Zeitschrift: Helvetica Physica Acta
Band: 62 (1989)
Heft: 5

Artikel: Problems in quantifying self-generated complexity
Autor: Grassberger, Peter
DOI: https://doi.org/10.5169/seals-116045

Helvetica Physica Acta
Vol. 62 (1989)

0018-0238/89/050489-23$1.50+0.20/0

PROBLEMS IN QUANTIFYING SELF-GENERATED COMPLEXITY

by

Peter Grassberger

Physics Department, University of Wuppertal
D - 5600 Wuppertal 1, Gauss-Strasse 20

Abstract:

We review possible measures of complexity which might in particular be applicable to situations where the complexity seems to arise spontaneously. We point out that not all of them correspond to the intuitive (or "naive") notion, and that one should not expect a unique observable of complexity. We finally concentrate on quantities which measure in some way or other the difficulty of classifying and forecasting sequences of discrete symbols, and study them in simple examples.

1. INTRODUCTION

The nature which we observe around us is certainly very "complex", and the main aim of science has always been to reduce this complexity by descriptions in terms of simpler laws. But very often, physicists have achieved this by studying archetypical and simple situations ("Gedanken experiments"), and maybe for that reason the study of complex behaviour has not been pursued very much in the past by physicists.

In the most complex situations studied traditionally in statistical mechanics, the large number of degrees of freedom can usually be reduced to few "order parameters". This is called the "slaving principle" by H. Haken [1]. Mathematically, it is related to central limit theorems of statistics and to center manifold theorems [2] of dynamical system theory.

But there are extremely simple systems (like the cellular automaton called "game of life" [3]) which can be programmed as universal computers in which case a purely statistical description is of course inappropriate. And there are systems like those describable by a quadratic map

$$x_{n+1} = a - x_n^2 \quad , \quad x_n \in [-a, a] \quad , \quad a \in [0,2] \qquad (1.1)$$

where even the reduction to a single order parameter does not imply a simple behaviour. The complexity of the latter can be seen in a number of ways. First of all, the behaviour can depend very strongly on the parameter a: there is a set of positive measure on which the attractor is chaotic [4], but this is believed to be nowhere dense, while windows with periodic attractors are dense. Secondly, at the transitions from periodicity to aperiodicity there are (an infinite number of) "Feigenbaum points" [5], each of which resembles a critical phenomenon. The richness inherent in eq.(1) becomes even more obvious if we let x_n and the parameter a be complex. The resulting Julia and Mandelbrot sets (see e.g. fig.1) have become famous even among a wider public for their intricate structure [6].

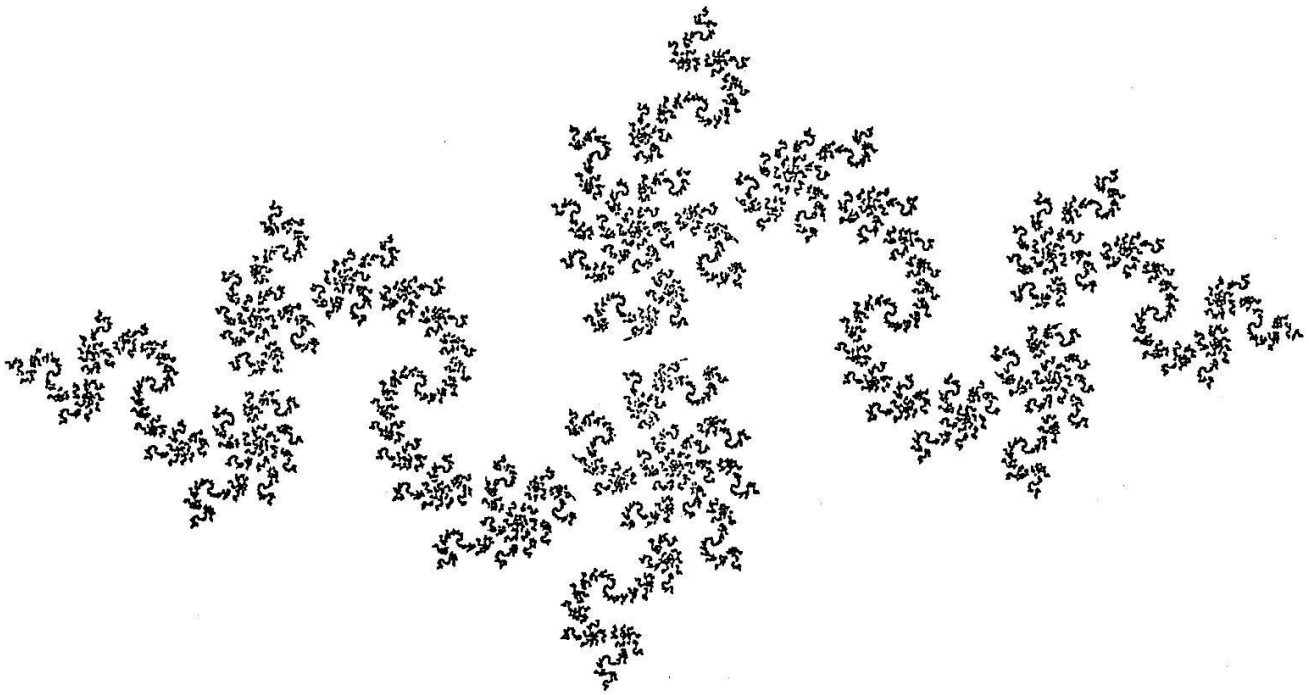

Fig.1: Julia set of the map $z' = z^2 - 0.86 - 0.25i$.

Finally (and most importantly for us), the trajectories of eq.(1) themselves can be very complex. To specify this, let us first discretize trajectories by defining [7] variables $s_n \in \{L,R,C\}$ with $s_n=L$ if $x_n<0$, $s_n=R$ if $x_n>0$, and $s_n=C$ if $s_n=0$ (similar "symbolic dynamics" can be defined also for other dynamical systems, with the mapping from the trajectories $(x_n ; n\in Z)$ onto the "itineraries" (s_n) being one-to-one for nearly all starting points x_0). The complexity of the map is reflected then in the fact that the itineraries obtained in this way show very specific structure, both "grammatically" (i.e., disregarding probabilities) and probabilistically.

Systems of similar complexity are e.g. the output of nonlinear electronic circuits, reversals of the earth's magnetic field, and

patterns created by chemical reactions or by hydrodynamic turbulence. Beautiful examples of the latter are pictures of Jupiter, with numerous turbulent eddies surrounding the giant red spot.

One characteristic common to all these instances is that the complexity is self-generated in the narrow sense that the formulation of the problem is translationally invariant, and the observed structure arises from a spontaneous breakdown of translational invariance (in the case of Jupiter, it is azimuthal rotation invariance which is broken). But the most interesting case of self-generated complexity in a wider sense is presumably life itself.

If we want to understand better the generation of complex behaviour, we have at first a semantic problem: there does not seem to exist a universally accepted and formalized notion of what is "complexity", though most of us certainly would agree intuitively that such a notion should exist. As physicists used to work with precise concepts it should thus be one of our first aims to find such a precise notion. In the ideal case, it should be fully quantitative, i.e. there should be an observable and a prescription for measuring it.

In a search for such an observable, we shall in the next section scan the literature. We shall find several concepts which have been proposed, and all of which have advantages and drawbacks. We shall argue that indeed no unique measure of complexity should exist, but that different definitions can be very helpful in their places. In the last section, we shall apply them to measure the complexity of some symbol sequences like those generated by the quadratic map or by cellular automata.

2. MEASURES OF COMPLEXITY

a) General

Among the phenomena which in the physics literature are considered as complex are, among others, chaotic dynamical systems, fractals, spin glasses, neural networks, quasicrystals, and cellular automata (CA). Common features of these and other examples are the following:

(1) They are somehow situated *between disorder and* (simple) *order* [8], i.e. they involve some hard to describe and not just random *structures*. As an example, consider fig.2. There, virtually nobody would call the left panel complex. Some people hesitate between the middle and right panels when being asked to point out the most complex one. But once told that the right one is created by means of a (pseudo-)random number generator, the right panel is usually no longer considered as complex.

(2) They often involve *hierarchies* (e.g. fractals and spin glas-

ses). Indeed, hierarchies have often been considered as a main source of complexity (see, e.g., [8]).

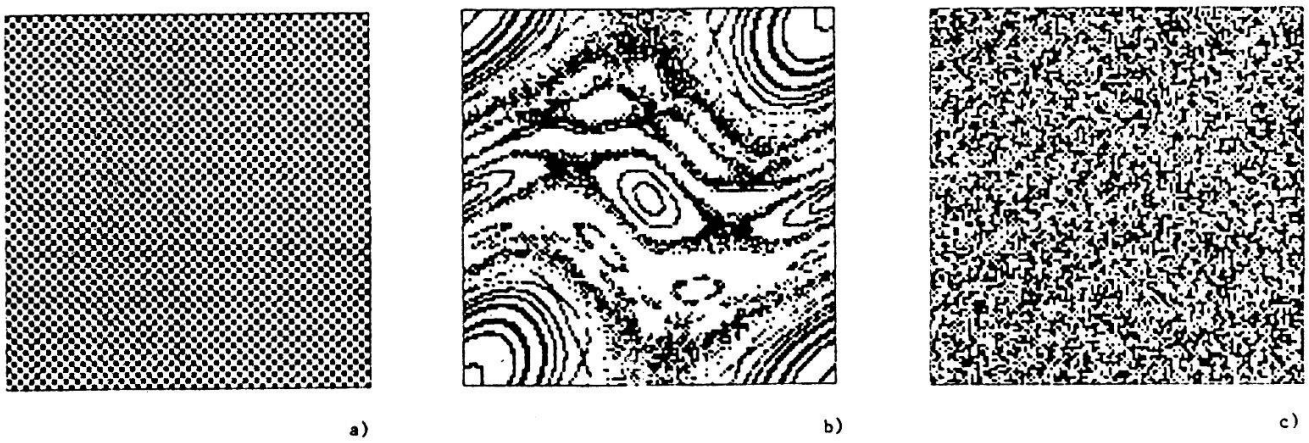

Fig.2: Three patterns used to demonstrate that the pattern that one would intuitively call the most complex is neither the one with the lowest entropy (left) nor the one with the highest (right). That is, complexity is *not* equivalent to randomness, but rather is between order and chaos.

(3) But as the example of human societies shows most clearly, a strict hierarchy can be ridiculously simple when compared to what is called a "*tangled hierarchy* " in ref.[9]. This is a hierarchy violated by *feedback* from lower levels, creating in this way "strange loops". Feedback as a source of complexity is also obvious in dynamical systems. On the logical (instead of physical) level, in the form of *self reference*, feedback is the basis of Goedel's theorem which seems closely tied to complex behaviour [9].

(4) In a particular combination of structure and hierachy, an efficient and meaningful description of complex systems usually requires *concepts of different levels*. The essence of self-generated complexity seems to be that higher-level concepts arise without being put in explicitely.

As a simple example, consider figs.3 to 5. These figures show patterns created by the 1-dimensional CA with rule nr.110 in Wolfram's [10] notation, in decreasing resolution. In this CA, a row of "spins" with $s \in \{0,1\}$ is simultaneously updated by the rule that neighbourhoods 111, 100, and 000 give a "0", while the other 5 neighbourhoods give "1". Figure 3 shows that this CA has a periodic invariant state with spatial period 14, i.e. a 14-fold degenerate "ground state". In a first shift from a low-level to higher-level description, we might call these "vacua", although the original vacuum is of course the state with zeroes only. Figure 4 shows that between different vacua there are kinks which on a coarser scale propagate like particles. In fig.5, finally,

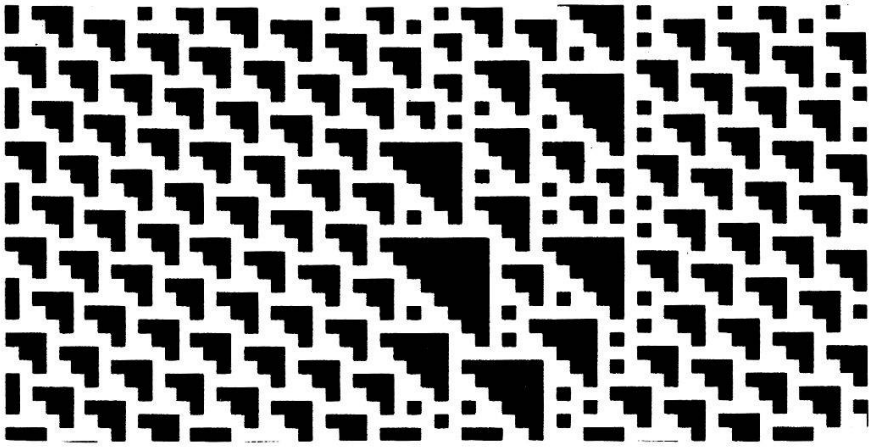

Fig.3: Pattern created with CA nr.110 from a random start. The periodic pattern with spatial periodicity 14 and time periodicity 7 seems to be attractive, but between two phase-shifted such patterns there must be stable kinks. Time increases downward.

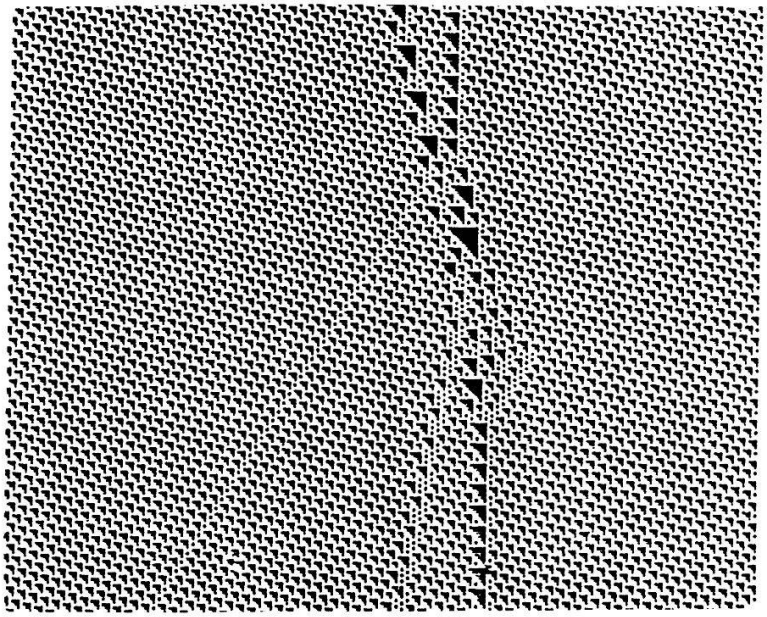

Fig.4: Collision between two kinks for CA 110 [11]. Notice that the kinks behave like "particles". There are such "particles" with at least 6 different velocities.

only the "particles" are shown, and we see a rather complicated evolution of a gas of these particles in which the original concept (the spins) are completely hidden. Notice that nothing in the original formulation of the rule had hinted at the higher level concepts (vacua and particles).

(5) Complex systems are usually composed of *many parts*, but this alone does not yet qualify them as complex: an ideal gas is not more complex than a human brain because it has more molucules than

sion of information and complexity will be a recurrent theme in the following. *Understanding* the meaning is just the act of replacement of a detailed description containing all irrelevant and redundant information by a compressed description of the relevant aspects only.

(7) From the above we conclude that complexity in some very broad sense always is a *difficulty of a meaningful task*. More precisely, the complexity of a pattern, a machine, an algorithm, etc., is the difficulty of the most important task related to it.

By "meaningful" we exclude on the one hand purely mechanical tasks, as for instance the lifting of a heavy stone. We do not want to relate this to any complexity. But we also want to exclude the difficulty of coding, storing, and reproducing a pattern like the right panel of fig.2, as the details of that pattern have no meaning to us.

A technical problem is that when we speak about a difficulty, we have to say what are our allowed tools and what are our most important limitations. Think e.g. of the complexity of an algorithm. Depending on whether CPU time is most seriously limited or core memory, we consider the time or space complexity as the more important. Also, these complexities depend on whether we use a single-CPU (von Neumann) or multiple-CPU computer.

(8) Another problem is that we don't have a good definition of "meaning", whence we cannot take the above as a real definition of complexity. This is a problem in particular when we deal with self-generated complexity.

In situations which are controlled by a strict outside hierarchy, we can replace "meaningful" by "functional", as complex systems are usually *able to perform some task* [14].

But in systems with self-generated complexity it is not obvious what we really mean by a "task". Also, we must be careful to distinguish between the mere *ability* to perform a task, and the *tendency* or *probability* to do so.

Let me illustrate this again with cellular automata. As we said, the "game of life" can be programmed as a universal computer.This means that it can simulate *any* possible behaviour, given only the right initial configuration. For instance, we can use it to compute the digits of π, but we can also use it to proof Fermat's last theorem (provided it is right). Thus it can do meaningful tasks if told so from the outside. But this does not mean that it will do interesting things when given a random initial condition. Indeed, for most initial conditions the evolution is rather disappointing: for the first 400 time steps or so, the complexity of the pattern seems to increase, giving rise (among others) to many gliders [3], but then the gliders collide with all the interesting structures and destroy them, and after about 2000 time steps only very dull structures survive. This is in contrast

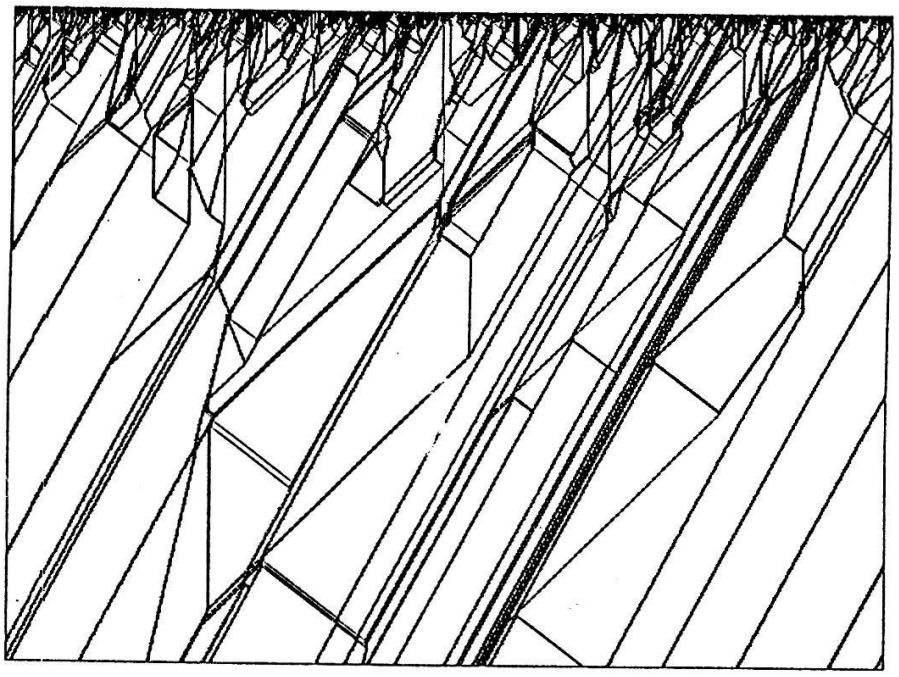

Fig.5: Evolution of a random initial state for CA 110. Every pixel represents a block of 14 spins. The pixel is white if the block is the same as the neighbouring block to the left, otherwise it is black. In this way, only the kinks are seen. In order to comress vertically, only every 21th time step is shown.

the brain has nerve cells. What is important is that there are strong and non-trivial *correlations* between these parts [12]. Examples are of course fractals and critical phenomena, but the correlations there seem still very simple compared to those in some cellular automata [13].

The correlations between the parts of a complex object imply that the whole object in a way is *"more than its parts"*. Let me be more specific. Of course, we can describe the whole object completely by describing its parts, and thus it cannot be more than these. But such a description will be redundant, and it misses the fact that the correlations probably indicate that the object as a whole might have some meaning.

(6) Thus, finally, complexity seems somehow related to the problem of *meaning* [14]. Persueing this seriously would lead us too far from physics into deep questions of philosophy, so I will not do this. But let me just point out that a situation acquires some meaning to us if we realize that only some of its features are essential, that these features are related to something we have already stored in our memory, or that its parts fit together in some unique way. Thus we realize that we can replace a full and detailed description by a compressed description which captures only these essential aspects, employing thereby already existing information. We mention this since the relation between *compres-*

to real life, which certainly has also started with a random initial configuration, but which is still becoming more and more complex.

(9) As a consequence of our insistence on *meaningful* tasks, the concept of complexity becomes *subjective*. We really cannot speak of the complexity of a pattern without reference to the observer. After all, the right-hand pattern in fig.2 might have some meaning to somebody, it is just we who *decide* that it is meaningless.

This situation is of course not new in physics. It arises also in the Copenhagen interpretation of quantum mechanics, and it appears also in Gibbs' paradoxon. In the latter, the entropy of an isotope mixture depends on whether on wants to distinguish between the isotopes or not. Yet it might be unpleasant to many, in particular since the dependence on the observer is much less trivial than in Gibbs' paradoxon.

Indeed, statistical mechanics suggests an alternative to considering definite object as we pretended above. It is related to the observation that when we call the right panel of fig.2 random, we actually do not pay attention to the fine details of that pattern. We thus do not really make a statement about that very pattern, but about the class of all similar patterns. More precisely, instead saying that the pattern is not complex, we should (or could, at least) say "that pattern seems to belong to the class of random patterns, and this *class* is trivial to characterise: it has no features" [15]. The question of what feature is "meaningful" is now replaced by the question of what ensemble to use.

We have thus a dichotomy: we can either pretend to deal with definite objects, or we can pretend to deal only with equivalence classes of objects or probability distributions ("ensembles"). In the latter case we avoid the problems of what is "meaning" by simply *defining* the ensembles we want to study. This simplifies things somewhat at the expense of never knowing definitely whether the objects we are dealing with really belong to the ensemble, resp. whether they aren't objects appearing with zero probability.

In the following, this dichotomy will be seen more precisely in several instances. In the tradition of physics, I will usually prefer the latter attitude.

Also in the tradition of physics and contrary to the main attitude of computer science, I will always stress *probabilistic* aspects in contrast to purely algorithmic ones. Notice that the correlations mentioned under point (5) are defined most easily if one deals with probability distributions. For a conjecture that correlations defined not probabilistically but purely algorithmically are basic to a mathematical approach to life, see ref. [16].

In this way, the complexity of an object becomes a *difficulty related to classifying the object, and to describing the set or rather the ensemble to which it belongs.*

These general considerations have hopefully sorted out somewhat our ideas what a precise definition of complexity should be like. They have made it rather clear that a unique definition with a universal range of application does not exist (indeed, one of the most obvious properties of a complex object is that there is no *unique* most important task associated with it).

Let us now take the above as a kind of shopping list, let us go to the literature, and let us see how the things we find there go together with it.

b) Space and Time Complexity of Algorithms [17]

We have already mentioned shortly the complexity of algorithms. The space and time complexities of an algorithm are just the required storage resp. CPU time. A family of problems depending on a discrete parameter N is considered complex (more precisely "NP-hard") if the fastest algorithm solving the problem for every N needs a time which increases exponentially, although the formulation of the solution alone would at most increase polynomially.

Although this is of course a very useful concept in computer science, and moreover fits perfectly into our broad definition of complexity, it seems of no relevance to our problem of self-generated complexity. The reason is that an algorithm is never self-generated but serves a purpose imposed from outside.

c) Algorithmic Complexity (Kolmogorov-Chaitin [18,19])

Let us consider an infinite sequence $S = s_1 s_2 \ldots$ of binary digits. Sequences and patterns using other discrete alphabets can be encoded by binary sequences, and as we have seen in the introduction, this holds also for the orbits of continuous dynamical systems. Thus the restriction to binary sequences is not a severe limitation.

The shortest program which produces on some given universal computer U the first N digits of S and then stops will be called $P_U(N)$, and its length (in bits) is called $M_{U,N}(S)$. The algorithmic or Kolmogorov complexity (per digit) of S is then

$$C(S) = \lim_{N \to \infty} M_{U,N}(S)/N \qquad (2.1)$$

Notice that C(S) is independent of U since any universal computer can be simulated by any other with a finite-length simulator. But notice also that this independence holds only due to the limit taken in eq.(2.1). Algorithmic complexity is *not uniquely defined for finite sequences* .

Since algorithmic complexity is obviously a measure of *infor-*

mation (thus called in the following also "algorithmic information"), we might wonder how it is related to Shannon entropy h. The first and main difference is that algorithmic information deals with *individual sequences*, while Shannon entropy is a property associated to *probability distributions*. Assume that S is drawn from some probability distribution P with stationary block probabilities $p_N(s_1 s_2 \ldots s_n)$. Then h is defined as

$$h = \lim_{N\to\infty} H_N/N \,, \qquad H_N = \sum_{\{s_1\}} p_N(s_1 \ldots s_N)\ log\ p_N(s_1 \ldots s_N) \,. \tag{2.2}$$

(here and in the following, *log* means logarithm to base 2). This difference is of course somewhat blurred by the restriction to infinite sequences, but there are cases where C(S) is clearly different from h.

One such case are the digits 3141592... of pi. Since there exist very efficient programs for the digits of π (of length *log* N for N digits), the algorithmic information of this sequence is zero. But these digits *look* completely random by any statistical criterion [20], thus it is conjectured that their entropy is maximal.

The problem with the digits of pi is that, although the sequence might *look* random, pi is of course not just a random number but carries a very special meaning. Technically, the difference between C(S) and h stems from the fact that C(S) measures the information needed to specify the *first* N digits, while h measures the much larger *average* information needed to specify *any* N consecutive digits.

For symbol sequences generated by self-organizing systems, this difference is absent. There, the first digits neither have more meaning nor are in any other way singled out from the other digits, and we conclude that there C(S) = h for P-nearly all sequences S.

Thus, for the instances we are interested in, algorithmic information is just a measure of randomness, and not a measure of complexity in our sense. It does measure the difficulty of a task (namely storing and retrieving a code for the sequence), but in requireing the code to be faithful to all irrelevant details, this is not the most meaningful task.

d) Ziv-Lempel Complexity [21]

For any finite sequence, we would get a trivial algorithmic information also because we could always build a computer with a special button just for this sequence, in which case the program

$P_{u,N}(S)$ would be trivial. But the situation is even worse.

Given an infinite sequence with unknown origin, we cannot estimate reliably its algorithmic information, since we never know whether we indeed have found its *shortest* description. More technically, C(S) is not *effectively computable*. A way out of this problem is to remember that when measuring a difficulty, we could arbitrarily restrict our tools if this seems convenient. In the literature, there exist several suggestions of how to restrict the encodings of a sequence in order to obtain effectively computable algorithmic information measures. The best known and simplest one is due to Ziv and Lempel [21].

There, the sequence is broken into words W_1, W_2, ... such that $W_1 = s_1$, and W_{k+1} is the shortest new word immediately following W_k. For instance, the sequence S = 110101001111010 01... is broken into (1)(10)(101)(0)(01)(11)(1010)(01... . In this way, each word W_k with k>1 is an extension of some W_j with j<k by one single digit. It is encoded simply by referring to the code of W_j, and adding the extension. The Ziv-Lempel complexity of S is then given via the length M of the resulting encoding of S as $C_{ZL}(S) = \lim_{N\to\infty} M/N$. It is shown in ref.[21] that $C_{ZL} = h$ for nearly all sequences, i.e. the Ziv-Lempel complexity agrees again with the Shannon entropy. Indeed, $C_{ZL} = h$ holds also for sequences like the digits of pi since, like h, the Ziv-Lempel coding is only sensitive to the statistics of (increasingly long) blocks of length $<< N$.

In practice, Ziv-Lempel coding is a very efficient method of data compression [22], and methods related to it are among the most efficient ones for estimating entropy [23]. But it should be clear that Ziv-Lempel "complexity" is a measure of randomness (even in cases where algorithmic information is more subtle), and not a measure of complexity in our sense.

e) Logical Depth [24]

A complexity measure more in our spirit is the "logical depth" introduced by C. Bennett [24]. The logical depth of a string S is essentially the time needed for a general purpose computer to actually *run the shortest program* generating it. Thus the task is now not that of storing and retrieving the shortest encoding, but that of actually performing the decoding. The difference with time complexity (sec.2b) is that now we do not ask for the time needed by the *fastest* program, but rather the *shortest*.

The reason why this is a good measure in particular of self-generated complexity is Occam's razor: if we find a complex pattern of unknow origin, it is reasonable to assume that it was generated from essentially the shortest possible program. The program must have been assembled by chance, and the chance for a meaningful program to assemble by chance decreases exponentially

with its length.

For a random string S, the time needed to generate it is essentially the time needed to read in the specfication, and thus it is proportional to its length. In contrast to this, a string with great logical depth might require only a very short program, but decoding the program takes very long, much longer than the length of S. The prime example of a pattern with great logical depth is presumably life [24]. As far as we know, life emerged spontaneously, i.e. with a "program" assembled randomly which had to be very short. But it has taken some 10^9 years to work with this program until life has assumed its present forms.

A problem in the last example is that "life" is not a single pattern but rather an ensemble. Noise from outside was obviously very important for its evolution, and it is not at all clear whether we should include some of this noise as "program" or not.

A more formal example with (presumably) large logical depth is the central vertical column in fig.6. This figure was obtained with cellular automaton nr. 86, with an initial configuration con-

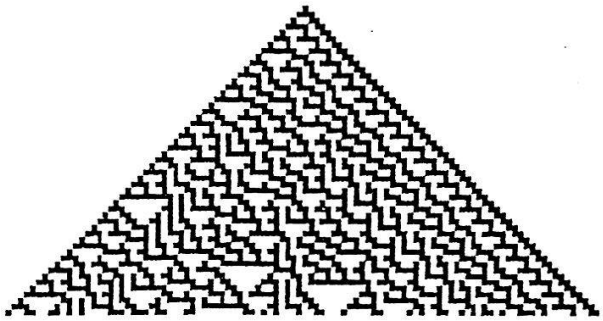

Fig.6: Pattern generated by CA #86, from an initial configuration having a single "1".

sisting of a single "1". Since both the initial configuration and the rule are very easy to describe, the central column has zero Kolmogorov complexity. From very long simulations it seems however that it has maximal entropy [25]. Furthermore, it is believed that there exists no other way of getting this column than by direct simulation. Since it takes $\alpha\ N^2$ opeations to iterate N time steps, we find indeed a large logical depth.

Logical depth shares with algorithmic complexity the problem of not being effectively computable. This problem is indeed even worse here. While we never could know whether there wasn't a *shorter* program for a not yet fully understood sequence, this shorter program could take either more *or* less time to execute.

f) Sophistication [26]

In practical computers there is a distinction between *program* and *data*. While the program specifies only the class of patterns, the data specify the actual object in the class. A similar distinction exists in data transmission by conventional (e.g., Huffman [27]) codes, where one has to transmit independently the rules of the code and the coding sequence.

It was an important observation by Turing that this distinction between program and data is not fundamental. The mixing of both is e.g. seen in the Ziv-Lempel code, where the coding sequence and the grammatical rules used in compressing the sequence are not separated. For a general discussion showing that the rule vs. data separation is not needed in communication theory, see ref.[28].

What the conventional separation into data and proper program suggests is that the combined program length M_N is a convex function of the sequence length, and increases asymptotically like $M_N \approx \text{const} + hN$, where the additive constant is the program length. For a sequence of finite algorithmic information, e.g., the total program length is shown schematically in fig.7. The offset of the asymptotically tangent line on the y-axis is the proper program length.

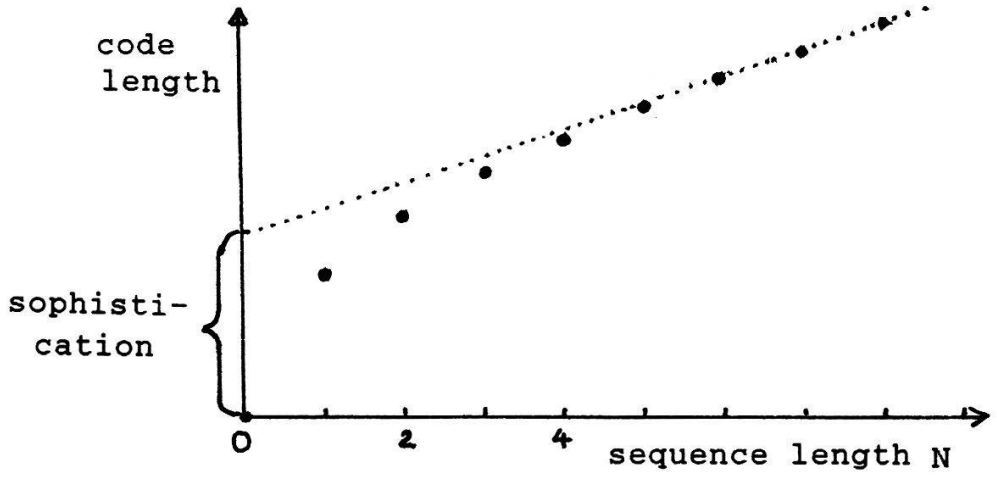

Fig.7: Total program length for a typical sequence with finite algorithmic information per digit, and with finite proper program length ("sophistication"; schematically).

It was shown by Koppel and Atlan [26] that this is essentially correct. The proper program length, called by them "sophistication", can moreover be defined such that it is indeed independent of the computer U used in defining M_N.

Sophistication is a measure of the importance of rules in the sequence. Equivalently, it is a measure of the importance of correlations. Rules imply correlations, and correlations between suc-

cessive parts of the sequence S imply that the description of a previous part of S can be re-used later, reducing thus the overall program length. This aspect of complexity in an algorithmic setting had been stressed before in ref.[16].

As we said already, the increase of average code length with N has been studied for probabilistic models in [28]. In particular, it was shown there that for simple models such as moving average or autoregressive models which depend on k real parameters one has

$$\langle M_N \rangle \approx hN + \tfrac{1}{2}k \log N \ . \qquad (2.3)$$

This is easily understood: for an optimal coding, we have somehow also to encode the k parameters, but for finite N we will only need them with finite precision. If the central limit theorem holds, their tolerated error will decrease as $N^{-\frac{1}{2}}$, whence we need $\frac{1}{2} \log N$ bits per parameter.

Equation (2.3) shows that unfortunately sophistication is infinite already in rather simple situations. This problem that the proposed measures tend often to degenerate to zero or infinity is also common to other complexity measures.

g) Effective Measure Complexity [15]

Let us now discuss a quantity similar in spirit to sophistication, but formulated entirely within Shannon theory. There, one does distinguish between rules and data, with the idea that the rules are encoded and transmitted only once, while data are encoded and transmitted again and again. Thus the effort in encoding the rules is negligible.

The average length of the data of a sequence of length N is the block entropy H_N defined in eq.(2.2). Just like the code lengths M_N, the H_N are convex, and thus their differences

$$h_N = H_{N+1} - H_N \qquad (2.4)$$

are monotonically decreasing to the entropy h. When plotting H_N versus N, we get a figure exactly like fig.7. The quantity corresponding to the sophistication is now called *effective measure complexity* (EMC),

$$\begin{aligned} EMC &= \lim_{N \to \infty} [\, H_N - N(H_N - H_{N-1}) \,] \\ &= \sum_{k=0}^{\infty} (h_k - h) \ . \end{aligned} \qquad (2.5)$$

The EMC has a number of interesting properties. First of all, within all stochastic processes with the same block entropies up to some given N, it is minimal for the Markov process of order N-1

compatible with these H_k. In particular, it is finite for Markov processes even if these depend on real parameters (in contrast to sophistication), so that it seems to be non-trivial.

Secondly, it can be written as a sum over the non-negative decrements $\delta h_N = h_{N-1} - h_N$ as

$$EMC = \sum_{N=1}^{\infty} N \, \delta h_N \quad . \tag{2.6}$$

The decrement δh_N is just the average amount of information by which the uncertainty of s_{N+1} decreases when learning s_1, and when all spins s_k between are already known. Thus EMC is the average usable part of the information about the past which has to be remembered at any time if one wants to be able to reconstruct the sequence S from its shortest encoding. Consequently, it is a lower bound on the average information kept about the past if one wants to make an optimal forecasting.

Finally, in contrast to all previous complexity measures it is an effectively computable observable, to the extent that the block probabilities $p_N(s_1 \ldots s_N)$ can be measured.

The main drawback of the EMC in comparison to an algorithmic quantity like sophistication is of course that we can apply it only to sequences with stationary probability distribution. This includes many interesting cases, but it excludes many others.

h) Complexities of Grammars [17]

A set of sequences (or "strings") over a finite "alphabet" is usually called a formal language, and the set of rules defining this set is called a "grammar". In agreement with our general remark that complexities are preferably to be associated to sets or ensembles, it is natural to define a complexity of a grammar as the difficulty to state and/or apply its rules.

There exists a well-known hierarchy of formal languages, the *Chomsky hierarchy* [17]. Its main levels are in increasing complexity and generality: *regular languages, context-free languages, context-sensitive languages, and recursively enumerable sets.* They are distiguished by the generality of the rules allowed in forming the strings, and by the difficulty involved in testing whether some given string belongs to the language, i.e. is "grammatically" correct.

Regular languages are by definition such that the correctness can be checked by means of a finite directed graph. In this graph, each link is labeled by a symbol from the alphabet, and each symbol appears at most once on all links leaving any single node (such graphs are called "deterministic"). Furthermore, the graph has a unique start node. Any grammatically correct string is then

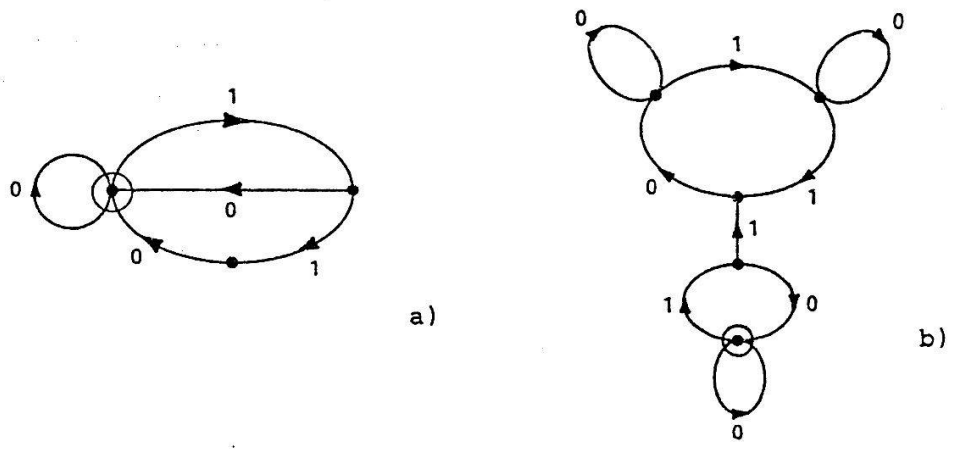

Fig.8: Deterministic graphs for the regular languages generated in 1 time step by CA rules nr. 76 (a) and 18 (b). The heavy nodes are the start nodes.

represented by a unique walk on the graph, while any wrong string is not. Scanning the string consists in following the walk on the graph.

Examples of graphs for regular languages are given in fig.8. They correspond to strings allowed in the second generation of two cellular automata, if any string is allowed as input in the first generation. Figure 8(a) corresponds e.g. to the set of all strings without blocks of three consecutive "1"s, and with no further restriction.

One migth define the complexity of the grammar as the difficulty to *write down the rules*, i.e. essentially the number of nodes plus the number of links. However, in ref.[29] the *regular language complexity* (RLC) was defined as

$$RLC = log\ n\ , \tag{2.7}$$

where n is the number of *nodes* alone, of the smallest graph giving the correct grammar (usually, the graph of a grammar is not unique [17]). This makes indeed sense as the so defined RLC is essentially the difficulty in *performing the scan*: during a scan, one has to remember the number of the present node, in order to look up the next node(s) in a table, and then to fetch the number of the next node. If no probabilities are given, the average information needed to fetch a number between 1 and n (and the average time to retrieve it) is *log* n.

Assume now that one is given not only a grammar but also a stationary probability distribution, i.e. a stationary *ensemble*. This will also induce probabilities P_k (k=1,...n) for being at the k-th node of the graph at any given time. Unless one has equidistribution, this will help in the scan. Now, both the average information about the present node and the time to fetch the next one will be equal to the "*set complexity*" (SC),

$$SC = -\sum_{k=1}^{n} P_k \log P_k \ . \qquad (2.8)$$

It is obvious that the SC is never larger than the RLC, and is finite for all regular languages. It is less obvious that while the RLC is by definition infinite for all other classes in the Chomsky hierarchy, the same need not hold for the SC. Indeed, at least all context-free and context-sensitive languages can be represented by infinite deterministic graphs [30], and very often one has measures whose P_k decrease so fast with the distance from the start node that the SC is finite [15].

i) Forecasting complexity [15,31]

Both the RLC and the SC can be considered as related to a restricted kind of forecasting. Instead of just scanning for correctness, we could have as well forecasted what symbol(s) is resp. are allowed to appear next. In a purely algorithmic situation where no probabilities are given, this is indeed the only kind of meaningful forecasting.

But if one is given an ensemble, it is more natural not only to forecast what symbols *might* appear next, but also to *forecast the probabilities* with which they will appear. We call forecasting complexity (FC) the average amount of information about the past which has to be stored at any moment, in order to be able to make an optimal forecast.

Notice that while the Shannon entropy measures the *possibility* of a good forecast, the FC measures the *difficulty* involved in doing so. That these need not be correlated is easily by looking at left-right symbol sequences for quadratic maps (the symbol "C" appears for nearly all start values x_0 with zero probability, and can thus be neglected in probabilistic arguments). For the map $x' = 2 - x^2$, e.g., *all* R-L sequences are possible [7] and all are equally probable. Thus, no non-trivial forecasting is possible, but just for that reason the *best* forecast is a trivial guess. In contrast, at the Feigenbaum point [5] the entropy is zero and thus perfect forecasting is possible, but as shown below, the average amount of information about the past needed for an optimal forecast is infinite.

Notice that the FC is essentially just the logical depth, applied to the case of an infinite string drawn from a stationary ensemble. Assume we want to reconstruct such an infinite string from its shortest code. Then we are provided only an information of h bits per symbol (h = entropy), and we are supposed to get the rest of 1-h bits per symbol from the past. But this involves exactly the same difficulty as an optimal forecast.

In addition to be related to the logical depth, the FC is

also closely related to the EMC. As discussed in sec.g, the EMC is a lower bound on the average amount of information which has to be stored at any time, since it is the time-weighted average of the amount of information by which the uncertainty about a future symbol decreases when getting to know an earlier one. Thus, we have immediately [15]

$$EMC \leq FC \quad . \tag{2.9}$$

Unfortunately, this seems to be a very weak inequality in most cases where it was studied [15,31].

The difficulty of forecasting the Feigenbaum sequence mentioned above comes from the fact that eq.(2.5) for the EMC is logarithmically divergent there.

3. APPLICATIONS

a) Complexities for the Quadratic Map [32]

For the R-L symbol sequences generated by the quadratic map, we have different behaviour in chaotic, periodic, intermittency, and Feigenbaum points. In the chaotic domain, we have also to distinguish between Misiurewicz (band-merging) points and typical chaotic points.

While all complexities are zero for periodic orbits, the EMC (and the SC) is infinite at Feigenbaum and intermittency points. The reason is that there the block entopies H_N diverge logarithmically. Thus the symbol sequences are there very restricted, but checking such a sequence for correctness is very difficult (the same holds, by the way, also for Penrose tilings of the plane).

For chaotic orbits, we have the problem (mentioned already in the analog context of sophistication) that for sequences depending on real parameters the forecasting complexity is infinite: when forecasting a very long sequence, it helps in using the control parameter a with ever increasing precision, leading to a divergent amount of work per symbol.

This does not apply to the EMC and to the SC. Block entropies seem (numerically) to converge exponentially, so the EMC seems to be finite in general [33]. Also, there exists a simple algorithm for approximate grammars which accept all sequences which contain no forbidden words of length $\leq n$ with any n [34]. Except at Misiurewicz points, the size of these graphs diverges with n (so that the RLC is infinite), but numerically the SC seems to stay finite. Thus one needs only a finite effort per symbol to check for grammatical correctness, for nearly all sequences with respect to the natural measure [32].

b) Complexities of Grammars for Cellular Automata [29, 15]

Wolfram [29] has studied the grammars of spatial strings s_i, $i \in Z$, generated by 1-dimensional CA's after a finite number of iterations (the input string is taken as random). He finds that after any finite number of iterations the laguages are always regular (this holds no longer if one goes to CA's with ≥2 dimensions, or to the strings after infinitely many iterations [35]).

One finds that for some rules the RLC increases very fast with the number of iterations. In many cases this corresponds to an actually observed intuitive complexity of the generated patterns, but for some rules (like, e.g., rules 32 or 160) the generated patterns seem rather trivial. In these latter cases, there is indeed a large difference between the RLC and the SC, the latter being very small [15]. Thus, most parts of the deterministic graphs needed to scan a sequence is hardly ever used there.

c) Forecasting Complexities for Cellular Automata [31]

The only class of sequences for which we were able to compute finite forecasting complexities exactly were sequences generated by CA's after *a single iteration step*. Cellular automata with just one iteration are of course ridiculously simple systems, and one might expect very trivial results. But this is not at all so.

Assume there is a random stream of input bits, and at each time step one output bit is formed out of the last 3 input bits. The question one is asked is to predict as good as possible the probbilities $p_i(0)$ and $p_i(1)$ for the i-th output bit to be 0 or 1, based only on the knowledge of previous output bits (in physics language, the input sequence is a "hidden variable").

It is possible to give the optimal stategy for such an forecast [31]. It involves constructing a deterministic graph similar to those needed for regular languages, but now to each link is attached, in addition to the label s, a forecast p(s). The FC is then given by the Shannon formula (2.8). While it is true that the FC is finite for all elementary CA's, the graphs are infinite for many rules [31]. Part of the graph for rule 22 (described in more detail below) is given in fig.9. For more details, see ref.[31].

d) Effective Measure Complexity for Cellular Automaton #22 [13]

The most interesting application in view of possible self-generation of complexity is to the CA nr.22. In this cellular automaton, rule for spin updating is that 001, 010, and 100 give "1", while the other 5 neighbourhoods give "0". When starting from a single "1", one get a Pascal's triangle, and when starting from a random configuration one finds a pattern (fig.10) which at first sight looks not very interesting.

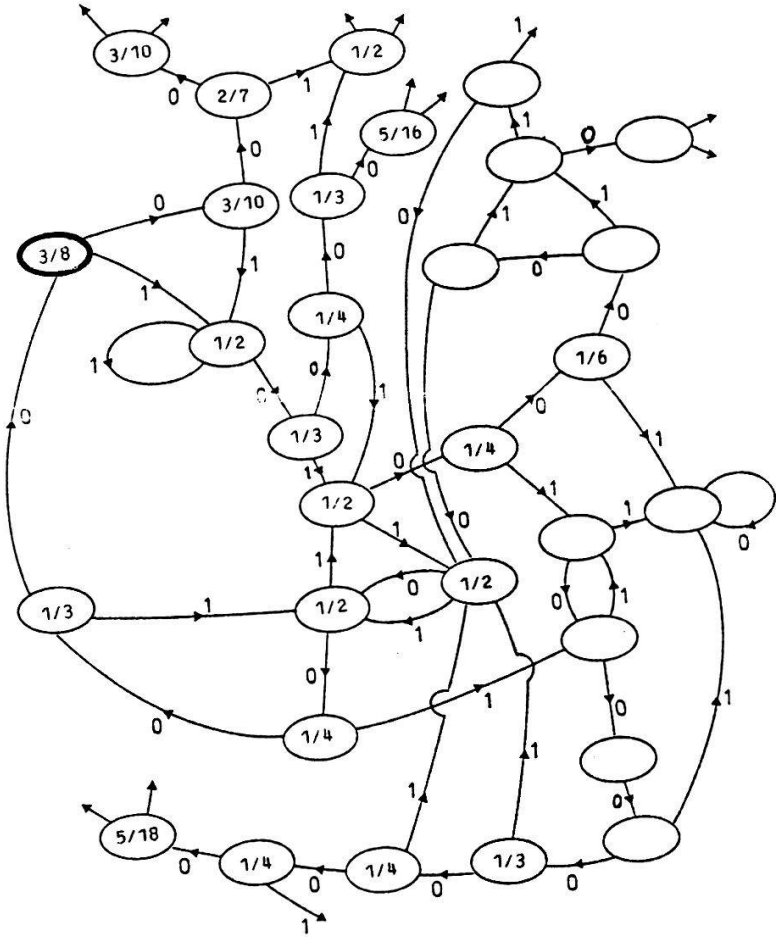

Fig.9: Part of the graph needed to forecast a sequence generated by CA rule 22 after 1 iteration. The heavy node is the start node. To each link is associated a fixed forecasting probability p(s), where s (the label of the link) is the next output symbol. In some of the nodes, we have indicated the value of p(s=1).

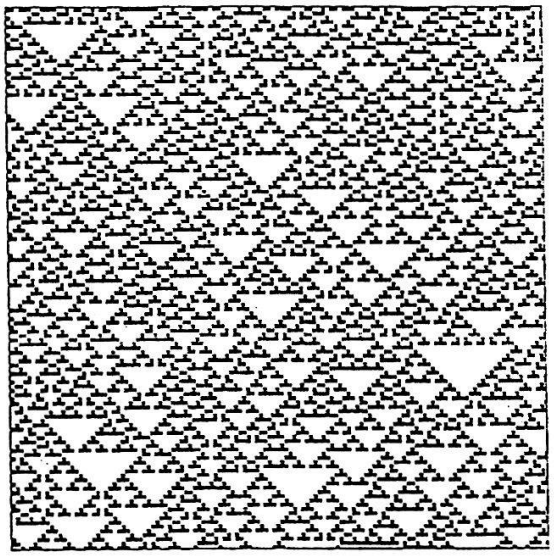

Fig.10: Pattern generated by CA nr.22 from a random start. Time runs again down, space horizontally.

A closer inspection shows, however, that the block entropies for horizontal and for vertical blocks in the stationary state (reached after several thousand iterations) do not seem to increase linearly with the block length. Otherwise said, the differences h_N seem to decrease like powers of N (fig.11), such that both temporal and spatial strings have zero entropy in the stationary state. This is very surprising as the block entropies do diverge. Hence these strings are neither periodic nor fractal nor quasiperiodic nor random. Indeed, to my knowledge they are not like anything which other authors have encountered anywhere else.

The interesting aspect of this result is that it is very similar to an important aspect of life. Life is self-organizing in the sense that it leads to very special forms, i.e. from a wide basin of attraction it leads to a much narrower set of meaningful states. But this alone would not yet be surprising. The surprising aspect is that this attraction is not at all rigid. Although the

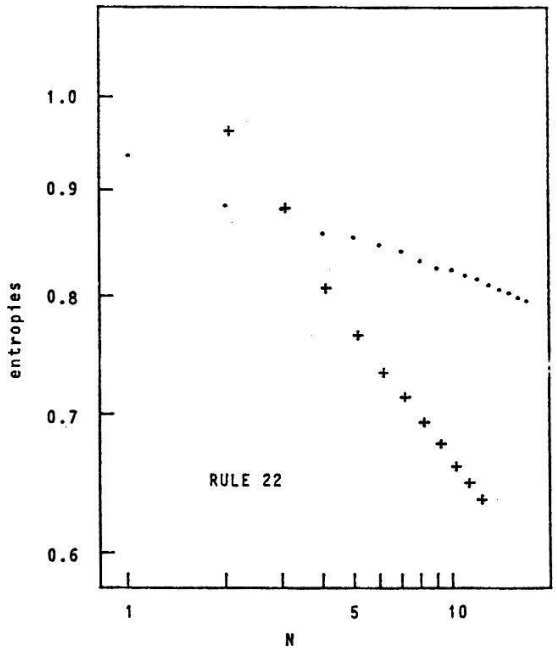

Fig.11: Entropies h_N for spatial sequences in bits (dots) and time entropies in natural units (crosses) in the stationary state of CA 22.

"attractor" is very small compared to full phase space, it is still huge and it therefore allows for a wide spectrum of behaviour. It is exactly this which is shared by rule 22. Having zero entropy, the attractor is extremely constrained, but having divergent block entropies it is still vast.

I have looked at a number of other sequences whether they show similar long-range correlations, leading to a similarly small entropy. One does not seem to encounter this in symbol sequences generated by dynamical systems with few degrees of freedom. One does however encounter it to some degree in natural languages. In

written English, e.g., the entropies h_N decrease from 4.4 bits per character for N=1 to less than 1 bit for large N [23]. Finally, an attempt to analyse in a similar way DNA sequences failed due to extremely high repetition rates and non-stationarities. While the assumption that written English forms an ensemble with well-defined statistical properties seems reasonable, this is not so for DNA.

REFERENCES

1. H. Haken, *Synergetics: an Introduction* (Springer, Berlin 1983)

2. J. Guckenheimer and P. Holmes, *Non-Linear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields* (Springer, New York 1983)

3. E.R. Berlekamp, J.H. Conway, and R.K. Guy, *Winning Ways for your Mathematical Plays* (Academic Press, London 1983)

4. M.V. Yakobson, Commun. Math. Phys. **81**, 39 (1981)

5. M. Feigenbaum, J. Stat. Phys. **19**, 25 (1978); **21**,669 (1979)

6. Peitgen and P. Richter, *The Beauty of Fractals* (Springer, 1986)

7. P. Collet and J.-P. Eckmann, *Iterated Maps on the Interval as Dynamical Systems* (Birkhäuser, Basel 1980)

8. H. Simon, Proc. Am. Phil. Soc. **106**, 467 (1962)
H.A. Cecatto and B.A. Huberman, Physica Scripta **37**, 145 (1988)

9. D.R. Hofstatter, *Goedel, Escher, Bach: An Eternal Golden Braid* (Vintage Books, New York 1980)

10. S. Wolfram, Rev. Mod. Phys. **55**, 601 (1983)

11. S. Wolfram, *Glider Gun Gidelines*, Princetown preprint (1985)

12. M.H. van Emden, *An Analysis of Complexity* (Mathematical Centre Tracts, Amsterdam 1975)

13. P. Grassberger, J. Stat. Phys. **45**, 27 (1986)

14. H. Atlan, Physica Scripta **36**, 563 (1987)

15. P. Grassberger, Int. J. Theor. Phys. **25**, 907 (1986)

16. G.J. Chaitin, Toward a Mathematical Definition of 'Life', in *The Maximum Entropy Principle*, R.D. Levine and M. Tribus, eds. (MIT Press, Cambridge, Mass., 1979)

17. J.E. Hopcroft and J.D. Ullman, *Introduction to Automata*

Theory, Languages, and Computation (Addison-Wesley, 1979)

18. A.N. Kolmogorov, Three Approaches to the Quantitative Definition of Information, Probl. of Inform. Theory 1, 3 (1965)

19. G.J. Chaitin, Journal of the A.C.M. 22, 329 (1975)

20. S. Wagon, Mathem. Intell. 7, 65 (1985)

21. J. Ziv and A. Lempel, IEEE Trans. Inform. Theory 24, 530 (1978)

22. T.A. Welch, Computer 17, 8 (1984)

23. P. Grassberger, to be published in IEEE Trans. Inform. Theory; G. Fahner and P. Grassberger, Complex Systems 1, 1093 (1987)

24. C.H. Bennett, in *Emerging Syntheses in Science*, D.Pines ed., 1985

25. S. Wolfram, *Random Sequence Generation by Cellular Automata*, to appear in Adv. Appl. Math.

26. M. Koppel and H. Atlan, *Program-Length Complexity, Sophistication, and Induction*, preprint (1987)

27. R.W. Hamming, *Coding Theory and Information Theory* (Prentice-Hall, Englewood Cliffs, 1980)

28. J. Rissanen, IEEE Trans. Inform. Theory 30, 629 (1984)

29. S. Wolfram, Commun. Math. Phys. 96, 15 (1984)

30. K. Lindgren and M. Nordahl, Chalmers Univ. preprint (1988)

31. D. Zambella and P. Grassberger, to be published in Complex Systems (1988)

32. P. Grassberger, Z. Naturforsch. 43a, 671 (1988)

33. G. Györgyi and P. Szepfalusy, Phys. Rev. A31, 3477 (1985)

34. F. Hofbauer, Israel J. Math. 34, 213 (1979); 38, 107 (1981)

35. M. Nordahl, Chalmers Univ. thesis (1988)