells is supposed

to be synchronized to a close tolerance and is periodically resynchronized. Higher order blocks are synchronized to progressively looser tolerances. Details must be worked out yet.

If worked out for the one-dimensional case, the result will be a formal refutation of the Positive Rates Conjecture featured in [Li].

For the three-dimensional case, Charles H. Bennett has a physical synchronization idea. If something of comparable simplicity does not work, then the simplicity of the construction of [GR] will be lost with the introduction of continuous time.

19.3. Space–Time Trade-off

This problem was discussed in Section 17. We do not consider the logarithmic lower bound of [DO1] significant as a lower bound on the redundancy of reliable computation. It shows only the necessity of encoding. But in our view, for the purposes of reliable computation, information must be viewed as *coming in encoded form.*

It is remarkable (though probably an artifact) that both here and in [GR], the product of the space and time redundancies came out $\log^2 N$. The bottlenecks in the present paper that caused $\log^2 N$ instead of $\log N$ occur in the problem of consensus. In Section 11.3 more than r^2 repetitions were required for r possible failures. We hope that this result can be improved to $O(r)$ repetitions. However, even more computation was needed in the procedure *FindVotes* that found reliably the votes whose consensus is sought.

19.4. Self-Organization

Our reliable media work reliably only if the starting configuration already contains the (input-independent) hierarchical organization. It would sound more natural, in one dimension, that if the input is one bit then the starting configuration should consist just of the repetition of this one bit. If error correction requires structure then the medium should be able to build up this structure, out of "nothing."

In two dimensions, remembering one bit starting from a homogeneous starting configuration can be done by Toom's Rule. But there is still the problem of *increasing depth in the presence of noise*, as outlined in Section 2, starting from a homogeneous configuration. This is what we consider the natural problem of self-organization.

This goal is the *most intriguing* among the ones proposed. Let us note first of all that Toom's Rule must be largely abandoned even

in two dimensions since its usefulness for structure maintenance depends on a globally existing structure. We must therefore return to the ideas of [G]. Still, we run into new problems at several points.

- Without errors, no structure can arise, since all cells have the same state and the same rule. Hence, randomness is no more than just an "enemy": whatever structures may arise will be random (e.g., if there are blocks, their position will be random).
- Killing a small organized island surrounded by inconsistencies was one of the main principles of the construction in [G]. Self-organization requires that we permit these islands to grow, at least very slowly. A possible new rule replacing the old one could be that islands die if they are prevented from growth for longer time.
- More probabilistic analysis is required with any of these ideas, to show that there will always be islands not too far from the origin whose size is greater than some increasing function of time.

Self-organization is, of course, a favorite topic of theoretical biology. What distinguishes our approach is that all the structure in [G], however life-like, seems to be forced upon us by the extremely simple requirement: the protection of information in a noisy homogeneous medium.

ACKNOWLEDGMENT

Charles H. Bennett's experiments with Toom's media and his discussions of the related physics were a source of inspiration. This is by far not a self-correcting manuscript, therefore my great thanks are due to Weiguo Wang and the referee for their careful reading resulting in the correction of many errors. Supported in part by NSF Grant DCR 8603727. Part of the present work was completed with the support of Bell Communications Research, NJ.

REFERENCES

[B] C. H. Bennett, "Logical depth and physical complexity." In R. Herken, editor, *Universal Turing Machie: a Half-Century Survey*, pages 227–257, Oxford Univ. Press, Oxford, 1988.

[Bl] R. E. Blahut, *Theory and Practice of Error-Control Codes*. Addison-Wesley, Reading, MA, 1983.

[BCG] E. R. Berlekamp, J. H. Conway, and R. K. Guy, *Winning Ways for Your Mathematical Plays*. Academic Press, London, 1982.
[Bo] A Borodin and I. Munro, *Computational Complexity of Algebraic and Numeric Problems*. American Elsevier, New York, 1975.
[BS] P. Berman and J. Simon, "Investigations of fault-tolerant networks of computers," *Proc. 20th ACM Symp. Th. Computing* (1988).
[DO1] R. L. Dobrushin and S. I. Ortyukov, "Lower bounds for the redundancy of self-correcting arrangements of unreliable functional elements," *Problems Inf. Transmiss.* 13(1): 59–65 (1977).
[DO2] R. L. Dobrushin and S. I. Ortyukov, "Upper bound on the redundancy of self-correcting arrangements of unreliable elements," *Problems Inf. Transmiss.* 13(3): 201–208 (1977).
[G] P. Gács, "Reliable computation with cellular automata," *J. Computer Syst. Sci.* 32(1): 15–78 (1986).
[GR] P. Gács and J. Reif, "A simple three-dimensional real-time reliable cellular array," *J. Computer Syst. Sci.* Vol. 36, No. 2, April 1988, pages 125–147.
[Kur] G. L. Kurdyumov, "An example of a nonergodic homogeneous one-dimensional random medium with positive transition probabilities," *Soviet Math. Dokl.* 19(1): 211–214 (1978).
[Kuz] A. V. Kuznietsov, "Information storage in a memory assembled from unreliable components," *Problems Inf. Transmiss.* 9(3): 254–264 (1973).
[La] S. Lang, *Algebra*. Addison-Wesley, Reading, MA, 1965.
[Le] L. A. Levin, "Randomness conservation inequalities; information and independence in mathematical theories," *Inf. Control* 61(1): 15–37 (1984).
[Li] T. M. Liggett, *Interactive Particle Systems* (monograph). Springer, New York, 1985.
[Ly] N. Lynch, "Easy impossibility proofs for distributed consensus problems," *Distributed Computing* 1(1): 26–39.
[M] N. Margolus, "Physics-like models of computation," *Physica* 10D: 81–85 (1984).
[Ta] M. C. Taylor, "Reliable information storage in memories designed from unreliable components," *Bell Syst. Tech. J.* 47(10): 2299–2337 (1968).
[Tof] T. Toffoli and N. Margolus, *Cellular Automata Machines—A New Environment for Modeling*. MIT Press, Cambridge, 1986.
[Too] A. L. Toom, "Stable and attractive trajectories in multicomponent systems," In *Advances in Probability* (Multicomponent Systems), (R. L. Dobrushin, ed.), Vol. 6, pp. 549–575. Dekker, New York, 1980.
[Ts] B. S. Tsirel'son, "Reliable information storage in a system of locally interacting unreliable elements," in *Interacting Markov Processes in Biology*, Lecture Notes in Math., Vol. 653. Springer-Verlag, New York/Berlin, 1978.
[VN] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," In *Automata Studies* (Shannon and McCarthy, eds.). Princeton University Press, Princeton, 1956.

THE COMPLEXITY OF PERFECT ZERO-KNOWLEDGE

Lance Fortnow

ABSTRACT

A *perfect zero-knowledge* interactive proof system convinces a verifier that a string is in a language without revealing any additional knowledge in an information-theoretic sense. We show that for any language that has a perfect zero-knowledge proof system, its complement has a short interactive protocol. This result implies that there are not any perfect zero-knowledge protocols for NP-complete languages unless the polynomial time hierarchy collapses. This chapter demonstrates that knowledge complexity can be used to show that a language is easy to prove.

1. INTRODUCTION

Interactive protocols and zero-knowledge, as described by Goldwasser, Micali, and Rackoff [GMR], have in recent years

Advances in Computing Research, Volume 5, pages 327–343.

ISBN: 0-89232-896-7