

# Self-replicating Sequences of Binary Numbers — The Build-up of Complexity

Wolfgang Banzhaf

Department of Computer Science, Dortmund University

Baroper Str. 301, 44221 Dortmund, GERMANY

banzhaf@tarantoga.informatik.uni-dortmund.de

## *ABSTRACT*

*A recently introduced system of self-replicating sequences of binary numbers (strings) is generalized. It is extended to include strings of arbitrary length. For this purpose, first, the folding methods of strings into two-dimensional operators are expanded to include strings of arbitrary size. Second, rules of interaction between strings of different lengths are established. As natural consequence of these interactions, changes in string length are observed. Using an effective model of length changes, the build-up of complexity as measured by the average sequence length in a string population is studied.*

# 1 Introduction

The emergence of complex structures in Nature from simple building blocks is one of the most fascinating topics in science [1]. The tendency of living beings in particular, to generate ever more complicated systems is striking. How is it that a network of self-reproducing entities, mainly based on the 6 abundant elements CHNOPS [2] i.e. life on planet Earth, could grow to such a complexity as we can observe today?

Over the last decades, molecular biology has revealed a great deal about the underlying basic interactions between structures that drive these entities of life [3]. One important result has been that double strands of DNA carry information from generation to generation in the form of molecular symbols (the nucleotides A, C, G, T) which, upon translation into another molecular system (the amino acids) becomes functional as 3D-site dependent catalysts (proteins) or as structural support systems.

Proteins especially are interesting objects of study, since they are able to accelerate chemical reactions by factors ranging from  $10^6$  to  $10^{12}$  [4]. They assume their phenotypic (native) form under the influence of the attraction and repulsion the molecules exert on each other. As in other catalytic systems, the 3D shape of a protein which results from the specific order of amino acids in its primary sequence is decisive.

Yet the DNA-protein system is believed to have been preceded by a more primitive RNA based system which should not have required separation into a conservation and a functional subsystem. Instead, RNA could play a double role, acting as both conservationist and catalyst. The former role could be played by the primary sequence of nucleotides on RNA strings (only differing in one nucleotide from DNA), whereas the latter role would be played by the 2- or 3-dimensional shape a string could assume under appropriate temperature conditions. But even in such a more primitive system, the question remains: What drove this system to steadily build-up complexity?

In the present contribution we want to examine the build-up of complexity in a simplified system which was derived using metaphors of the RNA paradigm. It is based on the fact that the major information processing machines of today, digital computers, use sequences of binary numbers to represent and manipulate information. In previous work [5, 6] we have proposed to introduce a *mapping* of these one-dimensional sequences of binary numbers into two-dimensional forms naturally able to interact with one-dimensional forms. A restricted class of sequences, namely those whose length was a square number, could be assigned corresponding 2D (matrix) forms by applying folding methods. We then proceeded by establishing rules of interaction between matrix and sequential forms of strings derived from matrix/vector operations in mathematics. The result was a set of effectively interacting binary strings, some members of which could replicate other strings or even themselves.

Here we want to generalize the proposed folding methods to include strings of every length. Then we shall set up the rules for interaction of strings of different length. Finally, we study the growth in complexity as measured by the average length in a population of strings, starting from a population of smallest size strings.

String length $N$	Folding approach		
	(a)	(b)	(c)
2	1 , 2	1 , 3	2 , 2
3	1 , 3	2 , 3	2 , 2
4	2 , 2	2 , 3	3 , 3
5	1 , 5	2 , 3	3 , 3
6	2 , 3	2 , 4	3 , 3
7	1 , 7	2 , 4	3 , 3
8	2 , 4	3 , 4	3 , 3
9	3 , 3	3 , 4	4 , 4
10	2 , 5	3 , 4	4 , 4

Table 1: Matrix size  $N_1, N_2$  or  $N_2, N_1$  for short strings using the different approaches (a) ... (c) to folding (see text).

## 2 General Folding of Strings

The previously proposed methods for a folding of sequences of (binary) numbers into a 2-dimensional matrix form were devised for strings whose length  $N_S$  is a square number. The reason is that if  $N_S$  belongs to the set of square numbers,  $\mathcal{N}_{sq} = \{1, 4, 9, 16, 25, \dots\}$ , a string  $\vec{s}$  containing  $N_S$  binary numbers  $s_i, s_i \in \{0, 1\}$

$$\vec{s} = \{s_1, s_2, \dots, s_{N_S}\}, \quad N_S \in \mathcal{N}_{sq} \quad (1)$$

could naturally fit into a two-dimensional matrix of size  $\sqrt{N_S} \times \sqrt{N_S}$ . We considered folding methods to be applied to *all* strings in the population, disregarding their respective combination of numbers. In principle, any random placement of numbers on the matrix grid is allowed and constitutes one special folding method. A particular class of foldings, the topological foldings, are the result of self-avoiding random walks on the matrix grid. Topological foldings are characterized by the fact that adjacent neighbors in the sequence of numbers are also neighbors in the 2-dimensional matrix.

There are three different approaches to a generalization of folding methods to strings of arbitrary length  $N$ . Two of these allow for non-square matrices of size  $N_1 \times N_2$ , and two approaches allow for a non-compact folding with empty spaces between numbers. We shall list these approaches here in brevity and later concentrate on approach (a).

(a) Compact<sup>1</sup> folding in non-square matrices, where

$$N_1 \times N_2 = N. \quad (2)$$

(b) Non-compact folding in non-square matrices

$$N_1 \times N_2 > N. \quad (3)$$

---

<sup>1</sup>Compact foldings do not have any spacing between adjacent string elements

$N$	<i>String number</i>							
	0	1	2	3	4	5	6	7
2	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$				
3	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$

Table 2: All string types of systems  $N = 2$  and  $N = 3$ . Compact folding leads to operators which look the same, except for a possible transposition.

(c) Non-compact folding in square matrices

$$N_1 \times N_2 > N, \quad N_1 = N_2. \quad (4)$$

In all three approaches a bias could be added toward the smallest solution, i.e.

$$N_i = \sqrt{N} + \varepsilon_i, \quad i = 1, 2 \quad (5)$$

with  $|\varepsilon_i|$  as small as possible.

Table 1 gives an overview of various folding methods for strings up to  $N = 10$ .

In approaches (a) and (b), a symmetry breaking has to occur, since only one of two alternatives,  $N_1 > N_2$  or  $N_2 > N_1$  can materialize in a folding. Both cases differ profoundly (at least in the simulations considered here) in their consequences, as the latter will lead to an increase of string complexity whereas the former will lead to a decrease (see Section 3).

If the length of a string is a prime number (as it was here for both cases depicted in the Table), a somewhat special situation occurs in approach (a), since the 1-dimensional form of a string and the 2-dimensional form look alike, except for a possible transposition. In Table 2 we list all strings of length  $N = 2$  and  $N = 3$ . Note that the system  $N = 1$  is trivial and  $N = 4$  was treated elsewhere [5, 6].

### 3 Interaction of strings of different length

The basis for the interaction of strings was a new interpretation of mathematical operations. We consider the outcome of an interaction in the form of a new string generated by a cooperation of two strings as being added to the existing population of strings. Thus we assume that both the operator and the operand be conserved in the course of an operation. The algorithm proposed in [5, 6] subsequently corrects for the addition of a string by destroying another member of the population. In this way, a competitive system is established in which efficiently replicating string types become more and more frequent whereas other types die out.

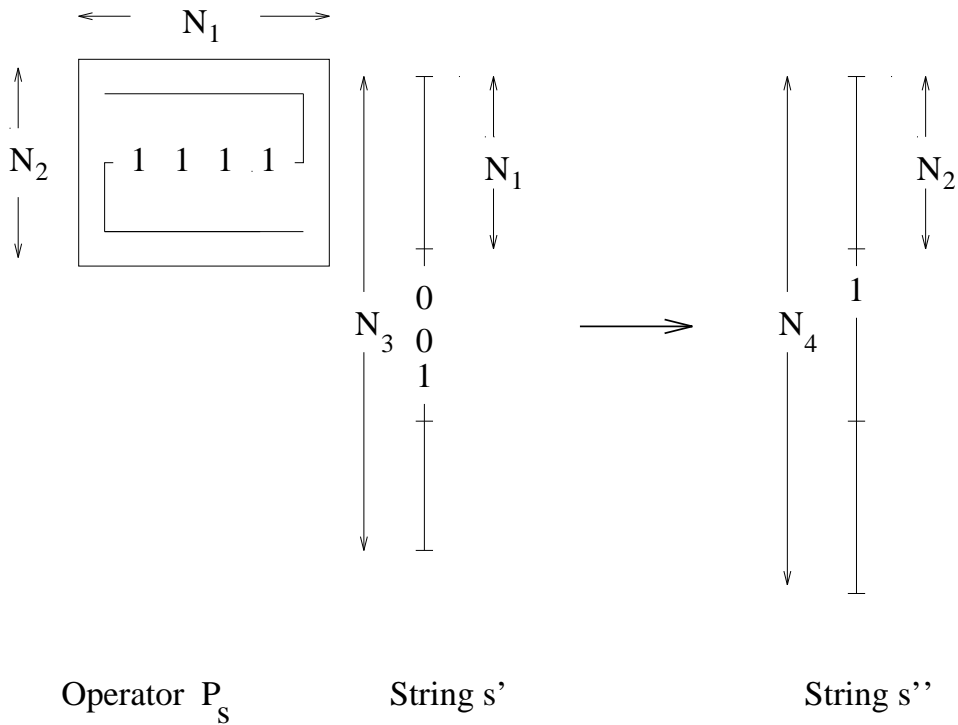


Figure 1: The interaction of string  $\vec{s}$ , folded into operator  $\mathcal{P}_{\vec{s}}$  and string  $\vec{s}'$  generates a new string  $\vec{s}''$  of usually different length.  $N_1, \dots, N_4$  are numbers of components.

An interesting interaction of strings may take place if one string is in the 2-dimensional form and the other is in the 1-dimensional form. Details have been explained in Ref. [5, 6], where we used the following squashed scalar product operation as the interaction rule between an operator/string pair ( $N = N_S \in \mathcal{N}_{sq}$ ):

$$s''_{i+k\sqrt{N}} = \sigma \left[ \sum_{j=1}^{j=\sqrt{N}} P_{ij} s'_{j+k\sqrt{N}} - \Theta \right] \quad (6)$$

$$i = 1, \dots, \sqrt{N} \quad k = 0, \dots, \sqrt{N} - 1.$$

$\sigma[ \ ]$  is the squashing function

$$\sigma[x] = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases} \quad (7)$$

and  $\Theta$  is an adjustable threshold, usually set to  $\Theta = 1$ .

Here we have to generalize the above rule to include interactions of strings of different length. A consistent generalization is possible in the following form (see Figure 1): Suppose we have a matrix of size  $N_1 \times N_2$  interacting with a string of length  $N_3$ . The operator locally interacts with  $N_1$  elements of the string in order to generate one product component. This operation will be repeated  $N_2$  times, after which the operator moves on to interact with the next  $N_1$  elements of the string. The product string will thus consist of  $N_4$  elements with

$$N_4 = \left\lceil \frac{N_3}{N_1} \right\rceil \times N_2. \quad (8)$$

$\lceil x \rceil$  are Gaussian parentheses producing the next larger integer to  $x$ .

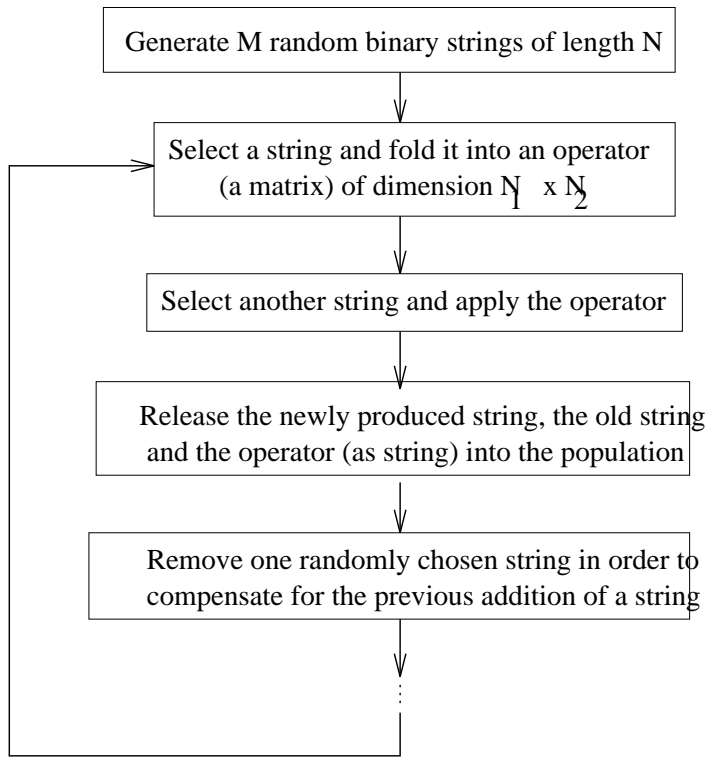


Figure 2: The important part of the algorithm for a population of  $M$  strings starting from minimal length  $N_{min}$ . The dotted line symbolizes other parts of the algorithm not relevant in the context discussed here.

In mathematical terms, the interaction reads:

$$s''_{i+kN_2} = \sigma \left[ \sum_{j=1}^{j=N_1} P_{ij} s'_{j+kN_1} - \Theta \right] \quad (9)$$

with

$$i = 1, \dots, N_2 \quad k = 0, \dots, \left\lceil \frac{N_3}{N_1} \right\rceil - 1.$$

Non-compact foldings possess matrix positions where no-operation (nop) symbols reside. In the context of operations used here, they are computationally equivalent to "0". This should be seen in contrast to the "wild-card" symbol \*, used in Holland's Genetic Algorithms [7] which usually means "don't care".

Let us consider an example of the interaction of a string  $s = (1 \ 0 \ 1 \ 0 \ 0 \ 0)^T$  with length  $N = 6$  ( corresponding operator:  $\mathcal{P}_s$ ) with a string  $s' = (1 \ 1 \ 1 \ 1 \ 1)^T$  of length  $N' = 5$ . The operator  $\mathcal{P}_s$  may be written as a matrix with size  $N_1 = 3, N_2 = 2$

$$\mathcal{P}_s = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}. \quad (10)$$

The resulting string  $s''$  will have  $N'' = 4$  components:

$$s'' = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad (11)$$

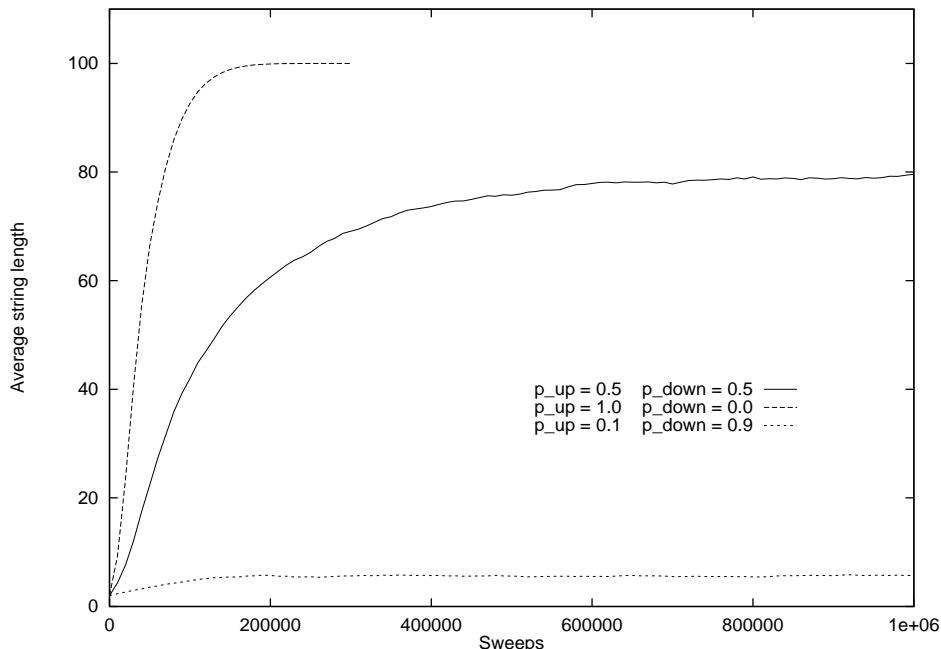


Figure 3: Development of average string length for a system with  $M = 10^5$  strings, starting all strings with length  $N = 2$ . 3 parameter settings which characterize different folding scenarios were used. Even for low probability of complexity increasing folding  $p_{up}$ , the average length of strings increases.

where we had to discard part of the operator in the second half of the computation due to missing partners in the string.

We can immediately recognize that interactions will usually change the length of a string, depending on the previously mentioned symmetry breaking feature. If  $N_1 > N_2$ , the length of the newly produced string will decrease:  $N_4 < N_3$ . If, on the other hand,  $N_1 < N_2$  then the length of it will increase:  $N_4 > N_3$ .

The important part of the algorithm is shown in Figure 2. It consists of various steps to ensure that a competition between different string types takes place in the system. What is important in the present context is that a population of strings with varying length which can be characterized by an average string length will react via the various reaction channels and relax to a state with different average string length.

## 4 The growth in complexity

The information contained in a bit string may be roughly measured by the length of that string. Longer strings are certainly more capable of performing complex tasks than are short ones. We may therefore take the length of a string (or its logarithm) as a very crude measure of its complexity [8, 9].

A natural consequence of the interaction of strings of different length was the change in this roughly defined complexity. As can be seen easily by inspection of equ. (8), changes

in complexity are particularly dramatic if a string whose length is a prime number folds into matrix form. If we allow for length increasing folding, the complexity build-up in a population seems inevitable, as soon as there exists a small number of strings with  $N > 1$ . Therefore, after starting from a population of  $N = 2$  strings we may observe a continuous increase in the average string length which will never stop, unless we introduce some limit on the length of strings. Here we shall put the limit at  $N_{max} = 100$ .

Since we are concerned with the build-up of complexity, we shall present only simulations where the length of strings is observed. Thus we neglect the lower level of the system, the resulting detailed sequences of binary numbers, and defer a study of related issues to a later contribution. In other words, we only consider an effective model here, concentrating on the most important aspect, the change in complexity.

We shall introduce two parameters  $p_{up}$  and  $p_{down}$  which determine the probabilities that a complexity increasing or a complexity decreasing folding is applied.

Figure 3 shows a typical run for a population with  $M = 100.000$  strings for some parameter settings. At the beginning, the string population consists of strings of length  $N_{min} = 2$  only. If the complexity increasing folding is applied exclusively, the system quickly ends up at the allowed maximum of string lengths. If, on the other hand  $p_{down} \neq 0$ , a length distribution results which is, after a certain growth period in the average length of strings, constant with small fluctuations. We can observe that even for small values of  $p_{up}$  the average length of strings increases. Note that the average level must also crucially depend on  $N_{max}$ .

Figure 4 a - e give a more detailed account of what is happening in a population of strings. They show the distribution of string lengths at certain moments in time for a run with  $p_{up} = p_{down} = 0.5$ . The drive towards higher complexity is clearly visible. The distribution stabilizes after approximately  $10^6$  iterations through the algorithm. Figure 5 shows the resulting distribution for the third parameter setting in Figure 3. Here, too, some strings have reached maximum length.

## 5 Summary

In this contribution we have extended the range of a previously proposed RNA-inspired algorithm to strings of arbitrary length and to interactions of strings with different length. We have found that, as a result of normal interactions, strings of different lengths were produced. By observing the growth in the average length of strings in a population starting with minimal length strings we have demonstrated the drive of the proposed system to build-up complexity.

The observed phenomenon is not constrained to the subset of compact folding methods reported in the simulation. Instead, it seems to be a constant theme through these kinds of algorithms. Even using a very small probability  $p_{up}$  led to an increase in the average string length. It thus seems to be more of an inherent feature of the algorithm



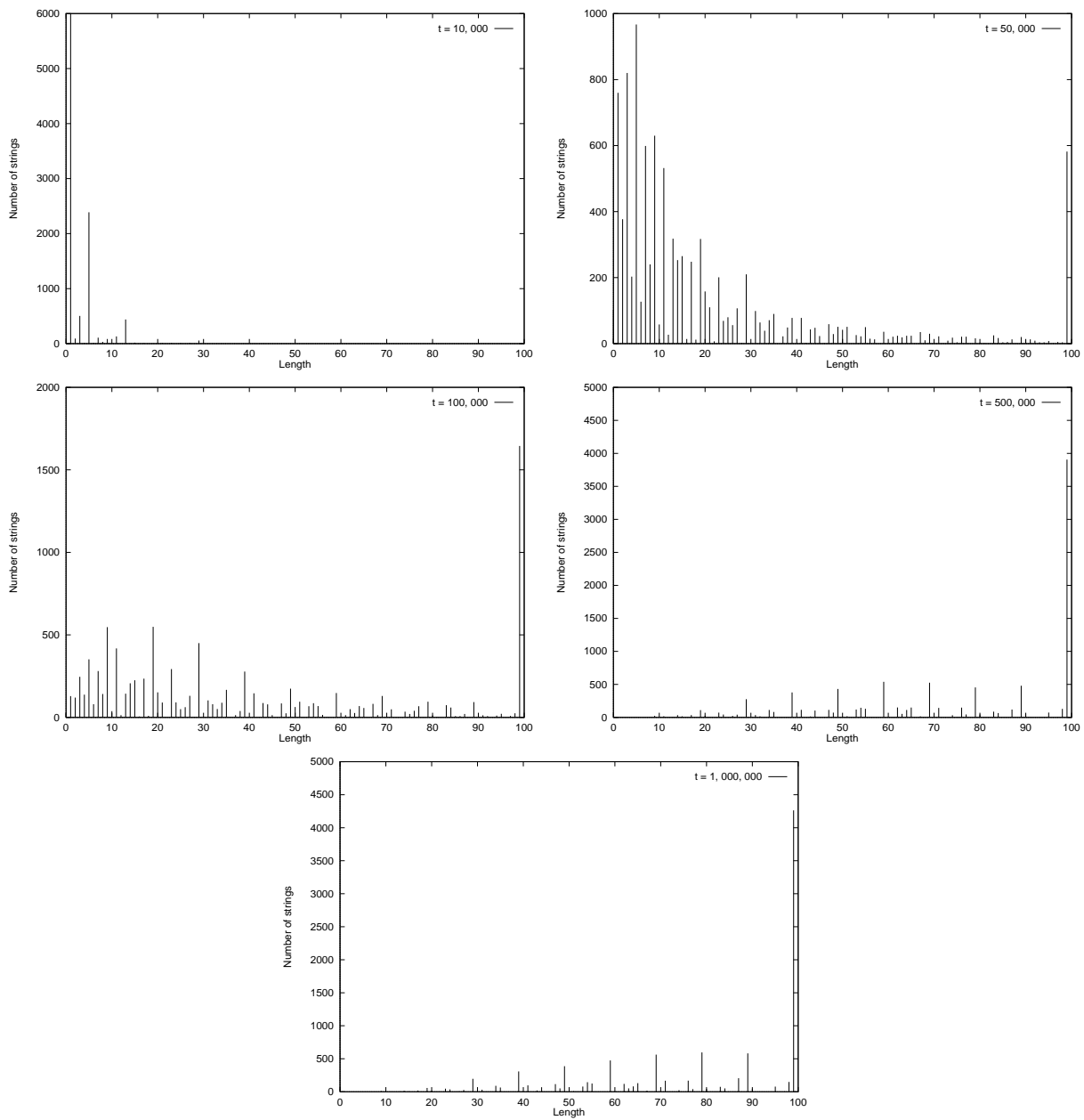


Figure 4: Distribution of string lengths for the simulation of Figure 3,  $p_{up} = p_{down} = 0.5$ , after 10.000 (a), 50.000 (b), 100.000 (c), 500.000 (d) and 1.000.000 (e) sweeps through the algorithm.

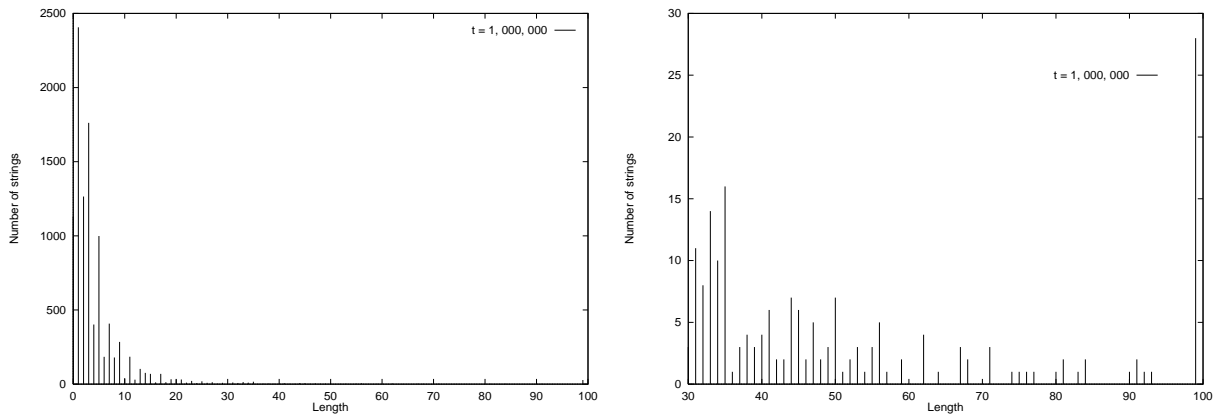


Figure 5: (a) Total distribution of string lengths in the simulation of Figure 3 for  $p_{up} = 0.1, p_{down} = 0.9$  after 1,000,000 sweeps through the algorithm. (b) Partial distribution of string lengths from  $N = 30$  to  $N_{max} = 100$  for the same run as (a).

than a product of particular circumstances.

Note, however, that we did start with strings of small length avoiding the trivial strings  $N = 1$  which are identical to the building blocks of all strings. Starting a system from  $N = 1$  strings seems to be possible but was omitted here for the sake of clarity.

The first results on the build-up of complexity presented here can only be a starting point for a more thorough investigation of the phenomenon. Certainly, the "bit" level of strings has to be included in order to gain a comprehensive picture. Since strings in 2-dimensional form work on 1-dimensional strings by repeatedly applying the same operations, we expect self-similar structures to play a prominent role.

## Acknowledgement

Part of this work has been done during my stay at Mitsubishi Electric Corporation in Japan. I am grateful to Drs. T. Nakayama and K. Kyuma for making my visit possible. Valuable discussions with Mr. T. Iwamoto, Dr. W. Fontana and Dr. M. Schmutz are gratefully acknowledged. I am indebted to Dr. J. Bell for carefully reading an earlier version of this manuscript.

# References

- [1] *SFI Studies on the Sciences of Complexity*. Addison-Wesley, Reading, Mass, 1988 - 1992. See, among others, the growing number of volumes in the series.
- [2] H.J. Morowitz. *Energy Flow in Biology*. Academic Press, New York, 1968.
- [3] J.D. Watson. *Molecular Biology of the Gene*. W.A. Benjamin, Menlo Park, CA, USA, 3rd ed. edition, 1976.
- [4] Garnier J. Robson B. *Introduction to Proteins and Protein Engineering*. Elsevier, Amsterdam, 1986.
- [5] W. Banzhaf. Self-replicating sequences of binary numbers. *Computers and Mathematics*, 26:1 - 8, 1993.
- [6] W. Banzhaf. Self-replicating sequences of binary numbers —foundations i and ii: General and strings of length  $n = 4$ . *Biological Cybernetics*, 69:269 - 281, 1993.
- [7] J.H. Holland. *Adaptation in natural and artificial systems*. Univ. Michigan Press, Ann Arbor, 1975.
- [8] C.H. Bennett. Dissipation, information, computational complexity and the definition of organization. In D. Pines, editor, *Emerging Synthesis in Science*. Addison Wesley, Reading, Mass, 1988.
- [9] R.G. Palmer. Broken ergodicity. *Adv. in Physics*, 31:669 - 735, 1982.