Complex Systems **2** (1988) 269–303

Complexity of Forecasting in a Class of Simple Models

Domenico Zambella
C.N.R. Istituto per lo Studio della Dinamica, delle Grande Masse, 1364 S. Polo, I-30125 Venezia, Italy

Peter Grassberger
Physics Department, University of Wuppertal, D-56 Wuppertal, Gauss-Str. 20, West Germany

Abstract. We deal in this paper with the difficulty of performing optimal or nearly optimal forecasts of discrete symbol sequences generated by very simple models. These are spatial sequences generated by elementary one-dimensional cellular automata after one time step, with completely random input strings. They have positive entropy and thus cannot be entirely predicted. Making forecasts which are optimal within this limitation is proven to be surprisingly difficult. Scaling laws with new anomalous exponents are found both for optimal forecasts and for forecasts which are nearly optimal.

The same remarks hold not only for forecasting but also for data compression.

1. Introduction

One of the basic aspects of the theory of deterministic chaos is the fact that very simple systems can behave in a way which is situated between order and chaos: while complete prediction of its behavior is impossible, one can improve forecasts by knowing sufficiently well the system and its history.

A quantitative measure of the impossibility of perfect forecasts is the metric (Kolmogorov-Sinai) entropy h [1]. Assume we have observed a system with some measuring device Γ (which has some finite resolution), at all times $\leq t$. Then we cannot completely predict what the same device Γ will measure at $t + \tau$, $\tau > 0$. Instead, there will be a gap $\leq h\tau + 0(\tau^2)$ between the information obtainable with Γ and the information of the *best* forecast based on measurements with Γ. If Γ is sufficiently fine that this bound is indeed saturated, it is called a generator [2].

The above argument does not tell us at all how hard it actually might be to perform the optimal forecast, i.e. the forecast which leaves the minimal uncertainty for any given Γ (that is, for any given discrete encoding of the trajectory). This difficulty can vary greatly from case to case, and it need not at all be related to h. Take, e.g., the quadratic map

$$x_{n+1} = 2 - x_n^2 \tag{1.1}$$

Here, a measurement of the sign of x is a generator, and the optimal forecast for it is trivial: since all sequences of signs appear with equal probability [3], a mere guess is already the optimal forecasting. On the other side, consider the quadratic map at the infinite bifurcation point (Feigenbaum point [4]),

$$x_{n+1} = 1.401155 - x_n^2 \tag{1.2}$$

There, forecasting the sign of x can always be slightly improved by storing a longer record of signs in memory. Thus, although the entropy of this string is zero, an *optimal* forecast is never possible [5].

The difficulty of performing some task is usually called its complexity [6]. Thus the Kolmogorov-Chaitin [7,8] (or algorithmic) complexity of a symbol sequence of length N is essentially the length of the shortest computer program needed to generate the sequence, divided by its length N. In cases where the sequence is an encoding of the trajectory of a dynamical system, this is just h: the difficulty of specifying the entire sequence is just proportional to the amount of information which has to be given in order to specify it, and this is proportional to h for long times.

But specifying a long symbol sequence is not the typical task associated with a dynamical system. Apart from deducing the underlying equations of motion and its parameters, the typical task is forecasting. We propose therefore to call *forecasting complexity of a dynamical system* the difficulty involved in forecasting it to the best possible extent. It was proposed in reference [5] that this is closest to everyone's naive notion of complexity of a pattern. More precisely, we have the following computational model in mind:

> We first encode the system by a discrete symbolic dynamics, i.e. we will deal in the following only with symbol sequences $S = s_0, s_1, \ldots$).
>
> We then assume that we have had the opportunity to observe S up to time n, and we are asked to forecast s_{n+1}. After having done this, we are told the actual s_{n+1}, and we go on forecasting s_{n+2}, etc.

Analogous to the case of complexity of computing functions, we could distinguish between space and time complexity [6], depending on whether we consider limitations in storage or in CPU time as more important. We shall do neither. Instead, we shall take as our primary measure the average amount of (Shannon) information about the past sequence which has to be kept at any particular time n (see equation (3.8) below for a formal definition). This was called "true measure complexity" in reference [5]. Notice that this is inbetween space and time complexity: by concentrating on the *average* information instead of on the *maximal* storage needed, we are also sensitive on how *often* this information is fetched into fast memory. Thus our measure seems more appropriate than the common worst-case measures, in situations where one has cheap slow and expensive fast memory, and where one is sharing resources in a computer or in a network of computers with other tasks. In addition, we will however also deal with space complexity in the restricted sense of the maximal (instead of average) amount of (algorithmic) information about the past which has to be stored for an optimal forecast.

In the following, we shall employ these ideas in a class of extremely simple models. In these models, the symbols are just bits, $s_n \in \{0, 1\}$. They can be understood as "outputs" of the expanding map

$$x_{n+1} = 2x_n \bmod 1 \tag{1.3}$$

with s_n depending only on the signs of $x_n - \frac{1}{2}$, $x_{n-1} - \frac{1}{2}$, and $x_{n-2} - \frac{1}{2}$ according to some fixed rule. Alternatively, we can consider (s_n) as the spatial string of spins in a one-dimensional cellular automaton with 3-spin neighborhood (called "elementary" by S. Wolfram [9]) after one single iteration. The initial configuration of the spins is assumed random.

There are 256 such models, and we shall use Wolfram's nomenclature for them [9]. Some of them are completely trivial (like e.g. the identity rule #204 or the extinction rule #0); others are more interesting, but all of them seem fairly elementary indeed at first sight. All non-trivial rules lead to sequences with positive entropy.

In striking contrast to the simplicity of these models and all its other aspects, the complexity of forecasts is in some of them extremely large (the worst-case complexities are even infinite in many cases). This is related to the fact that although these sequences are emitted by Markov sources, they are themselves not Markov sequences. Indeed, it is well known that sequences which are not Markovian can be very hard to forecast. The virtue of our models is that due to their simplicity many aspects can either be studied exactly or at least can be dealt with numerically to great precision. Thus, we can make quantitative statements concerning the difficulties of optimal or near-optimal forecasts. We believe that there are not many other systems where this can and/or has been done.

The plan of this paper is as follows: in section 2, we shall introduce the class of models we shall work with. In section 3, we then show how to perform optimal forecasts for them most efficiently, and we study some properties of the resulting algorithms. Those properties relevant to the more complex rules are treated in section 4. In section 5, we demonstrate that these methods are also efficient for estimating entropies and forecasting complexities. We also discuss there how they are related to Markov approximations. Finally, in section 6 we discuss other ways of making nearly optimal forecasts and present some scaling laws for the limit when the admitted errors tend to zero.

We should mention that we could have concentrated also on data compression (i.e., on coding the information missing for the forecast in the shortest way) instead of on forecasting. It is obvious that the difficulties there are related to the ones studied here: if one wants to compress a sequence maximally, one has to use for each new symbol all information obtainable from the past, just as in forecasting. A relevant complexity measure would then be the average delay between the uncoded sequence and the sequence encoded and then decoded back. Again, one could take the worst case (i.e., the maximum delay), but we would instead consider the average delay as more relevant for physics.

2. The models

Consider a random string of bits $T = \dots, t_{n-2}, t_{n-1}, t_n, \dots$ with $\text{prob}(t_i = 0) = \text{prob}(t_i = 1) = \frac{1}{2}$ for all i. The string S is then obtained as

$$s_n = F(t_{n-1}, t_n, t_{n+1}) \tag{2.1}$$

where $F(t, t', t'')$ is any one of the 2^8 possible Boolean functions of 3 arguments. Wolfram's [9] notation for these functions is based on writing the input triples in descending order, $(tt't'') = (111), (110), (101), \dots (000)$, and reading the string of 8 output bits as the binary representation of a number between 0 and 255.

It is known [10] that for each "rule" (i.e., for each function F) the set of all possible strings S forms a regular language [6]. This means that for each rule one can find a finite directed graph such that parsing the string corresponds to walking on the graph. The sizes M of these graphs (i.e., the number of nodes) vary between 1 (for trivial rules) and 15. The graphs for rules 18, 76, and 22 are e.g. given in figures 1a, b, and c.

At any time, one has to remember just the number of the present node if one wants to recognize grammatically wrong continuations as such. This means that the information needed to exclude wrong continuations of S is represented at any time by a number $\leq M$, and can thus be stored in $1 + \log_2 M$ bits. Therefore, $\log_2 M$ is called the "regular language complexity" (RLC) in reference [10].

In some cases (see, e.g., figure 1b) the graph contains a transient part. After some initial phase (of unknown length) the transient part is left and not re-entered. The relevant graph is then only the non-transient part of the total graph, and the effective complexity is smaller than during the transient period. During the latter, the entropy of the string S can be positive, as illustrated by figure 1b.

Actually, one can also study the average Shannon information about the past history which has to be stored at any time for a correct parsing. Call p_1 the probability to be at some given time at node i. This information is then equal to $-\sum_1 p_1 \log_2 p_1$. It is called the "set complexity" (SC) of the set $\{S\}$ in reference [5]. In contrast to the RLC, it is not a purely algebraic concept but depends also on a measure.

We should point out that we still get regular languages if we iterate equation (2.1) K times [10], $s_n^{k+1} = F(s_{n-1}^k, s_n^k, s_{n+1}^k)$ with $s_n^0 \equiv t_n$ and $s_n \equiv$

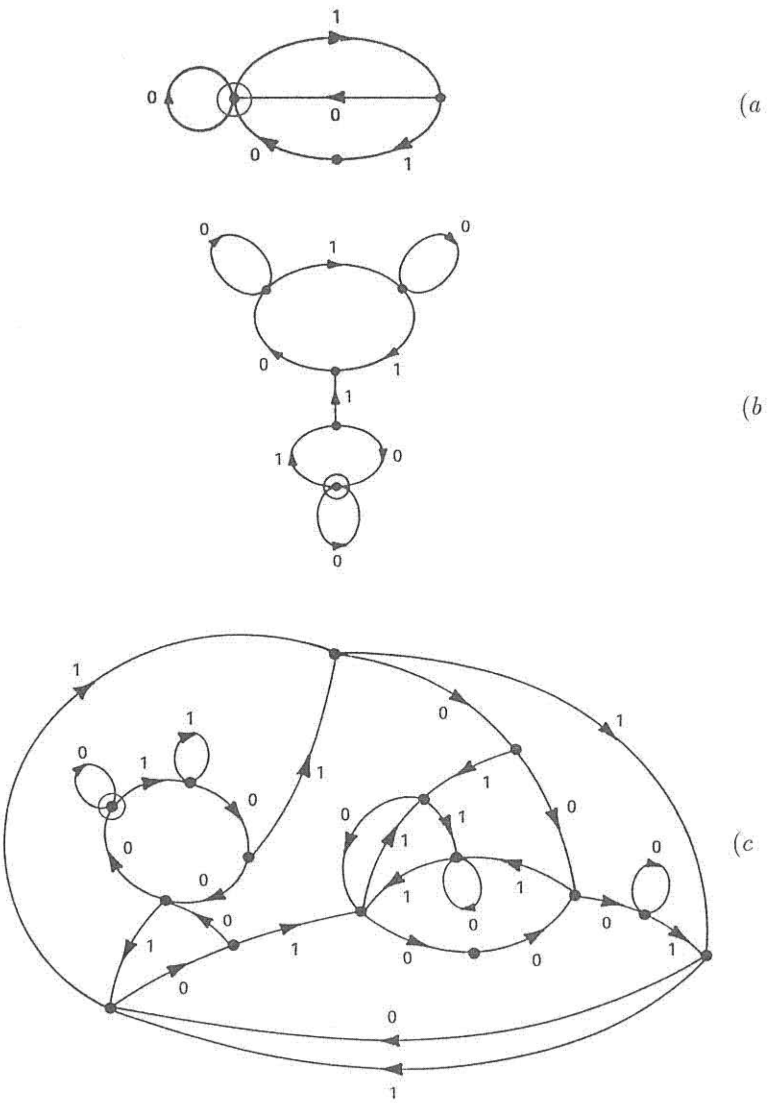


Figure 1: Deterministic graphs specifying the grammars of spatial strings generated by one iteration of cellular automata 76 (panel a), 18 (panel b), resp. 22 (panel c). Each link is labeled by a bit of the output string (from reference [10]).

s_n^k. (This is no longer true in general for the limit sets after an infinite number of iterations [11]). But even after very few iterations, the sizes of the graphs get astronomically large for the more complex rules, preventing their explicit construction and study. This holds also true for the investigations of the present paper: while there would not occur anything new in principle if we would study iterates of equation (2.1), the practical difficulties would be prohibitive already after one more iteration.

The sequences obtained by equation (2.1) have positive entropy for most rules. Except for the simplest cases, exact values of h are not found in the literature, but upper bounds are provided by topological entropies. These are known for all rules [10].

Also known are the lengths of the shortest excluded blocks [10]. While some of the rules lead to finite complement languages, this is not true in general. For rule 22, e.g., the shortest excluded block has length 8, and the language is not of finite complement type.

3. Forecasting S

We come finally to the problem of not only verifying the grammatical correctness of S but of actually forecasting it.

Basically, we proceed as follows: at time n, we have some (partial) knowledge of the input string T up to t_n. It allows us to make a conjecture about the 4 probabilities $P_n(t,t') = \text{prob}(t_{n-1} = t, t_n = t')$, *conditioned* on the output string $s_1 \ldots s_{n-1}$. Using these, and assuming t_{n+1} to be random, we can predict the conditional probabilities $p_n(s) = \text{prob}(s_n = s)$ as

$$p_n(s) = \frac{1}{2} \sum_{t,t',t''} P_n(t,t')\delta[s - F(t,t',t'')] \tag{3.1}$$

where $\delta[i-k]$ is a Kronecker delta.

After having forecasted thus s_n, we are shown its actual value, and we can update our conjecture on T. The new unbiased estimate is

$$P_{n+1}(t',t'') = [2p_n(s_n)]^{-1} \sum_t P_n(t,t')\delta[s_n - F(t,t',t'')] \tag{3.2}$$

In this way we alternate between "conjectures" about the unobserved string T and "forecasts" about the observed string S. The "conjectures" are made only for internal use and are only made as intermediate steps. The procedure starts of course with the conjecture $P_{n=0}(t,t') = \frac{1}{4}$, if the series of observations starts at $n = 0$.

Let us introduce now a new notation. We first write the 4 probabilities $P_n(t,t')$ as elements of a four-dimensional vector,

$$\mathbf{P}_n^{\mathrm{T}} = (P_n(0,0),\, P_n(0,1),\, P_n(1,0),\, P_n(1,1)) \tag{3.3}$$

such that the start vector becomes $P_0^{\mathrm{T}} = 1/4(1,1,1,1)$. Then, equations (3.1) and (3.2) can be written in matrix form as

$$p_n(s) = 2\mathbf{P}_0^T M_s \mathbf{P}_n, \qquad \mathbf{P}_{n+1} = [2p_n(s_n)]^{-1} M_{sn}\, \mathbf{P}_n \tag{3.4}$$

where M_0 and M_1 are the 4×4-matrices

$$M_s = \tag{3.5}$$

$$\begin{pmatrix} \delta[s-F(0,0,0)] & 0 & \delta[s-F(1,0,0)] & 0 \\ \delta[s-F(0,0,1)] & 0 & \delta[s-F(1,0,1)] & 0 \\ 0 & \delta[s-F(0,1,0)] & 0 & \delta[s-F(1,1,0)] \\ 0 & \delta[s-F(0,1,1)] & 0 & \delta[s-F(1,1,1)] \end{pmatrix}$$

with all elements either 0 or 1, and where $p_n(s)$ acts just as a normalizer. Up to the latter, we find thus that each observation of an output bit corresponds to multiplying the conjecture vector by the corresponding matrix. The trajectories of these vectors are obtained by applying the two matrices M_0 and M_1 in the random sequence given by S.

This construction of the set of $\mathbf{P}$-vectors is reminiscent of the limit set of "contraction maps" or "iterated function systems" [12]. The main difference is that in contraction maps one considers all index sequences as allowed, while in our case only those index sequences S are admitted which are output sequences of the cellular automaton.

The difficulty of forecasting S is related to the algebraic properties of the matrices M_s. Forecasting would be easy if there exists a finite set of N vectors $\mathbf{P}$ which is closed under allowed multiplications of M_s and which contains the start vector $\mathbf{P}_0$. Then, there would exist only a finite number of unbiased conjectures about the input string, and from these we would obtain via equation (3.1) only a finite number of forecasts $p(0)$ (remember that $(p(1) = 1 - p(0))$. The amount of information to be stored at any time would be finite and at most equal to $1 + \log_2 N$ bits. But in general, things are much more complicated.

Let us study specific rules in somewhat more detail.

1. Rule 90. There we find

$$M_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad M_1 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{3.6}$$

   and we find $M_0\mathbf{P}_0 = M_1\mathbf{P}_0 = \mathbf{P}_0$. Thus, from no observation we can ever learn anything about the input string, and correspondingly we never can make a better forecast than a pure guess.

2. Besides rules with such completely random output strings, having the trivial closed set $\{\mathbf{P}_0\}$, there are very few other rules with a finite closed set. These are rule 35 and all rules obtained from it by conjugation and reflection [10]. Here, one has

$$M_0 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} \tag{3.7}$$

Applying any grammatically correct sequence to $\mathbf{P}_0$ one finds the set of $\mathbf{P}$-vectors shown in figure 2. The $\mathbf{P}$'s form in this figure the nodes of a directed graph, the links being labeled by the output variables s (i.e., by the indices of M). For convenience, we have multiplied each $\mathbf{P}$ in the figure by the smallest integer which makes its components integer. The rational numbers written next to each node $\mathbf{P}$ are the forecasts $p(1,\mathbf{P})$ associated to $\mathbf{P}$. It is easily checked that these forecasts cannot be obtained from any simpler graph obtained from figure 2 by identifying some groups of nodes. Thus figure 2 is indeed the smallest graph usable for forecasting rule 35. Let us denote by $Q(\mathbf{P})$ the probability of visiting node $\mathbf{P}$ at any given moment when reading in a random input string. These probabilities are easily computed numerically (see also section 5). From them, we can compute the forecasting complexity C, defined as the average Shannon information to be stored during a forecast of the output of a random input string,

$$C = -\sum_{\mathbf{P}} Q(\mathbf{P}) \log_2 Q(\mathbf{P}) \tag{3.8}$$

Numerically, we find $C = 2.10846$ bits.

As can be shown by going through all 256 rules, there does not exist such a finite graph, corresponding to a finite closed set of $\mathbf{P}$-vectors, for any other rule not related to rule 35.

3. A representative of the structurally next complex class is rule 76. Here, one has

$$M_0 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.9}$$

Applying all correct sequences of M_0 and M_1 to $\mathbf{P}_0$, one finds the set of $\mathbf{P}$-vectors shown in figure 3. It forms the vertices of an infinite graph. By construction, the graph is such that all paths on it have images in figure 1b and vice versa. It is easily verified that there is not only an infinite number of nodes in this graph, but that also the number of different values of the forecasts $p(s)$ is infinite. Therefore we find that for this rule (which behaves simply in all other respects) an optimal prediction would indeed need an infinite amount of information to be stored in the worst case.

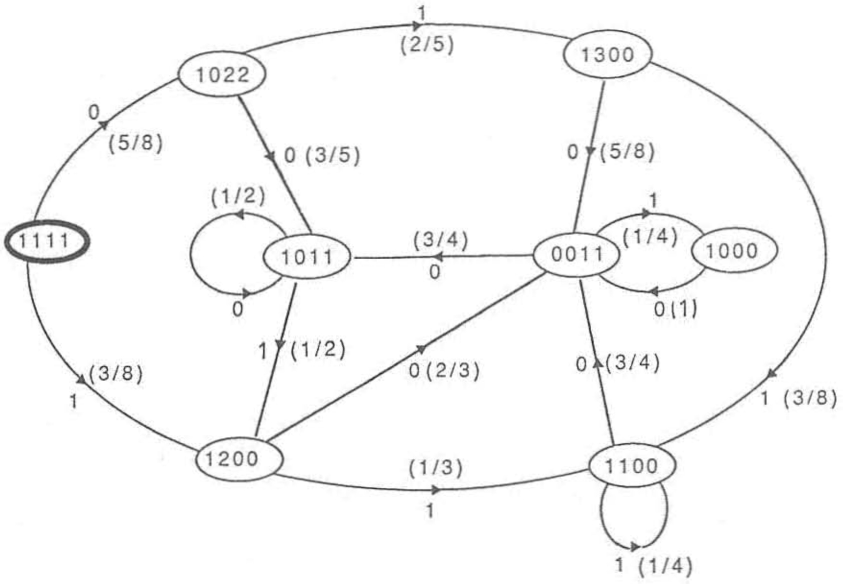


Figure 2: Deterministic graph for forecasting the first iterate of rule 35, with random input strings. As in figure 1, each link is labelled by an output bit. In addition, each node corresponds to one vector **P**, the numbers in each node representing its components. The rational numbers in brackets next to each link are the forecasted probabilities $p(s)$ to branch to this link and not to the other one leaving the same node. The node (1111) is the start node.

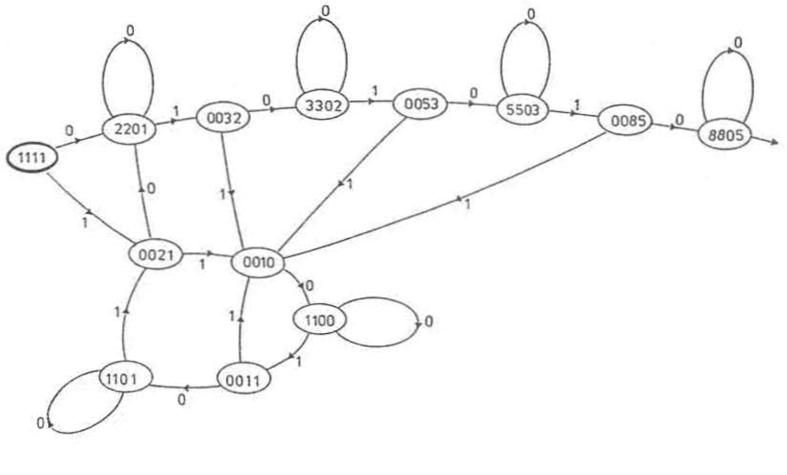


Figure 3: Part of the infinite graph for forecasting the first iterate of rule 76, with random input strings.

Notice that the number of different values of $p(0)$ is a lower limit to the number of vertices of the simplest graph allowing an optimal forecast. An improved lower limit on the size of a finite graph is obtained by a minimization similar to that used in regular formal languages [6]: we first divide the set $\{\mathbf{P}\}$ of nodes into sets such that within each class all nodes have the same $p(0, \mathbf{P})$. Then, we divide each subset further such that within each sub-subset all daughters are in the same subset. Finally, we repeat the last step until no further refinement is obtained. For infinite graphs, a lower bound of the number of nodes a distance n from the start node can be obtained. We just have to define the subsets somewhat differently: in the first step, *each* subset contains *all points* with distance $n+1$, in addition to the points with definite $p(0)$.

Although there is an infinite number of nodes in the graph for rule 76, the graph has a rather simple structure and the values of the predictions converge exponentially to a constant with the node number: for distances $d \geq 5$ from the origin, there is only 1 node per distance. Thus for each node $\mathbf{P}$, the forecast $p(0, \mathbf{P})$ and the probability $Q(\mathbf{P})$ that it is just visited at any given moment are again computable without much effort. Also, the forecasting complexity C defined in equation (3.8) is finite and computable. The result, obtained numerically by the method of section 5, is given in table 1.

4. The next complication is presented e.g. by rule 4. Applying here successively M_0 and M_1 to $\mathbf{P}_0$, we arrive at the graph shown in figure 4. Again, what is shown there is only part of an infinite graph, but this time it contains two branches extending to infinity. One of them is visited again and again, the other is transient: it is only visited if from the very beginning one observes a string of "0". After the first time a "1" is observed, the relevant graph consists of the lower part only. This part is however very simple. Any application of M_0 leads away from the node $\mathbf{P} = (0, 0, 1, 0)$, while any M_1 leads back to this node. This allows one to compute easily all $p(0, \mathbf{P})$ (they converge exponentially fast to 0.122561) and all $Q(\mathbf{P})$, and we obtain $C = 4.3462$ bits.

5. A still higher degree of complication is observed for rule 18. Here again the non-transient part consists of one single branch, but the transient part is now more complex, see figure 5: it has an infinite number of branches. As we are most interested in average quantities (where transients do not contribute) we shall not consider it further. We just mention that the weight $Q(\mathbf{P}(k))$ of that node in the non-transient branch which is a distance k away from the node $\mathbf{P} = (1, 0, 0, 0)$ is equal to $F_k/2^{k+2}$, where F_k is the k^{th} Fibonacci number $F_0 = F_1 = 1$, $F_2 = 2, \ldots$). From this we get that $p(0, \mathbf{P}(k))$ converges for $k \to \infty$ to $(2g)^{-1}$, where $g = 0.618\ldots$ is the golden mean, and that $C = 3.59394$ bits. The growth of the graph is somewhat unusual for rule 18: the number of nodes with a distance $\leq n$ from the start seems to grow $\sim \exp(\text{const.} n^{2/3})$.

6. A representative of the most complex class is rule 22. Part of the infinite graph obtained by applying all grammatically correct sequences of M-matrices to $\mathbf{P}_0$ is shown in figure 6. We have not been able to describe its structure in simple terms. The number of nodes at a distance $\leq N$ from the origin is found by exact enumeration to increase exponentially, as $e^{(0.386\pm.004)n}$. These data are shown in figure 7, together with the analogous data for rule 76. Also shown in figure 7 are the numbers of different forecasts. It seems numerically that they increase with the same exponent, so that the graphs cannot be reduced substantially (remember that we are primarily interested only in $p(0)$, not in $\mathbf{P}$).

A number of rules (e.g. rule 10) yield graphs which superficially look similar to figure 6. However, in the graphs of these rules the number of nodes increases only quadratically with the distance from the origin, instead of exponentially. Thus, though having also an infinite number of branches going off to infinity, these graphs are actually much simpler topologically than figure 6. Essentially, they have a comb-like structure as revealed by drawing only the links leading away from the start node. Moreover, the number of different forecasts increases in these rules only linearly. Nevertheless, the algorithm discussed in (3.c) shows that also these graphs cannot be simplified essentially (the number of nodes of the minimized graphs has to increase $\sim n^2/2$, too).

A complete list of the asymptotic behavior of the node number for all independent rules is given in table 1. Rules not appearing there are not independent, see appendix of reference [13]. The graph sizes are always those of the minimized graphs.

4. Forecasting complex rules

Let us study rule 22 in somewhat more detail. In figure 8 we show the average number $K(N)$ of nodes visited during a forecast of a string of length N. It is a measure of how fast we run into difficulties if we want to forecast optimally a long but finite string. We see a power law,

$$K(N) \sim N^\alpha \tag{4.1}$$

with $\alpha = .837 \pm .010$. Again, we have no theoretical explanation for this numerical result, which was obtained by analyzing ~ 200 random input strings.

During the same simulations, we also estimated the probabilities $Q(\mathbf{P})$ with which each $\mathbf{P}$ occurred, and estimated from this the complexity given by equation (3.8). We found (see figure 9) that it converged only very slowly with N, roughly like

$$C(N) = C + a.N^{-\beta} \tag{4.2}$$

with $C = 9.42 \pm .006$, $a \approx 3.7 \times 10^4$, and $\beta = 0.18 \pm .02$. For such slow

rule #	size	entropy	forecasting complexity	remark
0	1	0	0.	
1	n	.43432	4.95	
2	n	.48517	3.892	
3	n	.69924	3.59394	
4	$2n$	.50765	4.3462	
5	$n^2/4$	.69924	7.19	
6	$\exp(.25n)$	.74259	6.98	
7	$2n$	.85184	2.90325	
8	n	.48517	3.892	same as #2
9	$\exp(.31n)$	.73878	8.79	
10	$n^2/4$	.67787	5.42294	
11	$2n$	.79127	2.52508	
12	n	.67787	2.71147	
13	n	.79418	3.15290	
14	$2n$	.83893	3.37303	
15	1	1.	0.	
16	n	.48517	3.892	same as #2
17	n	.69924	3.59394	same as #3
18	$\exp(n^{2/3})$	.69924	3.59394	
19	n	.69924	3.59394	
20	$\exp(.25n)$	.74259	6.59	
21	$2n$	.85184	2.90325	same as #7
22	$\exp(.39n)$	.8931	9.4	
23	$\exp(.28n)$	.76253	8.58	
24	n	.67787	2.71147	same as #12
25	$\exp(.30n)$	.88947	6.13	
26	$\exp(.35n)$	.89026	8.36	
27	$29n/6$	.86084	4.16365	
28	n	.84962	4.640	
29	$2n$	.86279	3.09285	
30	1	1.	0.	same as #15
32	$2n$	.50765	4.3462	same as #4
33	$\exp(.28n)$	.76253	7.57	
34	n	.67787	2.71147	same as #12
35	8	.87500	2.10846	
36	n	.69924	3.59394	same as #3
37	$\exp(.35n)$	.8957	$\sim$ 9–10	
38	$2n$	.83893	3.10573	

rule #	size	entropy	forecasting complexity	remark
40	exp(.25n)	.74259	6.59	same as #20
41	exp(.32n)	.89319	8.63	
42	n	.83893	3.10573	
43	4n	.86279	4.09285	
44	3n	.83893	3.86839	
45	1	1.	0.	same as #15
46	n	.67787	2.71147	
48	n	.67787	2.71147	same as #12
49	9	.87500	2.10846	
50	n	.84962	4.640	same as #28
51	1	1.	0.	same as #15
52	3n	.83893	3.86839	same as #44
53	4n	.86084	3.64064	
54	exp(.25n)	.87127	6.865	
56	n	.84962	4.640	same as #28
57	3n	.89157	3.98523	
58	4n	.86279	3.88603	
60	1	1.	0.	same as #15
61	exp(.31n)	.88947	6.80	
62	exp(.27n)	.89232	5.963	
64	n	.48517	3.892	same as #2
65	exp(.31n)	.73878	8.85	
66	n	.67787	2.71147	same as #12
68	n	.67787	2.71147	same as #12
69	3n	.79418	3.49270	
70	n	.84962	4.640	same as #28
72	exp($n^{2/3}$)	.69924	3.59394	same as #18
73	exp(.36n)	.88649	9.14	
74	exp(.36n)	.89026	9.46	
76	n	.84962	4.640	same as #28
77	exp(.28n)	.76253	8.58	same as #23
78	2n	.86279	4.40794	
81	2n	.79127	2.52508	same as #11
82	exp(.36n)	.89026	9.46	same as #74
84	n	.83893	3.10573	same as #42
85	1	1.	0.	same as #15
86	1	1.	0.	same as #15
88	exp(.35n)	.89026	8.36	same as #26
89	1	1.	0.	same as #15

rule #	size	entropy	forecasting complexity	remark
90	1	1.	0.	same as #15
92	$4n$	.86279	3.88603	same as #58
94	$\exp(.34n)$	.880	8.89	
96	$\exp(.25n)$	.74259	6.98	same as #6
97	$\exp(.33n)$	.89319	9.20	
98	n	.84962	4.640	same as #28
100	$2n$	.83893	3.10573	same as #38
102	1	1.	0.	same as #15
104	$\exp(.39n)$	.8931	9.4	same as #22
105	1	1.	0.	same as #15
106	1	1.	0.	same as #15
108	$\exp(.25n)$	.87127	6.865	same as #54
110	$\exp(.27n)$	.89232	5.96	same as #62
112	2	.83893	3.37303	same as #14
113	$4n$	.86279	4.09285	same as #43
114	$2n$	.86279	4.40794	same as #78
116	n	.67787	2.71147	same as #46
118	$\exp(.27n)$	.89232	5.96	same as #62
120	1	1.	0.	same as #15
122	$\exp(.34n)$	.880	8.89	same as #94
124	$\exp(.27n)$	.89232	5.96	same as #62
126	n	.69924	3.59394	same as #3
128	n	.43432	4.95	same as #1
130	$\exp(.31n)$	.73878	8.85	same as #65
132	$\exp(.28n)$	.76253	7.57	same as #33
134	$\exp(.33n)$	.89319	9.20	same as #97
136	n	.69924	3.59394	same as #3
138	$2n$	.79127	2.52508	same as #11
140	9	.87500	2.10846	same as #49
142	$4n$	.86279	4.09285	same as #43
144	$\exp(.31n)$	.73878	8.77	same as #9
146	$\exp(.36n)$	.88649	9.14	same as #73
148	$\exp(.32n)$	.89319	8.63	same as #41
150	1	1.	0.	same as #15
152	$\exp(.30n)$	.88947	6.13	same as #25
154	1	1.	0.	same as #15
156	$3n$	.89157	3.89523	same as #57

rule #	size	entropy	forecasting complexity	remark
160	$n^2/4$	.69924	7.19	same as #5
162	$3n$	.79418	3.49270	same as #69
164	$\exp(.35n)$	.8957	$\sim$ 9–10	same as #37
168	$2n$	.85184	2.90325	same as #7
170	1	1.	0.	same as #15
172	$4n$	.86084	3.64064	same as #53
176	n	.79418	3.15290	same as #13
178	$\exp(.28n)$	.76253	8.58	same as #23
180	1	1.	0.	same as #15
184	$2n$	.86279	3.09285	same as #29
188	$\exp(.31n)$	.88947	6.80	same as #61
192	n	.69924	3.59394	same as #3
196	8	.87500	2.10846	same as #35
200	n	.69924	3.59394	same as #19
204	1	1.	0.	same as #15
208	$2n$	.79127	2.52508	same as #11
212	$4n$	.86279	4.09285	same as #43
216	$29n/6$	.86084	4.16365	same as #27
224	$2n$	.85184	2.90325	same as #7
232	$\exp(.28n)$	.76253	8.58	same as #23
240	1	1.	0.	same as #15

Table 1. Asymptotic behavior of forecasting graphs, metric entropy, and forecasting complexity for elementary cellular automata after 1 iteration. The entropies and complexities depend on the input string being completely random. Entropies and complexities are given in bits.

Rules not given in this list can be related to other rules which are in the list by means of table 1 of the appendix of reference [13].

The quoted graph sizes and complexities are those after minimizing as described in section 3c.

Notice that some of the entropies tabulated in the appendix of reference [13] are slightly larger than the values given here. Those values are wrong and should be replaced by the present ones.

By "size" we mean the number of nodes with (topological) distance $\leq n$ from the start node. Only the dominant asymptotic behavior is given. Rules are quoted as "same" if entropies, complexities, and minimal graph sizes are exactly (not only asymptotically) the same.

convergence it is, however, very hard to distinguish numerically between a power and a logarithmic behavior. From the data shown in figure 9 alone,

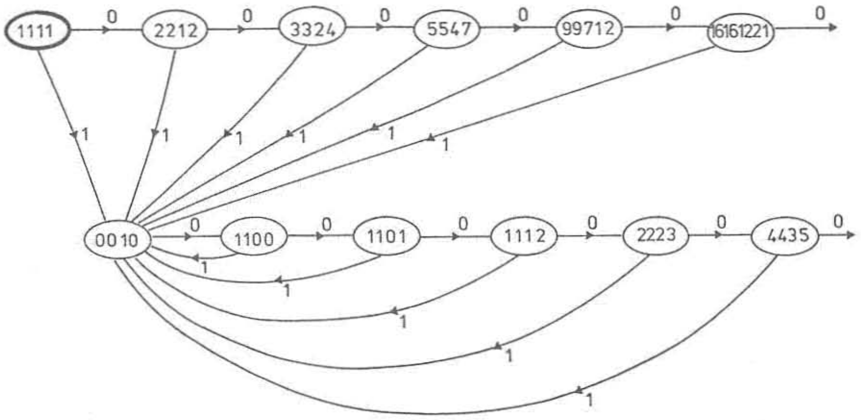


Figure 4: Part of the graph for rule 4.

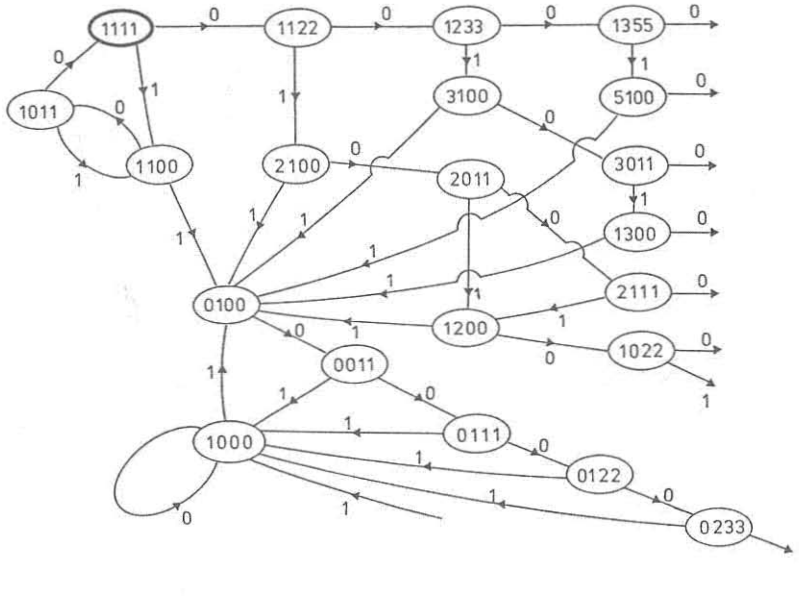


Figure 5: Part of the graph for rule 18.

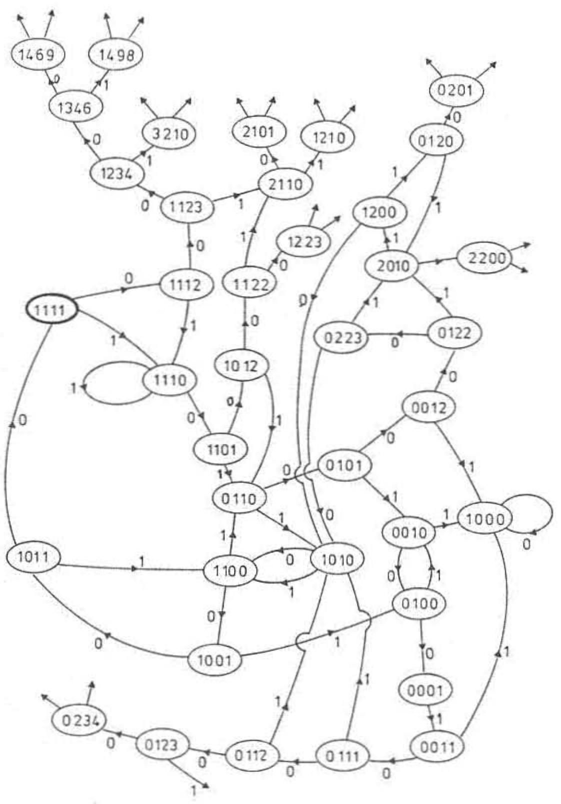


Figure 6: Part of the infinite graph for forecasting rule 22. Notice the much higher complexity compared to the previous graphs.

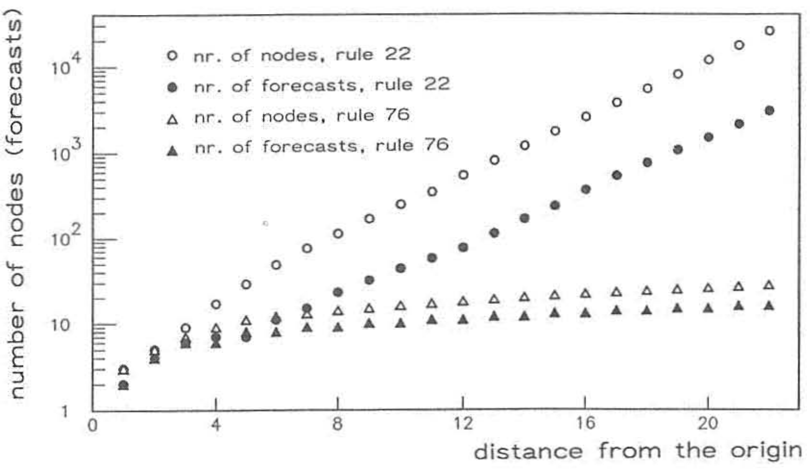


Figure 7: Number of vertices (= different conjecture vectors **P**; open circles) and of different forecasts (heavy dots) for rule 22 plotted versus their distance from the origin (node (1111)) in figure 5, and the analogous result for rule 76.

we could not have excluded a behavior like $C(N) = \text{const} + a \log N$. In the latter case, C would be infinite. Fortunately, we can rule this out by an independent rigorous argument given below.

Once we have an infinite graph, it is, of course, very interesting to know how fast the occupation probabilities decrease with the distance from the origin. With the number of nodes increasing exponentially with distance, it is obvious that the probability for the majority of nodes has to decrease exponentially. We shall prove now the much stronger result that the sum of all probabilities to be at *any* node a distance $\geq d$ from the origin,

$$Q(d) = \sum_P Q(P)\, \Theta(d(P) - d) \tag{4.3}$$

decreases exponentially with d for all rules. From this, it follows immediately that C is finite.

The proof follows from the existence of what we call "resetting strings". These are finite sequences which when observed in the output string lead to some node $\underline{\mathbf{P}}$, irrespective from the previous node. More formally, let $R = (s_1 \ldots s_m)$ be a resetting string of length m, and $M_R = M_s^m \ldots M_s^2 M_s^1$ be the product of M-matrices corresponding to it. Then all four columns of M_R are multiples of $\underline{\mathbf{P}}$, such that

$$M_R \mathbf{P} = \text{const}\ \underline{\mathbf{P}} \qquad \text{for all } \mathbf{P}. \tag{4.4}$$

For rule 22, such a resetting string with $\underline{\mathbf{P}} = \mathbf{P}_0$) is

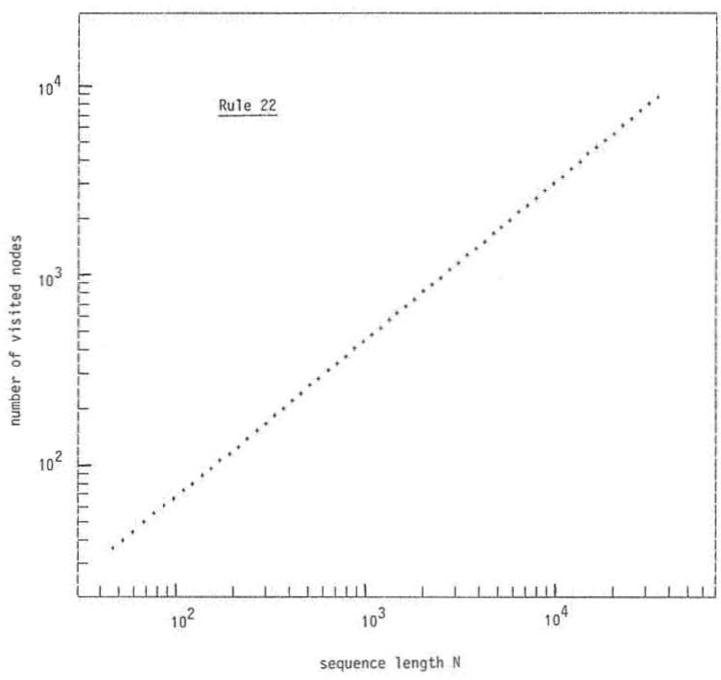


Figure 8: Average number of nodes reached during a forecast of a string of length N, obtained from a random input string (rule 22).

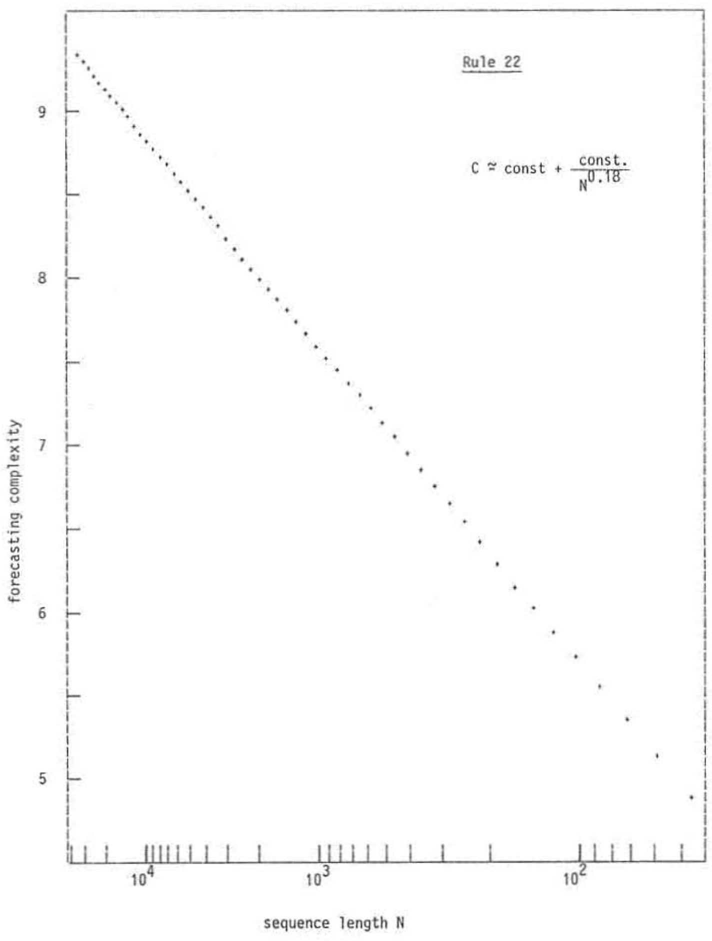


Figure 9: Average information which is stored after N observed symbols since it will be needed for keeping further forecasts optimal (rule 22). On the abscissa is plotted $N^{-.18}$, so that equation (4.2) with $\beta = 0.18$ would give a straight line.

$R = 00001101000010000.$

Furthermore, by working through the 4 possible terminations of the input string, it can be shown easily that R appears in any string with nonzero probability (i.e., for each input string there exists a finite extension which generates R as output). Thus, there is a finite rate by which any walk on the graph of figure 6 is lead back to the origin, and any information on the input string becomes useless for further forecasts. This rate sets a lower bound on the exponential rate by which $Q(d)$ has to decay, QED.

For some other rules, even a systematic search did not yield any resetting string in the above strict sense. But in each of these cases we found a finite number of strings, each one resetting part of the graph. Furthermore, from each node one of these resetting strings could be reached with finite probability. This is enough to prove a finite forecasting complexity in all elementary CA rules.

The existence of resetting strings has a number of further consequences.

As we had already shown in figure 7, the number of vertices in the graph of figure 6 and of similar graphs for other rules increases exponentially with distance from the origin. The set of vectors $\mathbf{P} \in R^4$ corresponding to these vertices is shown in figure 10 as a projection onto the $(P(0,0), P(0,1))$-plane. It obviously is a fractal set. Indeed, by straightforward box counting in three-dimensional space (remember that only three components of $\mathbf{P}$ are independent) we find a box-counting ("fractal") dimension $D_f \approx 2.2$. Also, qualitatively similar pictures are obtained for other projections and for different rules. As an example, we show in figure 11 a projection of the analogous set for rule 148. Finally, the set of different predictions $p(1)$ seems to be a fractally populated dense set as seen from figure 12.

Nevertheless, the Hausdorff-Besicovich dimensions (in contrast to the box-counting dimensions) vanish. This follows simply from the fact that these sets are countable. This might seem like a mathematical sophistry, but it is not at all. We can consider the evolution of $\mathbf{P}_n$ as a dynamical system, and we can estimate the information dimension of the attractor by a generalized Pesin-type formula $h = \sum_i \gamma_i D_i$ [1]. Here, the γ_i are the Lyapunov exponents, and the D_i are the partial dimensions. Due to the resetting strings, the Lyapunov exponents are all minus infinite: any difference $\delta\mathbf{P}_n$ will vanish after sufficiently many iterations, the number of which is independent of $\delta\mathbf{P}_0$. The above Pesin-type formula gives thus that the information dimension is zero, in agreement with the vanishing of the Hausdorff dimension.

Consequences of the positiveness of the box-counting dimension on one hand, and of the vanishing of the information dimension on the other, will be discussed in section 6. But before this, we shall use the graphs in figures 2 through 6 for computing entropies and forecasting complexities.

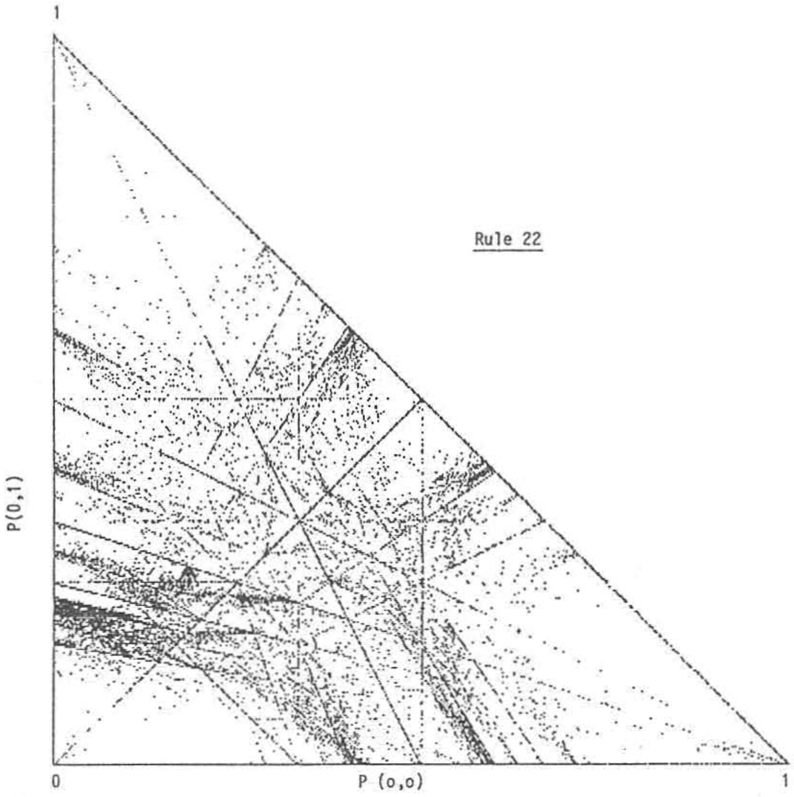


Figure 10: Projection of the set of vectors $\mathbf{P} \in R^4$ onto the $(P(0,0), P(0,1))$-plane.

5. Computing metric entropies and forecasting complexities; Markov chain approximation

It is obvious that figure 2 can be used to compute the entropy of rule 35. In each node i we have the probabilities $p_i(0)$ and $p_i(1)$ for the next link to be labeled "0" or "1". Together with the random sequence S of output symbols, this defines an ergodic Markov process. Its unique solution gives the node probabilities $Q(\mathbf{P}_i)$. The entropy is then given by

$$h = -\sum_i Q(\mathbf{P}_i)\{p_i(0)\,\log_2(p_i(0) + p_i(1)\,\log_s p_i(1)\}. \tag{5.1}$$

Numerically, a straightforward calculation gives $h = 0.875$ bits for figure 2.

Rule 35 leads to a finite-order Markov process of order 3. Thus, the above value of h could have been obtained also simply from the block entropies H_n. They are defined in terms of the probabilities $p\{S_n\}$ for finding the substring S_n of length n at any given place in the output string S, by

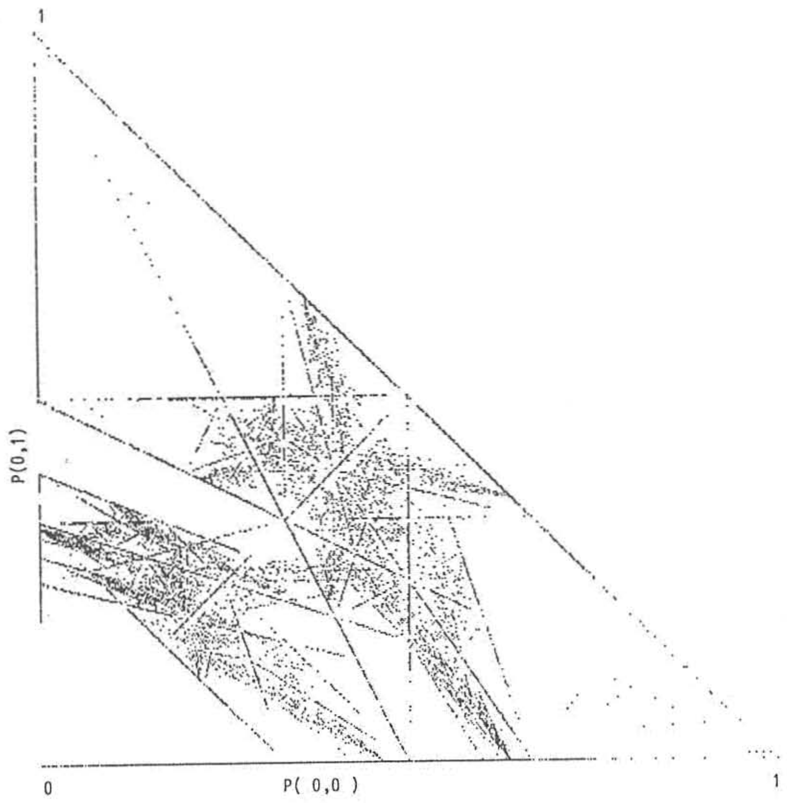


Figure 11: Analogous to figure 10, but for rule 146.

$$H_n = -\sum_{S_n} p\{S_n\} \log_2 p\{S_n\}. \tag{5.2}$$

The entropy is equal to

$$h = \lim_{n\to\infty} h_n \equiv \lim_{n\to\infty} (H_{n+1} - H_n). \tag{5.3}$$

An n^{th}-order Markov process is defined as a process where $h_m = h$ for all $m > n$, or equivalently by $p\{s_1 \dots s_{n+2}\} = p\{s_1 \dots s_{n+1}\}.p\{s_{n+2} \mid s_2 \dots s_{n+1}\}$.

In all other cases (except the trivial rules of subsection 3a), things are not so easy. In these cases, though the source of the output string is a Markov source, the string itself is non-Markovian. In physics terminology, the state of the source (i.e. the number of the actual node) is a hidden variable. It is well known that hidden variables can make an analysis very complicated.

In such cases, the h_n converge to h from above, in such a way that [5]

$$\sum_n (h_n - h)n \le C. \tag{5.4}$$

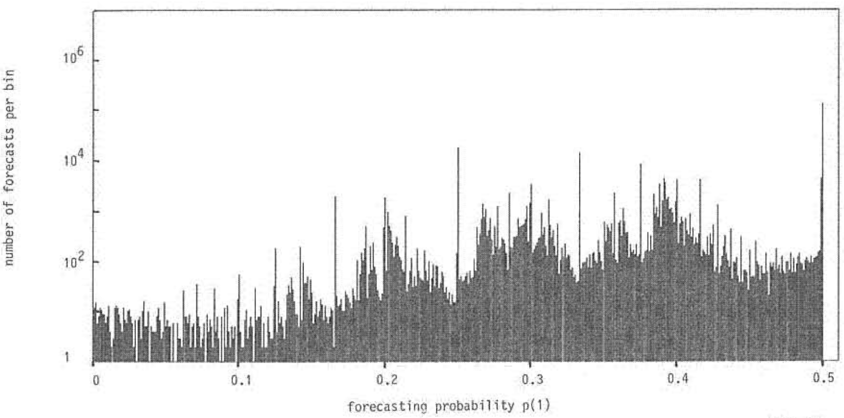


Fig. 12

Figure 12: Histogram of 3×10^5 forecasts $p(1)$ for rule 22. The unit interval on the abscissa is divided into 1000 bins. The height of each bar is proportional to the number of forecasts in the bin. Notice that the spikes are at rational values of $p(1)$.

This sum was called *effective measure complexity* in reference [5]. Equation (5.4) is amply fulfilled in all cases tested in the present paper.

Equation (5.1) can be used in some cases to compute h exactly, without using block entropies. This includes rules like 76, 4, and 18, where the graph contains at most one non-transient branch extending towards infinity. Since the transient parts can be neglected, $Q(\mathbf{P}_i)$ and $p_i(0)$ can be computed there exactly. Again, we shall not quote anything for rule 76, but we shall give the result for rules 4 and 18. Straightforward calculations give $h = 0.5076$ bits (rule 4) resp. $h = 0.6992$ bits (rule 18).

In both these cases, the entropy could also have been computed from equation (5.3). The most straightforward computation of block entropies H_n is by exact enumeration. In order to obtain H_n, one has then to go through all 2^{n+2} different input strings of length $n+2$. For large n (n larger than ~ 20) this becomes unpracticable. As this will also be of interest for the next section, we shall show now that the block entropies can be obtained more efficiently using our graphs and a slight modification of equation (5.1).

For a given rule, let us consider only that part of the graph which is a distance $\leq n$ from the origin. We claim that we can use this part for making approximate forecasts which essentially are equivalent to approximating the sequence S by an n^{th}-order Markov chain. The reason is simply that for each S_n the probabilities $p\{S_n\}$ can be read off the graph. We have just to start at the origin, take the first link according to the first symbol s_1, the next from there according to s_2, etc. The probability $p\{S_n\}$ is then simply obtained by multiplying the branching probabilities $p_i(s_i)$ read off at the i^{th} node.

Working through all S_n would involve exactly the same amount of work as direct exact enumeration. An important simplification is obtained by realizing that instead of the probabilities $p\{S_n\}$ all one needs are the quantities

$$Q^{(n)}(P_i) = \sum{}' p\{S_n\} \tag{5.5}$$

where the prime on the summation means that it runs over all sequences ending on node i. For $n \to \infty$, $Q^{(n)}(P_i)$ will tend towards $Q(P_i)$. Obviously, the $Q^{(n)}(P_i)$ satisfy a Markov equation

$$Q^{(n)}(P_i) = \sum_j p_j(s_n)\, Q^{(n-1)}(P_j) \tag{5.6}$$

where the sum runs over all nodes from which there is a link leading to P_i. Combining all this, we end up with

$$h_n = -\sum_i Q^{(n-1)}(P_i) \sum_{s=0,1} p_i(s) \log_2 p_i(s). \tag{5.7}$$

Since in all cases the number of nodes increases much slower than 2^n, it is much easier to compute the $Q^{(n)}(P_i)$ iteratively via equations (5.6) and (5.7) than by the direct enumeration. In addition, for the simpler rules the $Q^{(n)}(P)$ are very good approximations to the $Q(P)$ needed to compute C via equation (3.8). Most of the entries in table 1 are obtained in this way.

Each Markov chain can be represented by a finite graph. The construction of the minimal graph corresponding to the n^{th}-order Markov chain approximation is straightforward. First we build a binary tree of height n such that each allowed string $S = s_a \dots s_n$ is represented by a path leading up from the root. For each allowed string $s_1 \dots s_{n+1}$ of length $n+1$, we add then one link connecting the node reached by S to the node reached by $s_2 \dots s_{n+1}$. The graph obtained in this way is then minimized by identifying equivalent nodes as described in section 3c.

As an example, let us give the second order Markov approximation for rule 18. The part of figure 5 relevant for the block probabilities $p\{s_1 s_2 s_3\}$ is redrawn for convenience in figure 13. In this figure, we have also indicated the forecasts $p_i(0)$. The Markov graph according to the above description is given in figure 14a.

The careful reader will have realized that figure 13 contains indeed more information than utilized figure 14a. From figure 13, we cannot only obtain the probabilities $p\{s_1 s_2 s_3\}$, but we can also read off e.g. $p\{1011\} = 1/64$. It is a trivial exercise to verify that the second-order Markov approximation embodied in figure 14a would give a different result, $p\{101\}.p\{1|01\} = 1/48$.

Thus, finite parts of our graphs contain in general more information than that usable in a Markov chain approximation, suggesting that better approximate treatments should be possible. Different versions of alternative approximations will be discussed in the next section.

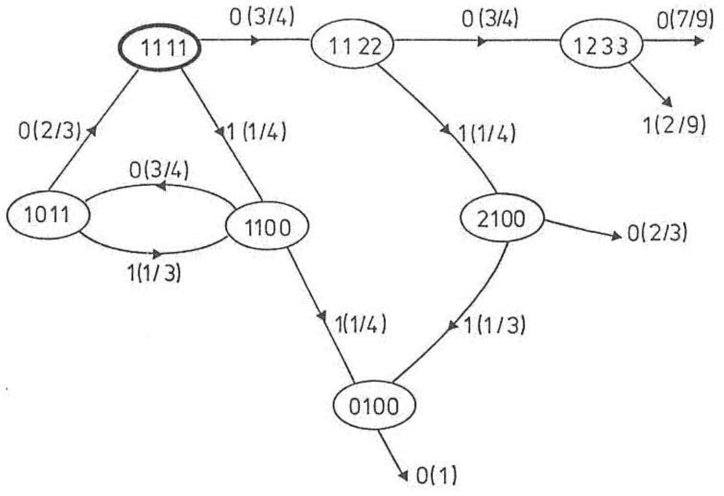


Figure 13: Enlarged part of the graph for rule 18 (figure 5). The rational numbers at the links indicate the forecasting probabilities $p_i(s)$.

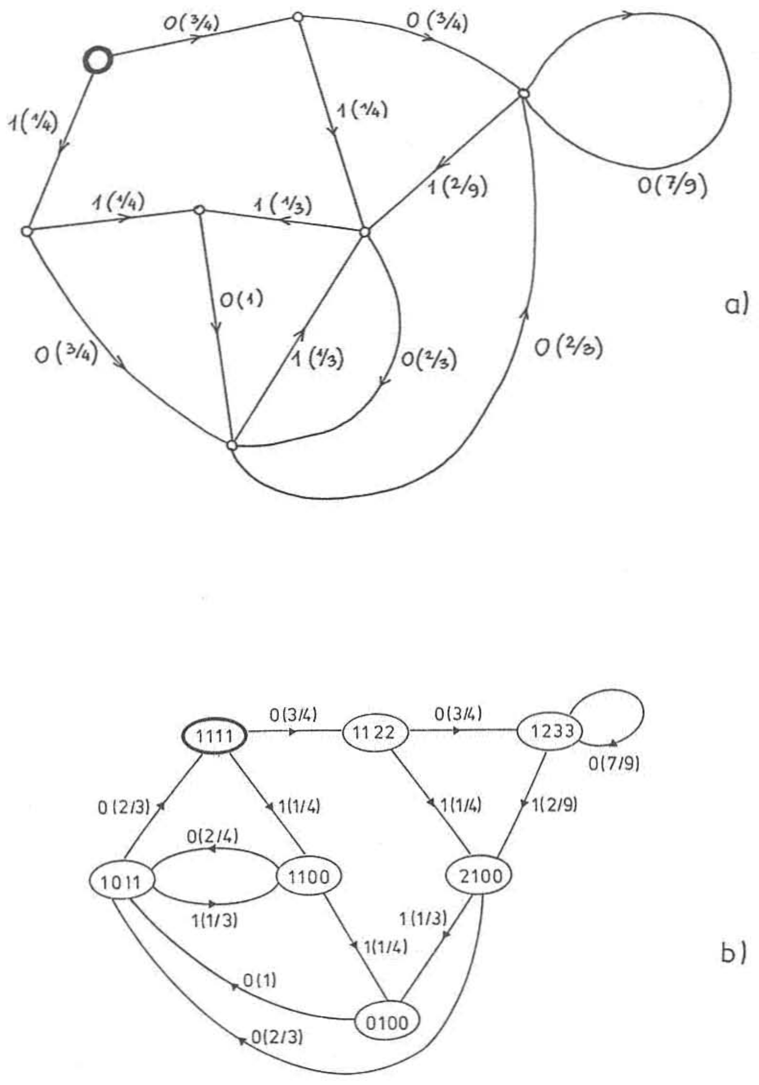


Figure 14: a) Graph for the Markov chain approximation based on figure 13; b) Graph obtained by truncating as explained in section 6.

6. Nearly optimal forecasting

We have seen above that in most cases we cannot practically perform optimal forecasts. Thus we have to take recourse to some kind of approximation. In addition to treating the output string S like a Markov chain (as in the last section), there are a number of other possible approximations:

(a) The first possibility is to simply truncate the graph. Links to nodes outside the retained part must then be replaced by links to some other nodes. The most natural way to do this is the following: when analyzing an output string S, we start at the origin and follow the links until we reach the first link (leaving form node $\mathbf{P}$, say) that would leave the truncated graph. Assume this happens at the k^{th} symbol of S. At this moment, we drop the first symbol of S, and start again at the origin with the second symbol. It might happen that the new path leaves the truncated graph already at $k' \leq k$. In that case we drop also the second symbol, and start a new path with the third symbol. This is repeated until we get a path for which symbol s_k leads to a node (say node $\mathbf{Q}$) in the truncated graph. We now connect node $\mathbf{P}$ to $\mathbf{Q}$ and continue with symbol s_{k+1} (leaving from node $\mathbf{Q}$), until the path leaves the graph again. At this point, we repeat the whole process.

In the example of rule 18, and with the truncated graph containing all nodes with distance < 3 from the origin, it is easily seen that this amounts in replacing figure 13 by figure 14b.

In this replacement we keep the $p_i(s)$ unchanged. This implies that in general the forecasted block probabilities $p\{s\}$, $p\{s_1 s_2\}, \ldots$ are no longer equal to the exact ones obtainable from the truncated graph. This is in contrast to the Markov approximation treated in the last section. It is the biggest drawback of this method, in particular for small truncated graphs. However, if the truncation involves only nodes far from the origin and with small weight, then this is much less of a problem, and the truncation method is very efficient.

While the approximation of section 5 effectively assumes that the output sequence is Markovian, the truncation approximation essentially assumes that this sequence is non-Markov but originates from a Markov *source*.

One might hope that it should always be possible to modify the $p_i(s)$ such that the forecasted block entropies agree with those from the truncated graph. That this is not so is seen by a counter example: for rule 76, one finds that the part of the graph which has distance < 8 from the origin specifies *all* block entropies (of any length) completely. Thus, no finite graph can incorporate all constraints implied by a truncation at any distance ≥ 7, and still give the correct block probabilities.

Among others, we can apply truncation in the following three ways:

(a1) by retaining only nodes with distances $< n$ from the origin.

- **(a2)** by cutting off the transient parts. Consider e.g. the figures 4 and 5. In both cases, the transient parts are cut off by replacing the links leaving the start node by figure 15.
- **(a3)** by truncating all nodes whose estimated weight $Q(\mathbf{P})$ is below some threshold value. While this would be the most preferable, we have not found a practical way to do this for complex rules such as rule 22. The reason is that the $Q(\mathbf{P})$ can only be estimated there by analyzing very long sequences, using much larger graphs than one wants to retain at the end.

(b) The above truncations do not use the fact that the nodes are vectors in 3-space, and that the resulting forecasts are real numbers. Assume we make an approximate graph by lumping together two nodes. Then, the error made will depend on the distance in $\mathbb{R}^3$ between nodes resp. the distance in $\mathbb{R}$ between forecasts. *A priori* one might think of specifying the maximal tolerated error or specifying the average tolerated error, or of specifying the probability with which a certain error may appear.

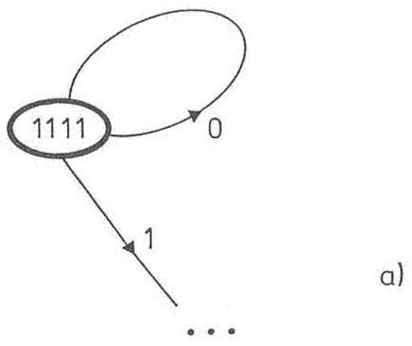
We found that the first possibility (not tolerating any forecast which is wrong by an error larger than some prescribed bound ε) is unfeasible for the more interesting cases like rule 22. The reason is that although the average Lyapunov exponents are all $-\infty$ as explained in section 4, there are particular strings with positive forecasting Lyapunov exponent. Thus we cannot guarantee that any small error will not blow up arbitrarily.

We did not study the last possibility (tolerating large errors only with a certain frequency) either. What we studied extensively instead were some variants of the second case:

- **(b1)** We put a three-dimensional grid in $\mathbf{P}$-space of lattice constant ε and replaced any vector $\mathbf{P}$ by the central point of its box. We applied the map (3.4) then to this central point. We call this the "lattice approximation" in the following.
- **(b2)** In contrast to the above, we did not replace the point $\mathbf{P}$ by the center of the box. Instead, if two points $\mathbf{P}_1$ and $\mathbf{P}_2$ fell into the same box, we replaced them by the point for which the string leading to it from the start node was shorter. If both lengths were the same, we retained the point with the lexicographically smaller string.

In the following, we shall present only results from approaches (a1) and (b1).

From the results of section 4 it should be clear that the forecasting complexity as defined in equation (3.8) stays finite in the limit of zero errors ($n \to \infty$ resp. $\varepsilon \to 0$). This is in contrast to other complexity measures. All non-optimal forecasting algorithms discussed above can be represented by finite graphs, and have thus a finite regular language complexity in the sense



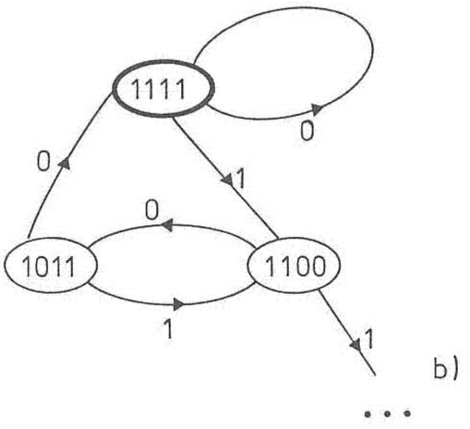


Figure 15: Modification of the start node for rules 4 (part a) and rule 18 (part b), cutting off the transient part.

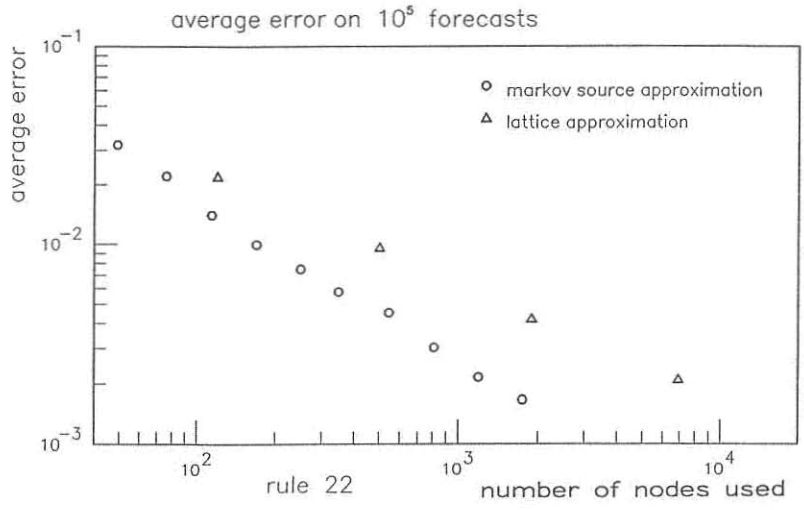


Figure 16: Average differences between exact and approximate forecasts of the probability $p(0)$ to observe "0", for rule 22, versus the size of the needed graph (whose logarithm is the regular language complexity).

of reference [10] as long as n resp. ε are finite. But in the limit of optimal forecasts, it diverges with the following asymptotic behavior (figure 16):

$$\langle | \Delta p(0) | \rangle = k \, . \, C_r^{-\beta} \tag{6.1}$$

where C_r is the regular language complexity as estimated from the truncated graphs without minimalizing them. Strictly, this is somewhat larger than the exact regular language complexity, since that is defined via the minimal graphs which accept the same language. But we believe that the difference is rather small. For truncation at fixed n we found $k = 0.72$ and $\beta = 0.82$, while for the lattice approximation we obtained $k = 0.36$ and $\beta = 0.58$.

For both approaches we computed the forecasting complexities and the average errors committed in the forecasts $p(0)$ when parsing a random string of length 10^5. At equal complexities, we found that the omitted errors were comparable in both both approaches (a) and (b) (see figure 17).

For the method (a) we found that the average error scales with the maximal allowed distance n from the start according to the law (figure 18):

$$\langle | \Delta p(0) | \rangle = k \, . \, e^{-\beta n} \tag{6.2}$$

where $k = 0.20 \pm 0.02$ and $\beta = 0.32 \pm 0.02$. In case (b) we found an analogous scaling of the average error with the mesh size (lattice constant) ε (figure 19):

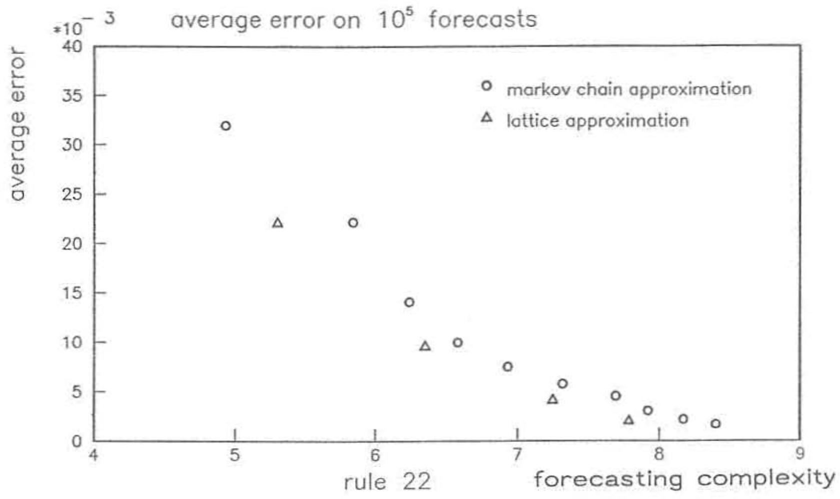


Figure 17: As in figure 16 but versus the forecasting complexity in bits.

$$\langle \mid \Delta p(0) \mid \rangle = k \,.\, \varepsilon^{-\beta} \tag{6.3}$$

where $k = 0.26 \pm 0.02$ and $\beta = 1.17 \pm 0.1$.

7. Conclusions

In dynamical systems one often encounters symbol sequences which are neither deterministic nor entirely random. Also, they are in general not Markov chains even if they are emitted by Markov sources. In such cases it can be non-trivial to utilize the known structure for making optimal forecasts, even if the system might seem very simple.

We have presented in this paper a rather detailed study of the complications one is driven into during such an enterprise. We studied only a class of extremely simple toy models which at first sight might have seemed without any interest. Our original motivation was that these models are simple enough to allow a rather detailed analysis (though not mathematically rigorous on many places). This was justified later by the richness of the structures found.

Our hope is of course that studying such simple models might be useful finally in understanding the difficulties in performing actual forecasts in practice. Little work has been devoted to this problem theoretically, compared to the large amount of work on the theoretical limitations to the possibility (not to be confused with the difficulty) of forecasts. We stress again that the difficulty of making an optimal forecast is not related to the Shannon entropy, metric entropy, or Kolmogorov complexity of the sequence. It is rather

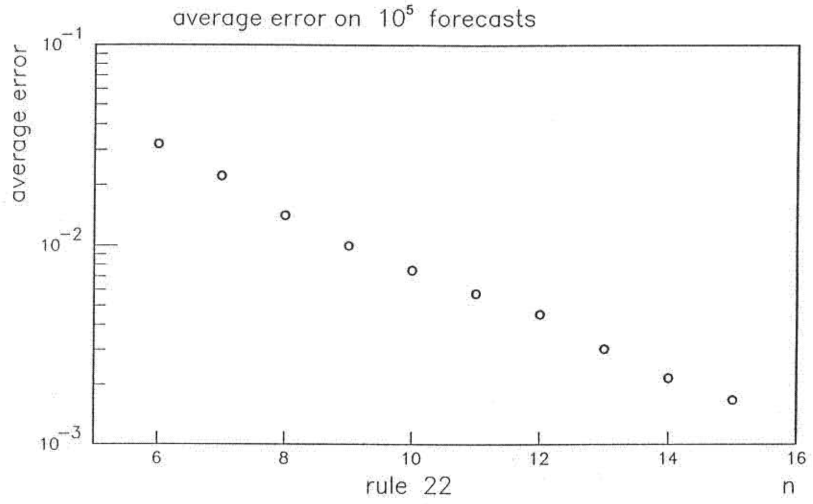


Figure 18: As in figure 16 for the Markov source approximation but versus the maximal distance from the starting node.

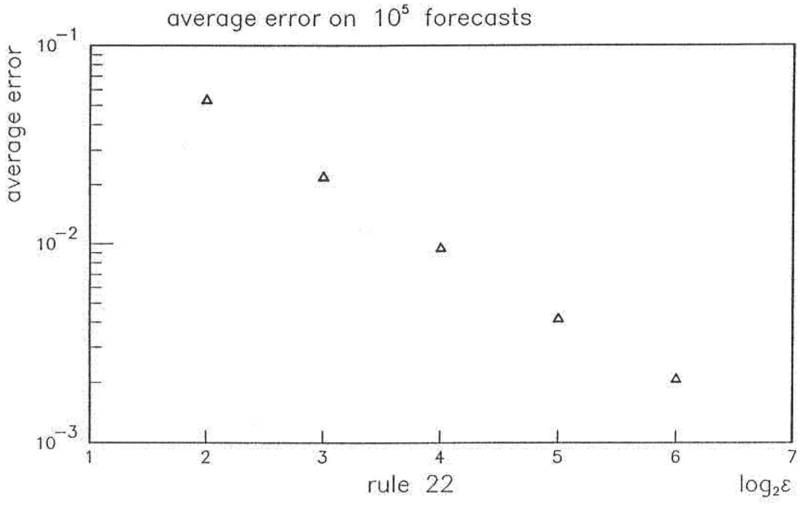


Figure 19: As in figure 16 for the lattice approximation but versus the binary logarithm of the mesh constant ε.

similar to a grammatical complexity of a formal language, but couched in a probabilistic setting.

It is clear that there is a rather close analogy between symbol sequences without a superimposed measure (formal languages) and such with a measure (discrete stochastic processes). In particular, Markov chains correspond to finite complement languages. In both, all features of the process can be read off from finite length trajectories. Chains emitted from Markov sources (i.e. functions of Markov processes, depending on "hidden variables") are analogous to regular languages, since in both cases the source has only a finite memory. Are there similar analogies between higher level languages (context free, context sensitive, recursively enumerable sets) and classes of stochastic processes? We do not know. It seems that not much work has been done on classifying non-Markovian stochastic processes.

We have seen that the concept of an infinite graph associated to a stochastic process is very useful. Similar infinite graphs have been used for logistic maps in reference [14]. The variability of the topology of these graphs was amazing. Also, the forecasting complexity of the model was often closely reflected in the complexity of the topology. Is it possible to characterize also the Chomsky hierarchy of formal languages by the topology of associated infinite graphs? It has been shown [15] that context-free grammars correspond to graphs which are essentially tree-like, but beyond that not much seems to be known in general.

References

[1] J. P. Eckmann and D. Ruelle, *Rev. Mod. Phys.*, **57**, (1985) 617.

[2] P. Billingsley, *Ergodic Theory and Information*, (Wiley, New York, 1965).

[3] P. Collet and J. P. Eckmann, *Iterated Maps of the Interval as Dynamical Systems*, (Birkhhäuser, Basel).

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

[5] P. Grassberger, *Int. J. Theor. Phys.*, **25**, (1986) 907.

[6] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, (Addison-Wesley, Reading, MA, 1979).

[7] A. Kolmogorov, "Three approaches to the quantitative definition of Information", *Problems of Information Transm.*, **1** (1965).

[8] G. Chaitin, "Algorithmic information theory", *IBM J. Res. Dev.*, **21** (1977) 350.

[9] S. Wolfram, *Rev. Mod. Phys.*, **55**, (1983) 601.

[10] S. Wolfram, *Commun. Math. Phys.*, **96**, (1984) 15.

[11] L. Hurd, "Formal language characterizations of cellular automaton limit sets", Princeton Univ. preprint (1986) M. Nordahl, Goteborg preprint (1987).

[12] J. E. Hutchinson, *Indiana Univ. Math. J.*, **30**, (1981) 713; M. Hata, *Japan J. Appl. Math.*, **2** (1985) 381; M. F. Barnsley and S. Demko, *Proc. Royal Soc. London*, **A399** (1985) 243; M. F. Barnsley and A. D. Sloan, *BYTE*, January 1988, 215.

[13] S. Wolfram, *Theory and Applications of Cellular Automata*, (World Scientific, Singapore, 1986).

[14] P. Grassberger, Wuppertal preprint WU-B 87-5 (1987).

[15] M. Nordahl, Chalmers University thesis (1988).