Physica 10D (1984) 117–127
North-Holland, Amsterdam

CELLULAR AUTOMATA AS AN ALTERNATIVE TO (RATHER THAN AN APPROXIMATION OF) DIFFERENTIAL EQUATIONS IN MODELING PHYSICS*

Tommaso TOFFOLI
MIT Laboratory for Computer Science, 545 Technology Sq., Cambridge, MA 02139, USA

Cellular automata are models of distributed dynamical systems whose structure is particularly well suited to ultrafast, exact numerical simulation. On the other hand, they constitute a radical departure from the traditional partial-differential-equation approach to distributed dynamics. Here we dicuss the problem of encoding the state-variables and evolution laws of a physical system into this new setting, and of giving suitable correspondence rules for interpreting the model's behavior.

1. Introduction

1.1. *Preview*

We shall present a train of thoughts that in summary runs more or less as follows. (a) There are novel computational resources which on certain tasks may outperform conventional resources by very, very many orders of magnitude. (b) In comparing the two classes of resources, it becomes obvious that the conceptual development of mathematical physics must have been strongly influenced by the nature of the available computational tools. (c) The new resources suggest a new approach to the modeling and simulation of physical systems; in particular, it is possible to replace the customary concepts of real variables, continuity, etc., with more constructive and "physically-minded" counterparts.

1.2 *Infinities in mathematical physics*

Mathematical physics, both classical and quantum-mechanical, is pervaded by the notion of a "continuum," that is, the set R of real numbers with its natural ordering and topology. Maxwell's equations provide a typical example. There, space is a structure S diffeomorphic to R^3, and the electromagnetic field at each point is an element of $Q = R^6$, so that the phase space for the whole field is $Q^S = (R^6)^{R^3}$, a very uncountable state set! On this phase space, we assign a dynamics in the form of a group of transformations T (time) indexed by R.

How do we manage to specify in some constructive way the behavior of a system beset by so many uncountable infinities? Part of the answer is that we do not deal with the "generic" system. Rather, we concentrate on systems having such very special properties (e.g., continuity, uniformity, locality, linearity, or reversibility – Maxwell's equations happen to have all of these properties at once†) that most of the infinities "cancel out," so to speak, and we can make some definite qualitative or quantitative statements about the system's behavior. Of those special properties, the most important for taming infinities is certainly *continuity*. Intuitively, a small uncertainty about the system's initial state leads to a correspondingly small uncertainty about its final state, so that we don't have to worry about capturing its state with "infinite" precision, whatever that may mean. More precisely, in mathematical physics, even when we choose for technical reasons to represent states as encoding an infinite amount of *information*, the

* This research was supported in part by the Defense Advanced Research Projects Agency and was monitored by the Office of Naval Research under Contracts Nos. N00014-75-C-0661 and N00014-83-K-0125, and in part by NSF Grant No. 8214312-IST.

† In particular, the elements of T are continuous with respect to the topology of Q and commute with the elements of S.

0167–2789/84/$03.00 © Elsevier Science Publishers B.V.
(North-Holland Physics Publishing Division)

temporal *correlations* between states introduced by the dynamics may be much more finite*.

In conclusion, in modeling physics with the traditional approach, we start for historical reasons (see below) with mathematical machinery that probably has much more than we need, and we have to spend much effort disabling or reinterpreting these "advanced features" so that we can get our job done *in spite* of them. On the other hand, in this paper, we outline an approach where the theoretical mathematical apparatus in which we "write" our models is essentially isomorphic with the concrete computational apparatus in which we "run" them. Starting from this finitary approach, the few infinities that we may still want to incorporate in a physical theory (e.g., as the size of a system grows without bounds) are defined as usual by means of the limit concept. However, in our approach the natural topology in which to take this limit is that of the *Cantor set*, rather than that of the real numbers.

1.3. *Old and new resources*

Traditional computation, whether by man or machine, involves the sequential processing of a few dozen or at most a few thousand "objects." In symbolic computation the objects are formulas and the processing is done by means of derivation rules, while in numerical computation the objects are finite numbers and the processing entails algebraic operations. (At a more microscopic level both kinds of computation use set operations on very small sets of symbols; however, both computers and people come already hardwired to perform higher-level operations on larger data "chunks.")

Let's consider numerical computation as performed by ordinary computers. As long as fast computer memory is very expensive (as was the case until recently), it is necessary to encode information about a system's state in a very compact way. If an n-bit machine word can encode 2^n different states, then, for instance, we use each one of these states to represent a different value for a scalar quantity†; thus we arrive at the INTEGER or REAL variables, of, say, FORTRAN, where a portion of the real line is chopped up into a number of segments‡ and a different binary code is assigned to each segment. Representation compactness is bought at a price; i.e., processing of these variables requires a rather complex piece of machinery called an "arithmetic/logic unit" (for INTEGER variable), or a much more complex piece of machinery called a "floating-point" processor (for REAL ones); the latter mechanism can be simulated by a lengthy program running on the arithmetic/logic unit. The cost of such hardware is many orders of magnitudes larger than that of a memory word, and thus the customary approach is to time-share it among the few thousand (or million) words that make up the memory.

We shall consider now a different approach. Today, *pure* memory, i.e., without input/output buffering and access circuitry (such as the customary binary-addressing tree), is essentially a free commodity: at one bit per micron square, one could pack ≈ 1 Giga ($\approx 2^{30}$) bits on a 1-inch chip. Then let's be bold, and decide to use some sort of *unary* – rather than *binary* – notation to encode a scalar variable. That is, the value of the variable will be just the number of ones in a certain portion of memory; that's extremely lavish – an integer that used to occupy a 16-bit word will now take 2^{16} bits! However, this extravagance in *storage* buys us certain advantages in *processing*. A scalar variable is now just a "bag" of ones (the position of each bit is irrelevant: each bit has the same weight), and to add two bags we can just "pour" their contents together. As we shall explain in detail later, variables that are encoded in a distributed, local, and

* Cf. a very clear discussion by Everett [2]. The salient point is that "the amount of information in a state" is not as important a concept as "the amount of correlation between two states." While information is measured in a way that has a certain amount of arbitrariness (it depends on the "gauge" chosen), correlation is a "gauge-invariant," absolute quantity.

† We call *scalar* a quantity whose values are naturally ordered and spaced on a linear scale – as contrasted to quantities that range over an unstructured set.

‡ These segments are of uniform width for INTEGER variables, and exponentially increasing width for REAL ones.

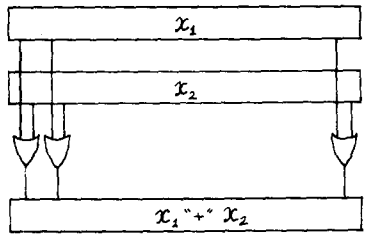

Fig. 1. The "sum" of two unary-encoded scalar variables, as performed by bit-wise ORing.

uniform way (as our bags of ones) naturally lend themselves to processing that is distributed, local, and uniform.

For instance (this is a very naive example – don't laugh but read on!), suppose we have two of these 2^{16}-bit words that we want to add. Let's be extravagant again, and attach a miniature arithmetic/logic unit to each bit. This will be a simple OR gate, no more complex than the FLIP-FLOP which realizes a memory cell. Then we can proceed as in fig. 1.1. After just one propagation delay, the result register will contain the "sum" of the two words. Note that, in this "sum," 1's that are in a homologous position in the two words will contribute only 1 – rather than 2 – to the total. Though the relative error decreases very fast as the 1's in the two words become sparse (intuitively, when our universe consists mostly of vacuum), this is still an approximate and rather inefficient way of doing things, and that's why we call the example "naive." The important point is that all the processing is local. There is no feedback, no carries, no long lines that traverse the whole chip and whose capacitance we have to charge and discharge. All can be done in a fraction of a nanosecond. We wasted many powers of two by using unary rather than binary encoding; but we recoup many powers of two from the fact that in our scheme memory only needs local access and thus can be somewhat denser, and processing is done locally and thus can be extremely faster. It is this lack of overhead that makes our approach attractive for many kinds of physical computation, where one deals with systems that are inherently distributed over spacetime and subjected to laws that are local and uniform.

In the above example we applied local processing to a machine word which encoded a scalar variable and which would typically represent a *lumped* quantity of a physical model. But lumping is usually introduced artificially, in order to adapt a problem to the techniques of ordinary numerical computation. However, if we are able to do local processing at an extremely fine scale, we might as well directly construct our models, much more naturally, as ones in which variables and parameters are *distributed.* We shall introduce a quite general class of such models and discuss their relevance – in the light of the new computational resources – for both theoretical and practical mathematical physics.

2. Cellular automata vs. differential equations

2.1. *Generalities*

We assume some familiarity with the concept of "cellular automaton"*.

In the *cellular-automaton* model of a dynamical system, the "universe" is a uniform checkerboard, with each square or *cell* containing a few bits of data; time advances in discrete *steps*; and the "laws of the universe" are just a small look-up table, through which at each time step each cell determines its new state from that of its neighbors; this leads to laws that are *local* and *uniform.* Such a simple underlying mechanism is sufficient to support a whole hierarchy of structures, phenomena, and properties. Cellular automata provide eminently usable models for many investigations in physics, chemistry, and biology, as well as for experiments in combinatories and for studies in parallel computation [8].

Many theoretical results have appeared on cellular automata. Yet, the fundamental problem (as in the case of partial differential equations, of which

* A well-known example of a cellular automaton is John Conway's game of "life" (cf. Martin Gardner, "The fantastic combinations of John Conway's game 'life'," Scientific American, **223:4** (1970) 120–123). This cellular automaton was shown by Bill Gosper to be computation-universal. For a systematic introduction to cellular automata, refer to Toffoli [8].

cellular automata represent a discrete counterpart) is to determine the temporal evolution of the system, and this problem in general does not have an analytical solution. That is, in order to describe the state q_t that the system will attain in t time-steps starting from a given initial state q_0, the only general method is to construct one-by-one the intermediate states $q_0, \ldots, q_{t-1}$, i.e., to perform numerical integration. Note, however, that because of cellular automata's intrinsic discreteness, here numerical integration is an *exact* process (there are no truncation or round-off errors to worry about), and the results that one obtains have thus the force of *theorems*. In other words, any properties that one discovers through simulation are guaranteed to be properties of the model itself rather than a simulation artifact.

2.2. *Physical realization of cellular automata*

Much as Gutenberg developed a way to reproduce in an arbitrary number of copies *static* information-bearing structures such as text, the introduction of integrated-circuit technology has made it possible to replicate at will *dynamic* information-processing structures such as electronic circuits.

There are four main factors that determine the cost/performance ratio of an integrated circuit, namely, circuit design and layout, ease of mask generation, silicon-area utilization, and maximization achievable clock speed; for a given technology, the latter is inversely proportional to the maximum length of critical signal paths. In terms of these four parameters, cellular automata are perhaps the computational structures best suited for a VLSI realization. In fact, circuit design reduces to the design of a single, relatively simple cell, and layout is uniform; the whole mask for a large cellular-automaton array (that is, not only the cells with their internal connections but also the interconnections between cells) can be generated by a step-and-repeat procedure; essentially no silicon area is wasted on long interconnection lines; and, because of the locality of processing, the length of critical paths is minimal and independent of the number of cells.

Ignore, for a moment, what it is that a cellular automaton actually computes, and whether it can be put to any good use. The fact remains that if I put one-million dollars' worth of cellular-automaton VLSI circuits in a black box, and ask somebody to simulate its behavior with one-million dollars' worth of general-purpose computer, their simulation will be perhaps 10^{12} (one million million) times slower. The challenge is, of course, unfair, because a general-purpose computer is optimized to do other things, but that is exactly the point! In other words, with suitably realized cellular automata one can see things that cannot be seen any other way. Whether these things are worth seeing – well, that's another matter, and this paper attempts to make educated guesses about it. Probably the issue can only be judged *a posteriori*.

In this context, we have constructed a special-purpose cellular-automaton machine [9] which, although based on serial processing, is about a thousand times faster that a general-purpose computer programmed for the same task. Experiments in parallel dynamics using this machine have been very rewarding (cf. Vichniac [10]), and we have confirmed at least in a qualitative way the feasibility of the approach discussed in 2.3 below*.

2.3. *Partial differential equations in spacetime*

Let us consider a partial differential equation with space- and time-independent parameters; for concreteness, let us choose the *heat* equation

$$c\frac{\partial T}{\partial t} = kV^2T. \tag{1.1}$$

This is a mathematical model which is widely used for two reasons: (a) it may represent passably well, at a certain level of description, the behavior of a physical system, and (b) we have a rich catalog of techniques for making *mathematical* deductions from it. To what extent these deductions apply to

* Though from an abstract viewpoint speed is irrelevant, imagine having to do actual genetics research using generations of *elephants* rather than of *Drosophila*!

the *physical* system itself depends on how good correspondence (a) is.

For instance, $T(x, t)$ in (1.1) is a real-valued function defined at each point of (abstract) space and time. In the solution of (1.1), even if we assign T at time $t = 0$ in a quite arbitrary way (for instance, as a sum of step functions), T will be a *continuous* function of x for any $t > 0$. This is important, since operationally we cannot measure a temperature *at a point*; we can only measure the mean temperature over a *finite volume*. The correspondence between system and model is then set up at this level; after that, a point temperature is defined, *within the model*, as the limit of mean temperature as the volume goes to zero. Continuity guarantees that this limit exists.

Now, for volumes that are not too small the correspondence between measured mean temperature and its mathematical counterpart works well, in the sense that as we make the volume smaller the measured values fall within a smaller and smaller interval. However, beyond a certain point this correspondence breaks up, and the smaller we make the volume the wilder are the results that we get. This applies not only to continuity in space but also to continuity in time: a temperature probe will reveal larger and larger fluctuations as its thermal inertia is made smaller.

In conclusion, if eq. (1.1) manages to model well, in the large, certain physical systems, it does so not because it rests on the axiomatics of the calculus, which are not shared with the physical system, but because it must somehow capture other essential properties of a diffusion process, such as locality of effects, conservation of certain quantities, etc. Other mathematical approaches might be as (or more) successful at modeling a physical system in the large, and at the same time provide a better insight into a system's microscopic behavior.

The great advantage of differential equations, such as (1.1), is that we have three centuries' experience with methods for their symbolic integration. As long as all computation had to be done by hand, it paid to stylize the physics in a certain direction so as to be able to handle the resulting mathematics. But few differential equations have a closed-form solution anyway, and the past fifty years have seen numerical computation make bolder and bolder claims at being recognized as an essential part of mathematics.

The moment one gives up symbolic manipulation as a major motive for using differential equations, one starts wondering whether one should still keep them as the starting point for numerical modeling. In fact, they lead to concrete numerical computation (e.g., as run on a general-purpose computer) that is at least three levels removed from the physical world that they try to represent. That is, first (a) we *stylize* physics into differential equations, then (b) we force these equations into the mold of discrete space and time and truncate the resulting power series, so as to arrive at *finite-difference equations*, and finally, in order to commit the latter to *algorithms*, (c) we project real-valued variables onto finite computer words ("round-off"). At the end of the chain we find the computer – again a physical system; isn't there a less roundabout way to make nature model itself?

2.4. *The origin of scalar quantities*

The present subsection introduces in an informal way the main point of this paper.

A partial differential equation whose independent variables are space and time, such as (1.1) above, is translated into a finite-difference equation by the following process: (a) continuous space and time are replaced by a discrete grid, (b) the system's state at each point remains a continuous variable of the same kind (e.g., real, complex, vector) as in the original equation, and (c) derivatives are replaced by differences between state-variables that are contiguous in space and time*.

When one translates the finite-difference equation into a computer algorithm, all one does is (a) *discretize* also the real variables, and further (b) restrict them to a *finite* range. State variables are then represented by finite, though quite large, sets.

* The choice of suitable differences, ostensibly made according to definite "correspondence rules," actually requires a certain amount of black magic to be really successful; cf. Labudde and Greenspan [3] for an interesting discussion.

The dynamics of the system is expressed by essentially the same difference rules, with the proviso that overflow or underflow will abort the computation; if the state set is made too small this aborting will happen so often as to make the method useless.

In terms of structure, if not of interpretation, a cellular automaton is a computational scheme just like the above. The only difference is that the variables at each point of the grid are only allowed to range over a very small set – say, two states per cell, "0" and "1." This, however, has profound consequences.

Clearly, with only two states per cell, it is out of the question to think of a cell as an approximation of a scalar variable (as we did with cells of type REAL or INTEGER in FORTRAN), or to think of a Boolean expression involving the states of adjacent cells as an approximation of a real function of real variables (as we did with FORTRAN's algebra). Intuitively, if a picture's essential features are on the same scale as the picture's "grain", then we have no picture at all. Now, we could program the cellular automaton (by choosing a suitable local rule and suitable initial conditions) to simulate a conventional difference-equation scheme; certain blocks of cells would represent machine words, other blocks would realize arithmetic/logic units, etc. – but this is very unnatural and extremely inefficient. Instead, we shall try an original – and much more natural – approach.

As an aid to intuition, we shall think of 1's as "balls" floating in a "vacuum" of 0's. Let us consider the *mean density* α_V over a certain volume V, i.e., the fraction of cells – within that volume – that are occupied by a ball. α will always be a number between 0 and 1. If V consists of only one cell, then α can take on only two values, that is, either 0 or 1. If V consists of – say – 100 cells, then the possible values for α will sample the interval [0, 1] much more finely: 0.00, 0.01, . . . , 0.99, 1.00. As the size of V grows toward infinity, the range of α approximates better and better the unit on the real-line.

However, in order to speak of a "density field"* we would like to define the density *at a point*. Let x be a point of the grid, and $V_{x,r}$ the "sphere"† of radius r and center on x. Let $\alpha_{x,r}$ be the mean density in this sphere. For the moment, we shall associate with a point x the whole sequence of mean densities $\alpha_{x,1}, \alpha_{x,2}, \ldots$ (without attempting to take its limit as r goes to infinity).

A few remarks are in order. (1) There is a trade-off between spatial resolution and resolution in the density domain. If r is small, we are looking at a definite place, but density is coarse-valued; to get finely-spaced values on the density scale we have to look at a large volume. (2) Let us take a random configuration (of cell states for the cellular automaton). For a fixed r, let us study how the density $\alpha_{x,r}$ varies as we move in space. If $r = 1$ (one-cell radius), then as we move x we get for $\alpha_{x,r}$ a sequence of 0's or 1's, with no correlation between the elements of the sequence. As r increases, the values of α will move up and down the unit interval in a smoother and smoother fashion, so that in the limit we can speak of a continuous function. This continuity is not imposed from above, but arises quite naturally if we observe that large spheres centered on neighboring points have much overlap, and thus share most terms in the summation that defines mean density. (3) We shall see later that, once we introduce a dynamics, we encounter an analogous discontinuity in the small and continuity in the large as we move in *time* rather than in space. (4) Here, we are using unary notation to represent the scalar quantity α, as in the example 1.3, but without committing ourselves to computer words – or "bags" – of a definite size. Moreover, under these circumstances the unary representation is not as wasteful as it might appear on first sight. When we want high resolution, and thus must use large bags, it turns out that most of the bits that make up one bag are shared by the neighboring bags (cf. (2) above); no matter how

* The concept of "density" in this discussion can be taken as a prototype for other scalar variables such as *pressure*, *temperature*, etc.

† The exact shape of this "sphere" does not matter. On an orthogonal lattice one might as well take a cube.

large r is chosen, encoding is done at a *constant* cost of 1 bit per bag! (5) At least so far as the present static picture is concerned, the properties of "density" as defined in our model parallel quite closely those of any of the so-called "point variables" of actual physics (e.g., charge density, temperature); indeed, the latter are "smooth" statistical constructs based on an actual "granular" substrate, and lose their meaning when one, attempting to take a microscopic limit, undermines their very statistical base. (6) Finally – and this is an essential point – the practical trade-off for using small-valued variables interacting only locally is that with the same bulk amount of computer space and time we can handle a grid that is thousands of times finer both in space and in time (cf. 1.3, 2.2). Thus, even though the interactions between such simple cells cannot but be elementary, we can hope to synthesize quite complex interactions through massive iteration. We know that the Gaussian curve can be handled by the mathematician by means of symbols on the paper* and can be drawn to any approximation by the numerical analyst; but whenever we find this curve in nature we don't see the mathematician or the analyst – we see an unsteady hand shooting at the same target over and over and over... (cf. Borel [1] for a very relevant discussion).

And now let's introduce some dynamics. To be specific, assume that balls interact according to some definite local rule but maintain their identity. For example, Norman Margolus has constructed a very clever "billiard ball" cellular-automaton rule of this kind [6] in which balls are conserved, undergo elastic collisions, and all travel at the same speed in one of four possible directions (except during a collision, where they slow down for a moment before bouncing back); this rule also supports clumps of balls that stick together and act as hard mirrors. Margolus' rule is strictly *reversible* – i.e., any configuration for the whole cellular automaton has exactly one predecessor (as well as exactly one successor as in any *deterministic* rule), and *computationally universal*. Reversibility guarantees, among other things, that the relations between microscopic and macroscopic behavior satisfy the laws of thermodynamics, while universality implies that in general there is no analytical shortcut to the system's dynamics – in other words, that there is no better way to tell what the system will do than let it do it and watch it!

Well, if we watch very closely a cellular automaton like this we see a binary computer in operation. But let's stand a certain distance away, and we will see clouds in all shades of gray pulsating and swirling and colliding and mixing .... In other words, if we associate with each point in space the "level of gray," (i.e., the mean density) in its vicinity – rather than just the Boolean value of the cell at that point – this scalar point-variable evolves smoothly, much as if it were "driven" by a differential equation.

Let's see what is involved in this new interpretation. For a fixed r, we have a *density field* $\alpha_{x,r}$. If $r = 1$, this is a Boolean-valued field whose evolution is deterministic and given directly by the cellular automaton rule. If $r \gg 1$, we have a scalar field whose evolution is nondeterministic; however, knowing the field at neighboring points, we can reconstruct from the cellular automaton rule the probability distribution $P(\Delta\alpha)$ that the field will change by an amount $\Delta\alpha$ in one time-step. If P is very sharp, we have a mechanism that is substantially identical to a finite-difference algorithm. We may expect P to be sharp when (a) the rule is suitably chosen, (b) the value of r was selected within a suitable range, and (c) the value of the field is not too close to 0 or to 1. (In our interpretation, 0 and 1 correspond to, respectively, "vacuum" and "infinite" density. Near these extremes our scheme fails to model a finite-differential equation because P will not be sharp enough to give an essentially deterministic result; on the other hand, near the same extremes a FORTRAN program will fail because of overflow or underflow conditions.)

In conclusion, we have an efficient computational mechanism based on microscopic *prim-*

* But already its integral is not expressible in terms of elementary functions.

itives; in this model one can introduce in a rather natural way, as *derived* constructs, macroscopic scalar quantities which, given an appropriate microscopic dynamics, behave much as the quantities predicated by the differential-equation model. The two models are definitely different, but the area of overlap permits one to establish "correspondence rules" between them, so as to arrive at criteria for determining to what extent a microscopic dynamics is indeed "appropriate" for generating the desired macroscopic behavior.

2.5. *A counting argument*

The variety of differential equations that one can write is enormous. In a cellular automaton, on the other hand, once we select the neighborhood on which the new state of cell will depend, all the choice we have for synthesizing the desired behavior is in assigning entries in the look-up table that defines the local map. With few states per cell, this choice doesn't seem too wide. For example, if the new state of a cell depends on the current state of the cell itself and of its immediate neighbors (say, North, South, East, and West) – five neighbors in all – then, with two states per cell, the table will consist of only $2^5 = 32$ binary entries. No matter how cleverly one assigns these entries, one certainly can't do much with the material at hand.

However, even in this simple case the number of different laws that one can write is 2^{32} ($\approx$ one billion!). With nine neighbors (as in the game of "life"), this number climbs to 2^{29} ($\approx 10^{150}$), more than one could explore in the universe's lifetime. Of course, many of these will be trivial variations on the same theme, and most will be utterly uninteresting; but at least we know there is plenty of room to play.

To have even more choice, one can enlarge the neighborhood and use more than two states per cell. However, in the many hours we have spent trying to construct rules that would do what we wanted, we have learned that blind exploration of such an enormous territory is not very rewarding. There are better ways to expand the number of choices in a structured fashion, with more predictable results, making explicit use of analogies from physics or of known combinatorial results. For example, one can make rules that are second-order in time (a class of these automatically yields behavior that is invariant under time reversal); one can make them dependent on the parity of space or time (odd or even steps, or black or white squares on the checkerboard); one can compose into a sequence ("microcode") a small number of different rules involving few neighbors; etc.

After all, even though there exists an uncountable number of differential equations, the set of those that we can explicitly write down is only *countable*, and so is the set of cellular-automaton rules. In conclusion, even though the language of cellular automata uses different primitives than differential equations, there is no a priori reason why it shouldn't have comparable expressive power.

As it happens, we have discovered extremely simple cellular-automaton rules for the "heat" equation (a first-order partial differential equation) and the "wave" equation (second-order). These two equations are the cornerstones of much mathematical physics. (See color plates enclosed in [9].)

3. The concept of continuity in the dynamics of cellular automata

3.1. *Generalities*

The above considerations suggest that, in spite of their discreteness, cellular automata may still support some concepts of continuity and metric, but not the same as in the real-number topology. Then, we shall look elsewhere. We shall start with some miscellaneous considerations.

Observe, first, that while the set of cell-states is *finite* and the set of cells is *countable*, the set of all configurations (i.e., the phase space) is uncountable, and indeed has the cardinality of the real-number *continuum*. Thus, our phase space is as large as that of finite-difference schemes (in spite of

the fact that these use real numbers for cell states). In other words, we have as much infinity to play with as the other guys – only ours is organized in a different way, and locally things are always finite.

Second, continuity means, intuitively, that we can choose states that are so close that their successors are still arbitrarily close. But cell states belong to a finite set, so that there is a discrete jump between one and the other. How can one have an arbitrarily small distance between states in this situation?

Third, all cellular automata having the same "format" (grid shape, number of states per cell) have the same phase space. On the other hand, they may have very different dynamics. In each dynamics the trajectories will interconnect the points of phase space in a totally different way. Can we hope to find a single phase-space topology that is natural and relevant to all of these dynamics?

3.2. *The Cantor-set topology*

Consider the generic cellular automaton. Its *cell-state set* A ("A" for alphabet") is a finite, unstructured collection of symbols, and cannot but be given the discrete topology. Its *phase space* C ("C" for "configurations") is the Cartesian product A^S of countably many copies of A, indexed by the elements of the *space group* S (i.e., the grid's symmetry group). If S were finite, then C would naturally inherit the discrete topology; but for an infinite index set the natural topology for the Cartesian product is the *Tychonoff product topology*, which is coarser than the discrete topology. In the product topology, open sets can be visualized as follows. Let us assign definite values to a finite number of cells, and consider the set of all configurations that match the given assignment (i.e., the values of all other cells are "don't care's.") Call such a set a *pattern.* Then an *open* is an arbitrary collection of patterns.

*I am indebted to Leonid Levin and Douglas Lind for formulating it and suggesting its use in proposition 1.

If the terms of the countable Cartesian product are finite sets (having, of course, more than one element), as in our case, then the Tychnoff product topology coincides with the topology of the familiar *Cantor set* (... take the unit segment on the real line, remove the middle third, and iterate on what is left.) Thus, we get the same "Cantor-set" topology for *all* nontrivial cellular automata (i.e., those having at least one dimension and at least two states per cell).

3.3. *A topological characterization of cellular automata*

Now, one can prove the following. Let τ be the automaton's global map (i.e., its dynamics, or the generator of the time group). Then, for all cellular automata,

Property 1 (*continuity*). τ is *continuous* with respect to the Cantor-set topology [7].

We also know that, by definition,

Property 2 (*commutativity*). τ *commutes* with any element σ of S (the space group)

(Briefly, time and space commute.) Finally, for every cellular automaton

Property 3 (*local finiteness*). There exists a continuous function $q: C \to Q$, where Q is a finite set, such that, if c, c' are distinct configurations, there exist a shift $\sigma \in S$ for which $q\sigma(c) \neq q\sigma(c')$.

By taking $Q = A$, q can be interpreted as a "window" function which projects a configuration on the coordinate axis of a selected cell. This obvious property is used in proposition 1 below to rule out certain pathological cases*.

The important fact in all of this is that, among the dynamical systems consisting of the Cantor set with a dynamics τ and a discrete group of transformations σ,

Proposition 1. Properties 1–3 above constitute a complete characterization of cellular automata.

Thus, we see that, in addition to local finiteness and commutativity between space and time (properties which are put in the very definition of a cellular automaton), *continuity in the Cantor-set topology* is the characterizing property of the dynamics of cellular automata. (The various components of this characterization were rediscovered in bits and pieces by several workers in cellular-automaton theory – including myself [8] – but had been known for a while, under the heading of "shift dynamical systems," to more professional mathematicians [4, 5].)

3.4. *The local point of view*

We shall try to interpret the results of the above subsection.

The main point is that to understand what goes on in a cellular automaton it is not necessary to look at an entire, infinite configuration. Rather, one's attention can be turned to specific place, and one's scope should be widened, in concentric circles, so to speak, only as longer and longer evolution times are considered. Thus, the customary picture which represents the state of a system as a point tracing an orbit in phase space is somewhat misleading: in general, we cannot handle in a finitary way the exact position of the point itself (which encodes an infinite amount of information); on the other hand, we can project the point on a finite subset of axes, and we can enlarge this subset as need requires. We shouldn't try to (and, at any rate, we *can't*) take in the whole picture!

We shall give but one example of the "conspiracy" that forces us to take a local viewpoint. The Cantor set is a *metric* space, that is, it admits of metrics compatible with its topology. What does this repertoire of metrics have to offer? We are faced with the problem of finding a satisfactory metric (a yardstick for "closeness") for a *uniform* system that extends infinitely in space. Because of spatial symmetry, all cells "look the same;" intuitively, we would require of our metric that if we change a 0 into a 1 in a given cell, we should move away a certain distance in phase space, and this distance should be the same no matter which cell we choose. But it turns out [8] that none of such "uniform" metrics is compatible with the Cantor-set topology; in which direction should we relax our requirements?

Here is one way. In spite of being immersed in a uniform sea of cells, we shall pick one arbitrarily. By what criterion? Well, by where *we* are! In comparing two configurations, we make a list of the places where they don't match; a mismatch occurring "here" will be given a weight of $\frac{1}{2}$, and in general any mismatch occurring within a shell of thickness 1 and radius r will be given a weight $2^{-(r+1)}kr^{d-1}$ (where d is the dimensionality of the space), so that successive shells will contribute at most 1/4, 1/8, etc. Thus, the distance between two identical configurations will be 0 and the distance between two configurations that differ everywhere will be 1. This metric is compatible with the Cantor topology. In this metric, two configurations get closer and closer as the nearest point where they differ moves away from us*. Quite seriously, we could characterize the Cantor topology as the topology of self-centeredness. What is nice is that other observers, with their own center of interest different from ours, may choose their own version of the metric, but nonetheless we will all agree on the same topology.

4. Conclusions

We have presented and motivated a new mathematical approach to the modeling of distributed physical systems. This approach is suggested by the availability of new, high-performance simulation

* To make the connections with the traditional δ–ϵ criterion for continuity, recall an obvious property of cellular automata, i.e., that the speed of propagation of information is bounded. If two configurations coincide within a radius r, and thus have a distance less than $\approx 2^{-r}$ then their successors will coincide within a radius of at least $r-1$, and thus their distance will be less than $\approx 2^{-(r-1)}$. This is all that is needed to arrive at a δ–ϵ criterion.

tools, and yields models whose formal structure closely matches that of the available computational resources.

In particular, we have discussed a concept of continuity that is adequate for an operational approach to physics over the whole range from macroscopics to microscopics, and yet does not postulate, like differential equations, an infinite amount of information within a finite volume.

Acknowledgements

I am greatly indebted to Edward Fredkin for giving the start to this train of thoughts, and to two anonymous referees for many helpful suggestions.

References

[1] Émile Borel, Probabilities and Life (Dover Ps, London, 1962).

[2] Hugh Everett, "The Theory of the Universal Wave Function", The Many World Interpretation of Quantum Mechanics, DeWitt and Graham, eds., (Princeton Univ. Press, Princeton, 1973) pp. 3–140.

[3] R.A. Labudde and Donald Greenspan, "Discrete Mechanics—A General Treatment", J. Comput. Physics 15 (1974) 134–167.

[4] G.A. Hedlund, "Endomorphysms and Automorphisms of the Shift Dynamical System", Math. Syst. Theory 3 (1969) 320–375.

[5] H.B. Keynes and J.B. Robertson, "Generators for Topological Entropy and Expansiveness", Math. Syst. Theory 3 (1969) 51–59.

[6] Norman Margolus, "Physics-like Models of computation", Physica 0D (1984) (these proceedings).

[7] D. Richardson, "Tessellations with Local Transformations", J. Comput. System Sci. 6 (1972) 373–388.

[8] Tommaso Toffoli, "Cellular Automata Mechanics", Tech. Rep. No. 208, Logic of Computers Group, CCS Dept., The University of Michigan (November 1977).

[9] Tommaso Toffoli, "CAM: A High-Performance Cellular Automata Machine," Physica 10D (1984) 195 (these proceedings).

[10] Gérard Vichniac, "Simulating Physics with Cellular Automata", Physica 0D (1984) (these proceedings).