Easily calculable measure for the complexity of spatiotemporal patterns

F. Kaspar and H. G. Schuster
Institut für Theoretische Physik der Universität Frankfurt, Robert-Mayer-Strasse 8-10, D-6000 Frankfurt am Main, Federal Republic of Germany
(Received 15 December 1986)

We demonstrate by means of several examples that an easily calculable measure of algorithmic complexity c which has been introduced by Lempel and Ziv [IEEE Trans. Inf. Theory **IT-22**, 25 (1976)] is extremely useful for characterizing spatiotemporal patterns in high-dimensionality nonlinear systems. It is shown that, for time series, c can be a finer measure for order than the Liapunov exponent. We find that, for simple cellular automata, pattern formation can be clearly separated from a mere reduction of the source entropy and different types of automata can be distinguished. For a chain of coupled logistic maps, c signals pattern formation which cannot be seen in the spatial correlation function alone.

I. INTRODUCTION

One of the recent major breakthroughs in the theory of dynamical systems is the discovery that already-low-dimensional systems can show chaotic movement in phase space which can be characterized by a spectrum of dimensions and entropies.[1,2] However the numerical effort needed to extract these spectra is rather large. This limits their determination to systems with dimensionality lower than ten.

In order to characterize chaotic motion in high-dimensional dynamical systems, e.g., spatiotemporal turbulence, poorly stirred chemical reactions, etc.,[1] it is therefore necessary to develop new tools. In this article we want to make a step in this direction. The idea is to demonstrate by means of several examples that an easily calculable measure of algorithmic complexity, which has been introduced mathematically by Lempel and Ziv,[3] is a useful quantity to characterize spatiotemporal patterns. Our main intention is to provide experimentalists with a practical tool.

The remainder of this article is organized as follows. In Sec. I we introduce the measure of algorithmic complexity $c(n)$ for finite strings of symbols of length n which has been suggested by Lempel and Ziv[3] and explain the algorithm which allows its numerical calculation by means of a flow diagram. In Sec. II we calculate $c(n)$ for time series generated by simple one-dimensional maps. It will be shown that for a piecewise linear map $c(n)$ increases with the Liapunov exponent λ until it saturates for a finite value of λ, and that for the logistic map $c(n)$ mirrors the complexity of the Metropolis-Stein-Stein sequences.[4] Both examples demonstrate that $c(n)$ reflects the order which is retained in the one-dimensional temporal pattern better than its chaotic behavior. In Sec. III we use $c(n)$ to characterize the time evolution of some simple one-dimensional cellular automata (CA's).[5] It is found that, starting from random initial conditions, pattern formation is indicated by a decrease of $c(n)$ with time (where n now measures the spatial extension of the CA) and that it can be discriminated from a mere reduction of the source entropy. Different types of CA can be distinguished via $c(n)$. In Sec. IV we compute $c(n)$ for a chain of coupled logistic maps.[6,7] Although conventional measures of pattern formation such as the spatial correlation function and the power spectrum show no indication that the system reaches a state of dynamical equilibrium, it is found that the temporal development of $c(n)$ shows clearly that such a state can be reached within different time scales which depend on the parameters of the system. The dynamical equilibrium corresponds to a situation where the average number of kinks remains approximately constant.[7] This example demonstrates that $c(n)$ is indeed a useful measure by which the development of spatiotemporal patterns can be characterized. In Sec. VI we draw our conclusions and discuss the relation of $c(n)$ to other measures of complexity which have been suggested, e.g., by Grassberger[8] and Wolfram.[9]

II. THE COMPLEXITY MEASURE OF LEMPEL AND ZIV

In the following we introduce the complexity measure of Lempel and Ziv.[3] We consider for simplicity only strings which are composed of zeros and ones; extensions to larger numbers of variables can be found in Ref. 3. According to Kolmogorov[10] and others,[11] the complexity of a given string of zeros and ones is given by the number of bits of the shortest computer program which can generate this string. A general algorithm which determines such a program cannot be given.[11] Lempel and Ziv[3] have chosen from all possible programs one class that allows only two operations: to copy and to insert (see below). Furthermore they have not calculated the length of the program which generates a given string of length n but a number $c(n)$ which is a useful measure of this length. We will not repeat here the mathematical proofs of Lempel and Ziv, which show that $c(n)$ is an appropriate measure of the Kolmogorov complexity. Instead we will explain the algorithm which is needed to calculate $c(n)$ and explain the usefulness of $c(n)$ by way of several examples.

The calculation of $c(n)$ proceeds as follows. Let us assume that a given string $s_1 s_2 \cdots s_n$ has been reconstructed by the program up to the digit s_r and that s_r has been newly inserted (i.e., it was not obtained by simply copying it from $s_1 s_2 \cdots s_{r-1}$). The string up to s_r will be denoted by $S = s_1 s_2 \cdots s_r \cdot$, where the dot indicates that s_r is newly inserted. In order to check whether the rest of S, i.e., $s_{r+1} \cdots s_n$ can be reconstructed by simple copying (or whether one has to insert new digits), Lempel and Ziv proceeded with the following steps. First, one takes $Q \equiv s_{r+1}$ and asks whether this term is contained in the vocabulary (i.e., in one of the substrings or words) of the string S so that Q can simply be obtained by copying a word of S. This is, of course, equivalent to the question of whether Q is contained in the vocabulary $v(SQ\pi)$ of $SQ\pi$ where $SQ\pi$ denotes the string which is composed of S and Q and π means that the last digit has to be deleted (i.e., here $SQ\pi = S$). This last formulation of the question can be generalized to situations where Q also contains two (i.e., $Q = s_{r+1} s_{r+2}$) or more elements. Let us, for example, assume that s_{r+1} can indeed be copied from the vocabulary of s. Then, by using the formulation given above, we next ask whether $Q = s_{r+1} s_{r+2}$ is contained in the vocabulary of $SQ\pi$ [i.e., in $v(Ss_{r+1})$] and so on until Q becomes so large that it can no longer be obtained by copying a word from $v(SQ\pi)$ and one has to insert a new digit. The number c of production steps to create a string, i.e., the number of newly inserted digits (plus one if the last copy step is not followed by inserting a digit), is used by Lempel and Ziv as a measure of the complexity of a given string.

Let us give a few examples. If we have a sequence which contains only zeros we could intuitively say that it should have the smallest possible complexity of all strings. Indeed one has only to insert the first zero and can then reconstruct the whole string by copying this digit, i.e.,

$$00000 \cdots \rightarrow 0 \cdot 000 \cdots ,$$

and the complexity of this string is $c = 2$.

Similarly one finds for a sequence which is only composed of units 01, i.e.,

$$01\,01\,01 \cdots \rightarrow 0 \cdot 1 \cdot 01\,01 \cdots ,$$

the value $c = 3$.

The complexity c of the sequence 0010 can, e.g., be determined via the following steps.

(1) The first digit has always to be inserted $\rightarrow 0 \cdot$.

(2) $S = 0$, $Q = 0$, $SQ = 00$, $SQ\pi = 0$, $Q \in v(SQ\pi) \rightarrow 0 \cdot 0$.

(3) $S = 0$, $Q = 01$, $SQ = 001$, $SQ\pi = 00$, $Q \notin v(SQ\pi) \rightarrow 0 \cdot 01 \cdot$.

(4) $S = 001$, $Q = 0$, $SQ = 0010$, $SQ\pi = 001$, $Q \in v(SQ\pi) \rightarrow 0 \cdot 01 \cdot 0$.

Now c is equal to the number of parts of the string that are separated by dots, i.e., $c = 3$. Figure 1 shows the flow diagram of a computer program that determines $c(n)$ for a given string of length n.

Finally we consider two analytic results for $c(n)$. It is well known that the rationals in the interval [0,1], say are of measure zero. This implies that for almost all numbers

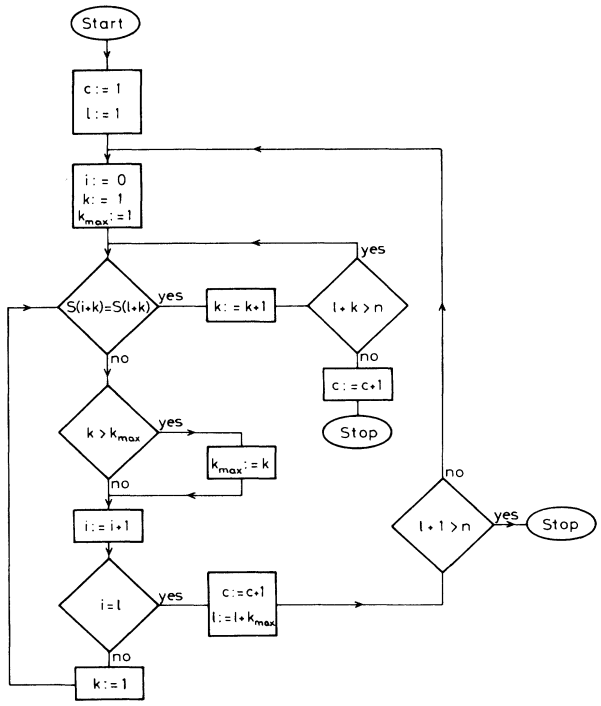

FIG. 1. Diagram for the algorithm to calculate the complexity $c(n)$ of a string of length n.

in [0,1] (i.e., for all irrationals) the string of zeros and ones which represents their binary decomposition is not periodic. Therefore we expect that almost all strings which correspond to a binary representation of a number $x \in [0,1]$ should be random and have maximal complexity. It has indeed been shown by Lempel and Ziv that for almost all $x \in [0,1]$ the complexity $c(n)$ (of the string which represents the binary decomposition) tends to the same value:

$$\lim_{n \rightarrow \infty} c(n) = b(n) \equiv n / \log_2 n \ . \tag{1}$$

$b(n)$ gives, therefore, the asymptotic behavior of $c(n)$ for a random string and we will often normalize $c(n)$ via this limit, i.e., we will only consider the finite ratio $0 \leq c(n)/b(n) \leq 1$, instead of $c(n)$.

If we have a string from a random source where, however, the probability p of finding 1 is different from 0.5, we could expect that its complexity should be smaller than that of a random string with $p = 0.5$. It has also been shown by Lempel and Ziv that in this case $c(n)$ tends to

$$\lim_{n \rightarrow \infty} c(n) = hb(n) \ , \tag{2}$$

where $h = -[p \log_2 p + (1-p) \log_2 (1-p)] \leq 1$ is the so-called source entropy which has its maximum at $p = 0.5$. This means that one can decide [by determining p and $h(p)$] whether deviation of $\lim_{n \rightarrow \infty} [c(n)/b(n)]$ from 1 is only due to the fact that the source entropy differs from 1 (and the string is still as random as possible) or

due to the formation of a pattern in the string (if $\lim_{n\to\infty}[c(n)/b(n)]<h$).

These examples show that the complexity measure $c(n)$ can be calculated easily and its numerical value agrees with our intuitive notion of complexity. In the following we will investigate what can be learned about the behavior of dynamical systems from a computation of $c(n)$.

III. ANALYSIS OF TIME SERIES GENERATED BY SOME ONE-DIMENSIONAL MAPS

In this section we analyze strings that are generated by a random-number generator and by simple one-dimensional maps, in order to answer the following questions.

(1) How fast does the complexity $c(n)$ of a string produced by a random-number generator converge to the asymptotic value $n/\log_2 n$? This determines the minimum length $n_{\min}$ needed to approach the asymptotic value of $c(n)n/\log_2 n$ if one analyzes a general string.

(2) How does the computer time that is needed to compute $c(n)$ depend on n? This gives an estimate of the largest length $n_{\max}$ which can be analyzed by our method.

(3) How is the complexity of a given string related to the (largest) Liapunov exponent λ of the corresponding time series, i.e., what is the relation between the complexity measured by $c(n)$ and the conventional measure λ of chaos?

(4) How is the period-doubling route to chaos[1] mirrored in the complexity of the corresponding time series? This should tell us how well different nonchaotic time patterns are mirrored in $c(n)$.

Let us begin with the answers to the first question. Figure 2 shows the normalized complexity $c(n)/b(n)$ of a string $s_1 s_2 \cdots s_n$ that has been generated by a pseudo-random-number-generator.[12] (The program produced random numbers $x_i \in [0,1]$ and we took $s_i=0$ if $x_i \le 0.5$ and $s_i=1$ otherwise.) It follows that the limit $\lim_{n\to\infty}[c(n)/b(n)]=1$ is already reached to within 5% for $n>n_0=10^3$. This means that one needs at least strings with a length $n>10^3$ in order to obtain results for $c(n)/b(n)$ that are independent of n. For $n>10^3$ one can compare strings of different length and it makes sense to speak of a complexity per digit. Figure 3 shows that the computer time $I(n)$ (Ref. 13) needed to analyze a random string of length n increases approximately as n^2. This can be understood crudely by the argument that the computer program for $c(n)$ has to compare, at most, each number in the string with the rest of the string in order to find out which new digit has to be added, i.e., one has approximately n^2 elementary operations. For a string of only zeros or ones [i.e., for $c(n)=2$] the computer time needed is of course of order n. This means one has $n<I(n)<n^2$ which implies that one can analyze, e.g., on a Univac 1091 within one minute a string of length $n\simeq 10^5$.

Next we analyzed the complexity $c(n)$ of strings $s_1 s_2 \cdots s_n$ generated by the piecewise linear map

$$x_{i+1}=rx_i \bmod 1; \quad x_i \in [0,1] \tag{3}$$

as a function of the control parameter r which is for this example related to the Liapunov exponent λ via $\lambda=\log_2 r$. Again we used $s_i=1$ if $x_i>x_m$ [where $x_m=(1/n)\sum_{i=1}^{n} x_i$ is the mean value of x_i] and $s_i=0$ otherwise. In order to avoid computational artifacts we added to Eq. (3) a small random-noise term $\sigma\xi_n$ where the ξ_n were homogeneously distributed in $0<\xi_n<1$ and $\sigma=10^{-6}$. Figure 4 shows that $c(n)$ increases monotonically with r, i.e., with the Liapunov exponent λ until it saturates at $r=2$, i.e., $\lambda=\log 2$. This can be understood as follows. For $r=1$ Eq. (3) is the identity. All x_i are equal and we obtain a string which contains only zeros and therefore has the complexity $c(n)=2$.

For $r=2$ the generated string is just the binary representation of the initial point x_0 which is with probability 1 an irrational number such that the string has the complexity of a random number $\lim_{n\to\infty} c(n)=n/\log_2 n$. Between $r=1$ and $r=2$ $c(n)$ increases with r because one interpolates between $c(n)=2$ and $c(n)=n/\log_2 n$. Since $n/\log_2 n$ which is reached at $r=2$, is already the largest possible value of $c(n)$, the complexity remains constant for $r>2$.

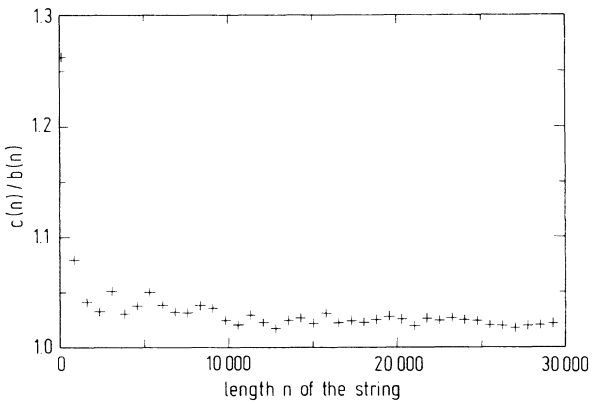

FIG. 2. The normalized complexity measure $c(n)/b(n)$ vs the string length n for random 0-1 strings.

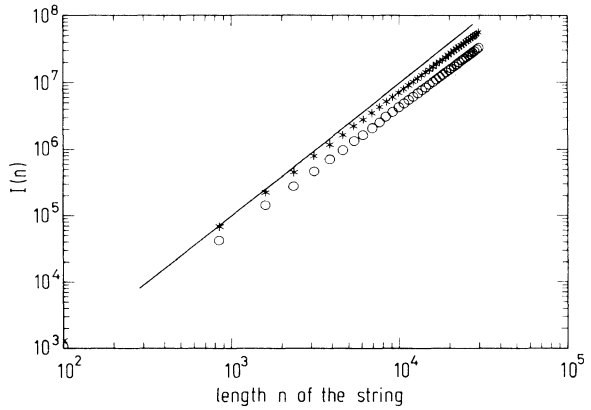

FIG. 3. The computer time $I(n)$ needed to calculate the complexity $c(n)$ of random strings of length n with (*) $h=1$ and (○) $h=0.5$. The solid line denotes $I(n)\sim n^2$.

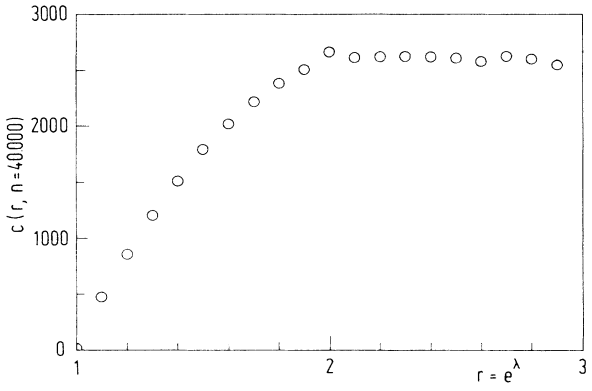

FIG. 4. The normalized complexity measure $c(n)/b(n)$ vs the control parameter $r=e^{\lambda}$ for 0-1 strings given by the noisy piecewise linear map.

Figure 5 shows the bifurcation diagram, the Liapunov exponent and the complexity $c(n)$ as a function of the control parameter $3 \leq r \leq 4$ for the logistic map

$$x_{i+1}=rx_i(1-x_i);\ x_i \in [0,1]\ . \tag{4}$$

The string $s_1 s_2 \cdots s_n$ was generated by taking $s_i=0$ if $x_i < 0.5$, $s_i=1$ otherwise, i.e., for superstable fixed points of the m-fold-iterated Eq. (4) these strings represent just the Metropolis-Stein-Stein[4] (MSS) sequences with $L \longleftrightarrow 0$ and $R \longleftrightarrow 1$. The lengths of the MSS sequences increase with r as one approaches r_∞ from below. This is nicely mirrored by the increasing values of $c(n)$. For values above $r=r_\infty$, $c(n)$ shows minima at the windows, as expected and reaches again its largest possible value at $r=4$ where the Liapunov exponent of the logistic map has also its maximum.

This means that three different kinds of behavior of $c(n)$ have been found for the logistic map.

(i) For periodic orbits, $c(n)$ reaches a finite value for large n.

(ii) At $r=r_\infty$, $c(n)$ diverges with n, but the normalized complexity $c(n)/b(n)$, i.e., the complexity per digit goes to zero.

(iii) For chaotic orbits the normalized complexity $c(n)/b(n)$ asymptotically reaches a finite positive value.

All marginally stable periodic orbits including $r=r_\infty$ have the Liapunov exponents $\lambda=0$. However $c(n)<\infty$ for $r<r_\infty$ and $c(n)\to\infty$ for $r\to r_\infty$. This means that the complexity c is a more precise measure than the Liapunov exponent for characterizing order or disorder.

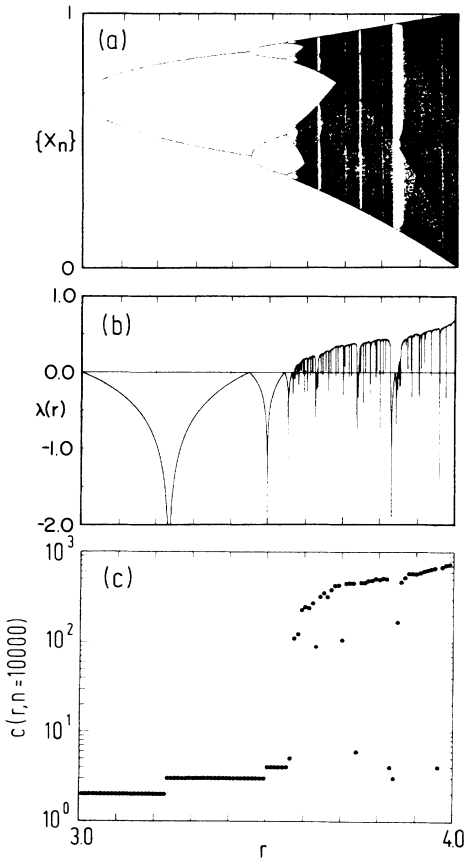

FIG. 5. (a) Attractor; (b) Liapunov exponent; and (c) complexity vs control parameter r for the logistic map.

IV. PATTERN SELECTION IN SIMPLE CELLULAR AUTOMATA

In this section we want to investigate the question of how the selection of spatial patterns is mirrored in the complexity $c(n)$. We take as simple examples several one-dimensional, two-state (0,1), nearest-neighbor cellular automata[5] and start at time $t=0$ with random strings. One would conclude at first sight that pattern selection always takes place if the complexity $c(n)=n/\log_2 n$ of the initial spatial string of length n decreases with time because a random string has maximum complexity and reduction in complexity means more order, i.e., pattern formation. However it could happen that the initial value

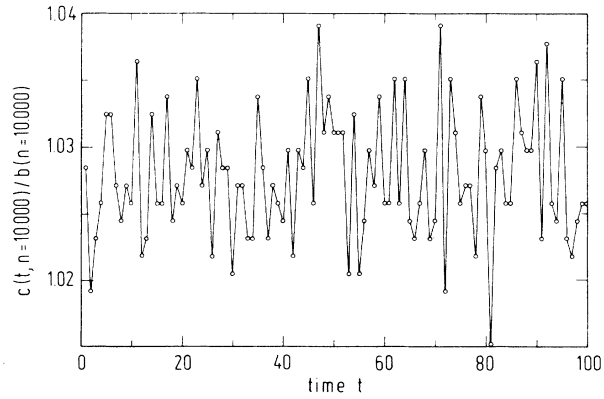

FIG. 6. The normalized complexity measure $c(t,n=10\,000)/b(n)$ of the 0-1 string given by the CA evolution with rule 90.

TABLE I. Summary of the CA results. $p_\infty(1)$ denotes the average fraction of sites with value 1 for $t\to\infty$ taken from Ref. 5. $p_{100}(1)$ denotes our observed fraction of sites with value 1 for $t=100$. h_∞ and h_{100} denote the normalized entropy calculated from $p_\infty(1)$ and $p_{100}(1)$, respectively. The results concerning the minimal number of nodes in deterministic finite automata (DFA) representing regular languages defined by the CA rules are taken from Table 1 of Ref. 9.

Rule[a]	Figure	Expected $p_\infty(1)$	Observed $p_{100}(1)$	Expected h_∞	Observed h_{100}	Pattern selection	Nodes of minimal DFA $t=1$	$t=2$	$t=3$
18	7	0.25	0.25	0.83	0.83	yes	5	47	143
22	8	0.35	0.35	0.93	0.93	yes (two time scales)	15	280	4506
90	6	0.5	0.5	1	1	no	1	1	1
122	analog to 7	0.25	0.5	0.83	1	yes	15	179	5080
126	analog to 7	0.25	0.5	0.83	1	yes	3	13	107
150	analog to 6	0.5	0.5	1	1	no	1	1	1

[a]Reference 5.

$c(n)=n/\log_2 n$ decreases only to $hn/\log_2 n$ with $h<1$. That means that the only thing which changes during the time evolution of the CA is the probability of finding 1 (p becomes different from 0.5). Otherwise the string remains completely random, i.e., there is no order within the 1's and 0's. We say that a CA shows pattern selection only if $c(n)$ decreases below $h\log_2 n/n$, i.e., if the string becomes ordered (a mere change of p is not enough).

Figures 6–8 show the time evolution of $c(n=10^4)$ for several legal class-3 CA's. Our observations are summarized in Table I. They agree—with one exception— with previous results obtained by other methods.[8,9] The exception is the CA which corresponds to rule 22. It displays two time scales in the development of $c(n)$. In the first four time steps $c(n)$ decreases rather rapidly until p takes the value 0.35, then the decay slows down considerably (see Fig. 8). This result is new.

Indeed Grassberger[14] found that rule 22 shows a totally different behavior compared to the other legal class-3 two-state CA with nearest-neighbor interaction by mapping the CA onto one-dimensional maps. This result concerned the behavior for all times. Wolfram[9] introduced an algorithmic complexity (AC) measure to investigate the time evolution of the complexity of a CA. The AC for the CA at a specific time is given in terms of the minimal graph of the regular language defined by a CA. Because of the limitation of available computer time Wolfram was only able to calculate the AC for three time steps. Grassberger followed this idea and calculated the AC and a similar measure [the set complexity (SC)] for $t=1,2,3$. But this method yielded no evidence for his conjecture that rule 22 is the most complex among those mentioned above. Our approach clearly reveals the reason for this. The difference between rule 22 and the other rules shows up only if more than four time steps are investigated.

A statistical approach to the investigation of pattern formation which differs from our approach is to calculate block entropies, i.e., probabilities for the occurrence of substrings of specific length in a string. Block entropies were used in Ref. 8 to define the so-called effective measure complexity (EMC) denoted $C(n)$. In the case of dynamical systems block entropies can be viewed as approximants of the Kolmogorov entropy[15] and the EMC can be understood as measuring how fast these approximants converge to the Kolmogorov entropy.

Grassberger investigated with the EMC the spatiotemporal behavior of some CA's. For two-state CA's the determination of block entropies for spatial, temporal, or spatiotemporal patterns of length n requires the computation of 2^n probabilities which correspond to the possible

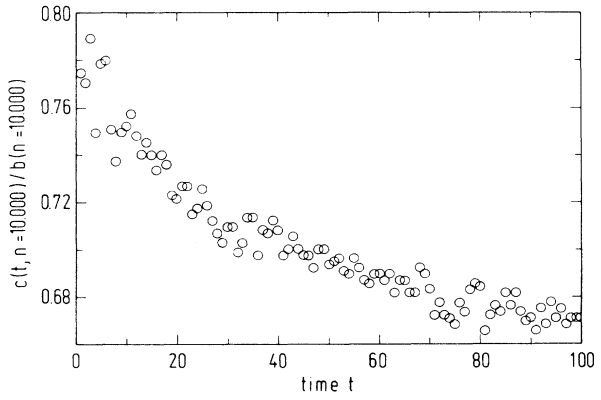

FIG. 7. Same as Fig. 6 with rule 18.

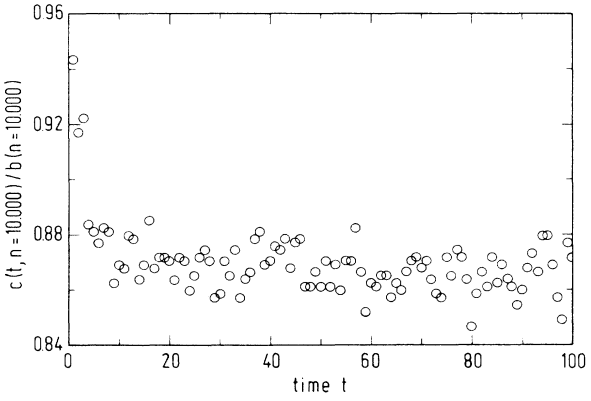

FIG. 8. Same as Fig. 6 with rule 22.

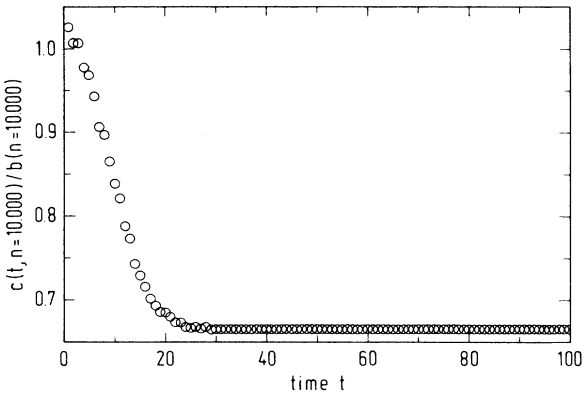

FIG. 9. The normalized complexity measure $c(t,n=10\,000)/b(n)$ for spatial strings given by the time evolution of coupled logistic maps. $r=3.62$, $\varepsilon=0.1$.

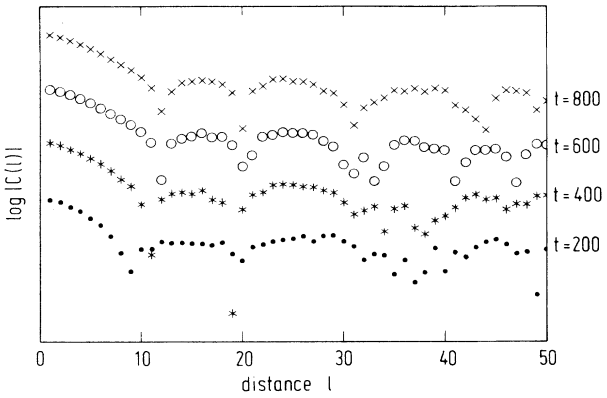

FIG. 11. The spatial autocorrelation function

$$C_t(l)=\frac{1}{n^2}\sum_i [x_t(l+i)-\bar{x}][x_t(i)-\bar{x}]\ ,\bar{x}=\frac{1}{n}\sum_i x_t(i)$$

for $n=10.000$ coupled logistic maps for four different times. $r=4.0$, $\varepsilon=0.1$.

configurations of a chain of length n. Thus the computer time needed for a calculation of $C(n)$ increases like 2^n as compared to n^2 for $c(n)$. If the dynamics of the CA does not generate long-range spatial correlations it is enough to analyze $C(n)$ for a short substring (with n of the order of the correlation length) and $C(n)$ can be determined with a reasonable amount of computer time. But for long correlated strings ($n \gg 1$) $C(n)$ becomes difficult to compute whereas $c(n)$ can still be calculated with reasonable effort.

V. COUPLED LOGISTIC MAPS

Coupled-map lattices provide a convenient tool for investigating the development of temporally induced spatial patterns.[6,7] In this section we calculate the time development of the spatial complexity $c(n)$ for a one-dimensional chain of elastically coupled logistic maps:

$$x_{t+1}(i)=\frac{1}{1+2\varepsilon}[f(x_t(i))+\varepsilon f(x_t(i-1))+\varepsilon f(x_t(i+1))] \tag{5a}$$

$$f(x)=rx(1-x);\quad x\in[0,1];\quad t,i \text{ integers}, \tag{5b}$$

where ε measures the strength of the spatial coupling and r the "strength" of temporal disorder generated by the individual logistic map. We used chains of length $n=10^4$ with cyclic boundary conditions, and started from a set of random values for $\{x_0(i)\}$. In order to reduce $x_t(i)$ to two values we took $s_t(i)=1$ if $x_t(i)$ was above the spatial average $\bar{x}_t=(1/n)\sum_{i=0}^{n-1} x_t(i)$ and $s_t(i)=0$ otherwise. Figures 9 and 10 show the time development of $c(n)$ for $r=3.62$. (i.e., before the last band merging point) and for $r=4$ (where the Liapunov exponent of the individual map has its largest possible value). In both cases c ($n=10^4$) decreases with time to a stable value. This indicates a pattern selection process which leads to a stable dynami-

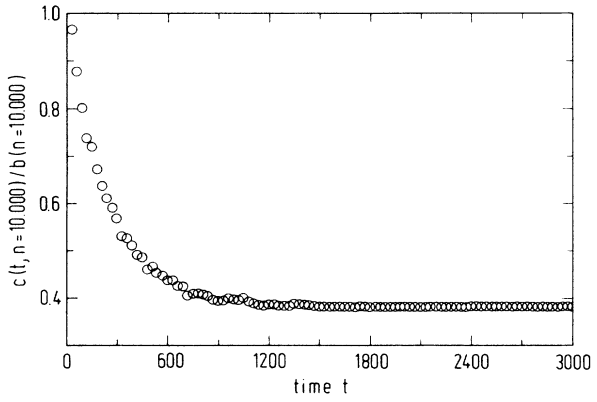

FIG. 10. Same as Fig. 9; $r=4.0$.

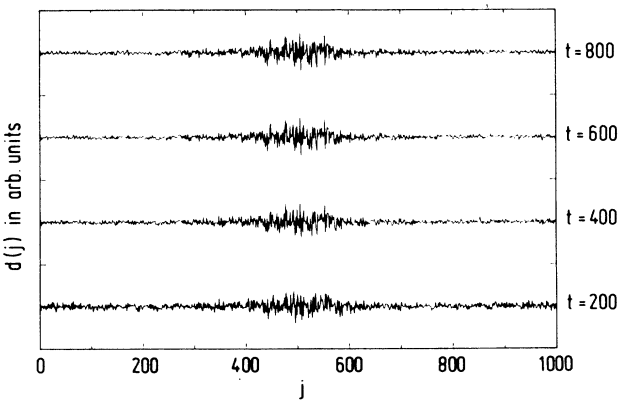

FIG. 12. The coefficient d_j where $d_j=a_j$ for $1<j<500$ and $d_j=-b_j$ for $501<j<1000$ of the fast Fourier transform $\hat{x}_t(j)=\sum_{k=1}^{n} x_t(k)e^{ikj/2\pi}$, $\hat{x}_t(j)=a_j+ib_j$ of $n=1000$ coupled logistic maps. Times and parameters are as in Fig. 11.

cal equilibrium which is probably characterized by a dynamical kink structure with a conserved number of kinks.[7] However the time scale differs in the two cases by three orders of magnitude. It is important to note that this pattern selection process leaves no trace at all in more conventional measures of spatial pattern selection such as the spatial correlation function or its Fourier transformation (see Figs. 11 and 12). This shows that $c(n)$ can be used generally (and not only for CA's) as a convenient tool to obtain new results about spatial pattern formation. But it is, of course, of vital importance to encode the variable (which varies in space and time) efficiently, i.e., without loss of important information into a finite alphabet. This problem will be considered in a forthcoming paper.[16]

VI. CONCLUSIONS AND DISCUSSION

It has been shown that the complexity measure $c(n)$ allows an analysis of spatiotemporal patterns which indicates (i) whether a dynamical equilibrium state is reached and what its complexity is as compared to a random pattern, (ii) how this state is reached in the course of time, and (iii) whether true pattern selection takes place or only the true source entropy shows up. This analysis concerns only the pattern in real space and time but not the motion in phase space, i.e., it tells nothing about the structure of a possible strange attractor of the system (as do other methods).[2]

Another complexity measure, the EMC, was proposed recently by Grassberger. It was discussed briefly in Sec. III in connection with the characterization of CA's. Let us summarize the most important differences to our measure: EMC works best for "strongly chaotic" systems with short correlation length and is less suited for an analysis of long-range patterns. $c(n)$ is suited best to detect order, i.e., pattern selection, in long chains and does not discriminate very sharply among different strengths of chaos in short chains.

It should also be noted that it is easier to compute $c(n)$ via the algorithm shown in Fig. 1 than to determine the probabilities which enter $C(n)$. Both methods "suffer" from the problem that before they can be applied the measuring variable, space, and time all have to be discretized appropriately. For the case of $c(n)$ we will discuss the influence of different partitioning on the results, as well as an extension of our method to higher dimensional patterns in a future publication.

ACKNOWLEDGMENTS

We thank Dr. S. Grillenberger for calling our attention to the work of Lempel and Ziv and the Sonderforschungsbereich 65 Frankfurt/Darmstadt for financial support. Furthermore, we thank Professor L. Hirst and K. Pawelzik for carefully reading the manuscript.

[1]H. G. Schuster, *Deterministic Chaos* (Physik Verlag, Weinheim, 1984).
[2]T. C. Halsey, M. H. Jensen, L. P. Kadanoff, I. Procaccia, and B. I. Shraiman, Phys. Rev. A **33**, 1141 (1986).
[3]A. Lempel and J. Ziv, IEEE Trans. Inf. Theory **IT-22**, 75 (1976).
[4]M. Metropolis, M. L. Stein, and P. R. Stein, J. Combinatorial Theory (A) **15**, 25 (1973).
[5]S. Wolfram, Rev. Mod. Phys. **55**, 601 (1983); N. H. Packard, and S. Wolfram, J. Stat. Phys. **38**, 901 (1985).
[6]F. Kaspar and H. G. Schuster, Phys. Lett. **113A**, 451 (1986).
[7]I. Waller and R. Kapral, Phys. Rev. A **30**, 2047 (1984); K. Kaneko, Prog. Theor. Phys. **72**, 480 (1984); **74**, 1033 (1985).
[8]P. Grassberger (unpublished).
[9]S. Wolfram, Commun. Math. Phys. **96**, 15 (1984).
[10]A. N. Kolmogorov, Probl. Inf. Transmission **1**, 1 (1965).
[11]R. J. Solomonoff, Inf. Control. **7**, 224 (1964); P. Martin-Löf, *ibid.* **9**, 602 (1966); G. J. Chaitin, J. Assoc. Comput. Machinery **13**, 547 (1966).
[12]The subroutines G05CBE and G05CAE from the NAG library of the Numerical Algorithm Group Ltd., Mark 11, Oxford (1984) were used.
[13]As the measure I for the computer time we took the number of comparisons of digits needed to obtain c.
[14]P. Grassberger, Physica D **10**, 52 (1984).
[15]J. P. Eckmann and D. Ruelle, Rev. Mod. Phys. **57**, 617 (1985).
[16]F. Kaspar and H. G. Schuster (unpublished).