SFI Working Paper #07-08-016

Factorizable Language: From Dynamics to Biology

Bailin Hao and Huimin Xie

29 May 2007

Chapter 5 in *Annual Reviews of Nonlinear Dynamics & Complexity* Ed. by Heinz G. Schuster, published by Wiley-VCH, 2008

Here is the draft (Page numbers are tentative)

Bailin Hao

T-Life Research Center & Department of Physics, Fudan University, Shanghai 200433, China
Santa Fe Institute, Santa Fe NM87501, USA
hao@itp.ac.cn

Huimin Xie

Department of Mathematics, Suzhou University, Suzhou 215006, China
szhmxie@yahoo.com.cn

1 Factorizable Language: From Dynamics to Biology

Bailin Hao and Huimin Xie

There is no universal measure of complexity. When the problem under study leads to description by means of symbolic sequences, formal language theory may provide a convenient framework for analysis. In this review we concentrate on a special class of languages, namely, factorizable languages to be defined later, which occur in many problems of dynamics and biology. In dynamics we have in mind symbolic dynamics of unimodal maps and complexity of cellular automata. In biology we draw examples from DNA and protein sequence analysis.

1.1 Coarse-Graining and Symbolic Description

Let us start by making the following observation [1].

A high energy physicist recognizes the six lower case letters u, d, c, s, b, t as quark names and associates them with a certain mass, charge and quantum numbers such as "charm" or "flavor". More scientists use the symbols p, n, e to denote proton, neutron and electron each having a certain mass, charge, spin or magnetic moment, but they are not concerned with from which three quarks a proton or neutron is made.

Chemists consider $H, C, N, O, P, S, \cdots$ to be element names and know their atomic number, ion radius, chemical valence and affinity. Chemical compounds may be denoted by combined use of such symbols as H_2O, NO, CO_2, and so forth. However, when it comes to writing chemical formulas for the nucleotides and amino acids which are the constituents of DNAs and proteins, there is no need to write down the tens of atomic symbols each time.

Biochemists call the nucleotides a, c, g, t and denote the amino acids by $A, C, \cdots, W, Y$. Now all one has to know is c and g are strongly conjugated by three hydrogen bonds while the weak coupling of a and t is made by two hydrogen bonds. Here "strong" and "weak" differ by many orders of magnitudes from that in high energy physics. In a biochemical pathway or a metabolic network, proteins/enzymes are denoted by simple names and there is no need to spell out the amino acids that make the proteins.

This observation can be continued further. What is the morale learned? In describing Nature one cannot grasp the details on all levels at the same time; one has to concentrate on a particular level at a time treating larger scales as background and reflecting smaller scales in "parameters". For example, in describing the Brownian motion of a pollen the environment at large is represented by a temperature while the friction force is given by using a coefficient of friction. If necessary, one could go down to the molecular level to calculate the coefficient directly. This is called coarse-grained description. Coarse-graining is reached by making "approximations", that is, by ignoring details on finer scales. Nevertheless, it may lead to rigorous conclusions. Geoffrey West, the President of the Santa Fe Institute, once made a remark that had Galileo be equipped with our high precision measuring instruments he would not be able to discover the law of free falling body and would have to write a 42-volume *Treatise on Falling Bodies*.

Furthermore, coarse-grained description of Nature is always associated with the use of symbols. If one is lucky enough these symbols may form symbolic sequences. Coarse-grained description of dynamics leads to symbolic dynamics [2]. Biochemists represent DNA and proteins as symbolic sequences. It is an essential fact that all these sequences are one-dimensional, directed and unbranching chains made of letters from a finite alphabet, thus bringing these sequences into the realm of language theory.

Since we have come to the notion of symbolic sequences, it is appropriate to recollect a basic fact on huge collections of symbolic sequences. In Shannon's seminal 1948 paper [3] that laid the foundation of modern information theory, besides the famous definition of information now familiar to all students, he stated a few other Theorems. Theorem 3 in [3] can be roughly interpreted as follows. Given a sequence of length N made of 0's and 1's, there are in total 2^N such sequences. Generally speaking, when N gets very large, these 2^N sequences can be divided into two subsets: a huge subset of "typical" sequences and a small group of "atypical" sequences. The statistical property of a typical sequence resembles that of any other typical sequence or the bulk of the huge group, while the property of any atypical sequence is very specific and has to be scrutinized almost individually. The simplest members of the atypical set are sequences made of N consecutive 1's or 0's as well as various kinds of periodic and quasi-periodic sequences. However, the most significant ones from the atypical set are those with hidden regularities mixed with seemingly random background. These are the truly complex sequences we have to characterize. While the typical set may be characterized by statistical means, the atypical sequences require more specific method to explore, including combinatorics, graph theory and formal language theory.

One should not be misled by the adjective “formal”. Given the right context, language theory may provide a framework for rigorous description of complexity and workable scheme for down-to-number calculation of characteristics.

1.2 A Brief Introduction to Language Theory

Basic notions of formal language theory may be found in many monographs, for example, [4, 5], and in the comprehensive handbook on formal languages [6]. Therefore, we only give a brief account of some basic notions.

1.2.1 Formal Language

We start from an alphabet Σ made of a finite number of letters. For example, $\Sigma = \{0, 1\}$ or $\{L, R\}$ in symbolic dynamics of unimodal maps [2]; $\Sigma = \{A, C, G, T\}$ when dealing with DNA sequences; in studying protein sequences Σ consists of the 20 single-letter symbols for the amino acids. Collecting all possible finite strings made of letters from the alphabet Σ and including an empty string ϵ, we form a set Σ^*. The empty string ϵ contains no symbol at all, but plays an important role in language theory. Its presence makes formal languages a kind of monoid in algebra [7]. The collection of all non-empty strings is denoted by Σ^+. Obviously, $\Sigma^* = \Sigma^+ \bigcup \{\epsilon\}$.

Now comes the definition of a formal language: any subset $L \subset \Sigma^*$ is called a language. With such a general definition one cannot go very far. The key point is how to specify the subset L. A powerful way to define formal languages makes use of generative grammar. A subset of the alphabet Σ is designated as initial letters. Then a collection of production rules are applied repeatedly to the initial letters and to strings thus obtained. All strings generated in this way form the language L.

Obviously, all strings in the complementary set $L' = \Sigma^* - L$ are inadmissible in the language L. We reserve the term *forbidden word* for members of a special subset of L':

Definition 1.1 A word $x \in \Sigma^+$ is called a *forbidden word* of language L if $x \notin L$ but every proper substring of x belongs to L.

The set of all forbidden words of a language L is denoted by L''. The set L'' may be defined for any language L and may be empty. However, it will play a key role in the study of a special class of languages, namely, factorizable languages, to be defined later in Section 1.2.2.

N. Chomsky classified all possible sequential production rules in the mid 1950s and defined four classes of languages as briefly summarized in Table 1.1. In the order of increasing complexity these classes are regular language (RGL), context-free language (CFL), context-sensitive language (CSL), and recursively enumerable language (REL). Each class of language corresponds to a class of automata with different memory requirement.

Tab. 1.1 The Chomsky hierarchy of languages and automata.

Language	**C**orresponding Automaton	**M**emory
Regular	Finite state automaton (FA)	Limited
Context-free	Push-down automaton	A stack
Context-sensitive	Linearly bounded automaton	Proportional to input
Recursively enumerable	Turing machine	Unlimited

As seen from Table 1.1, each language class corresponds to a class of automata. The finite state automata (FA) that recognize RGL are especially instructive because they may be constructed explicitly in many cases. If a FA has a designated starting state and every state has unambiguous transitions to other states upon reading in a symbol it is a deterministic FA (DFA); otherwise it is called a non-deterministic FA (NDFA). DFA and NDFA are equivalent in their capability to recognize the appropriate language. An NDFA may be transformed into a DFA by means of "subset construction". Among all DFAs accepting the same language there exists one with a minimal number of states. It is called a minimal DFA (minDFA). The size of the minDFA is determined by the index of an equivalence relation R_L generated by the language L in Σ^*. We will give an example of constructing a minDFA in Section 1.5.4. All related notions may be found in standard textbook like [4].

In 1968 the developmental biologist A. Lindenmayer introduced parallel production rules to study the growth of simple multicellular organisms. This approach developed into another framework for the classification of formal languages — the Lindenmayer system or L-system. There are more classes in the L-system: D0L (Deterministic no interaction), 0L and IL (non-deterministic no interaction and with interaction); if the production rules are chosen from a set of rules called a Table the languages become T0L and TIL. In the Chomsky system symbols are divided into terminal and non-terminal ones, the latter being working variables that do not appear in the final products. The L-system has been later extended to include non-terminal symbols to make E0L, ET0L and EIL languages. Referring the interested readers to the monograph [8], we show the relation of various languages in Fig. 1.1[1].

1) We take this opportunity to correct an inexactitude in the original figure on p. 389 of [2].

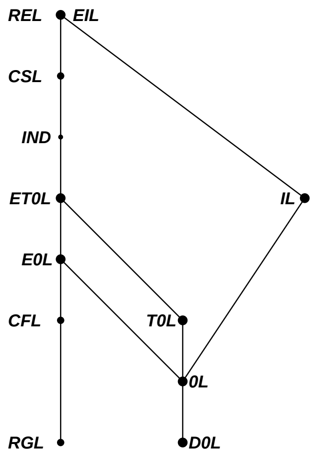

Fig. 1.1 The relation between Chomsky hierarchy and the L-system. Ind denotes *indexed language* not discussed here.

1.2.2 Factorizable Language

In this review we concentrate on factorizable languages which appear in dynamics and some biological applications but have not been mentioned in standard textbook like [4] or the handbook [6]. First, the definition:

Definition 1.2 A language L has *factorial property* or L is called a *factorizable language* if from $x \in L$ it follows that all substrings of x including the empty string ϵ belong to L.

A factorizable language L is determined by the set L'' of forbidden words:

$$\begin{aligned} L &= \Sigma^* - L' &= \Sigma^* - \Sigma^* L'' \Sigma^*, \\ L' &= \Sigma^* - L &= \Sigma^* L'' \Sigma^*. \end{aligned} \tag{1.1}$$

In fact, it follows from the factorizability of L that any string $x \notin L$ must contain at least one forbidden word; contrariwise, putting any number of letters in front or behind a forbidden word must lead to strings not contained in L. A member of the set L'' was called a *Distinct Excluded Block* (DEB) in Wolfram's analysis of grammatical complexity of cellular automata [9].

In order to calculate L'' from L' we need two operators MIN and R acting on any language M and defined in formal language theory [4]:

$$\begin{aligned} \mathrm{MIN}(M) &= \{x \in M \,|\, \text{no proper prefix of } x \text{ is in } M\}, \\ \mathrm{R}(M) &= \{x \,|\, x^R \in M\}, \end{aligned}$$

where x^R means the mirror of x obtained by reversing the string x. Then we have

$$L'' = R \circ MIN \circ R \circ MIN(L'). \tag{1.2}$$

We sketch the proof of Eq. (1.2) by considering a word $x \in L'$. Inspecting x from the start until encountering a forbidden word $v \in L''$, we may write $x = uvz$ where v is the only forbidden word in uv and z denotes all the rest of x after the first forbidden word v. In fact, z may be any string $w \in \Sigma^*$ and uvw represents any member of L'. Now $\mathrm{MIN}(L')$ leaves only strings of form uv where $v \in L''$. Further operations are simple: $\mathrm{R} \circ \mathrm{MIN}(L')$ produces strings of form $v^R u^R$, $\mathrm{MIN} \circ \mathrm{R} \circ \mathrm{MIN}(L')$ strips the latter to v^R and an additional R restores v. Since v may be any member of L' the Eq. (1.2) holds.

However, Eq. (1.2) is rarely needed in practice when the set of forbidden words L'' is known beforehand. Then the language is determined directly from $L = \Sigma^* - \Sigma^* L'' \Sigma^*$ as we shall see in subsequent sections.

1.3 Symbolic Dynamics

Symbolic dynamics [2] arises in coarse-grained description of dynamics. Generally speaking, a dynamics f maps the phase space X into itself:

$$f: \; X \to X.$$

Suppose X is a compact space then there exists a finite covering of X. We label each covering by a letter from a finite alphabet Σ. By ignoring the precise location of a point $x \in X$ and recording only the label of the corresponding covering, the action of the dynamics f corresponds to a shift $\mathcal{S}$ in the space of all strings over the alphabet Σ:

$$\mathcal{S}: \; \Sigma^* \to \Sigma^*.$$

The shift dynamics $\mathcal{S}$ acting on the space of symbolic sequences over the alphabet Σ makes a symbolic dynamics. In this general setting the notion of symbolic dynamics applies to one-dimensional as well as multi-dimensional dynamics, and to conservative as well as dissipative systems. Further specification of f and X enriches the symbolic dynamics as we have learned from symbolic dynamics of one- and two-dimensional mappings and ordinary differential equations [2].

1.3.1 Dynamical Language

Symbolic sequences occurring in a symbolic dynamics may be viewed as a language L. Any symbolic sequence (word) generated by the dynamics is admissible in the dynamical language L. It was recognized in the early study of symbolic dynamics [10] that a dynamical language L has the following two properties:

1. Factorizability: any substring of a word in L also belongs to L, because any part of a longer symbolic sequence is also generated by the same dynamics. This property alone makes L a factorizable language.
2. Prolongability: any word in L may be appended by a letter from Σ to get another word in L due to the time development of the dynamics. Even a "non-moving" fixed point corresponds to adding one and the same letter repeatedly to prolong the symbolic sequence.

1.3.2 Grammatical Complexity of Unimodal Maps

In one-dimensional discrete dynamical systems the unimodal maps of the interval and circle maps are best studied examples. Since these cases have been discussed at length in monographs such as [2] and [5], we only summarize the new knowledge on grammatical complexity of languages in symbolic dynamics of unimodal maps as of the end of 1990s in Fig. 1.2 [11] and Table 1.2.

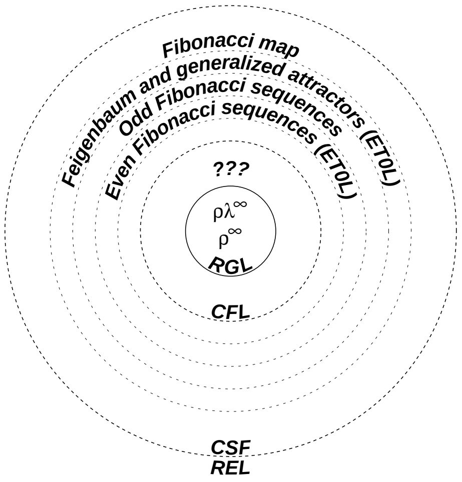

Fig. 1.2 Grammatical complexity of languages in symbolic dynamics of unimodal maps as of the end of the 1990s.

A few explanations follow. Symbolic dynamics of unimodal maps uses two letters R and L to denote the **R**ight and **L**eft parts of the unit interval divided by the critical point C of the map. A special symbolic sequence corresponding to the numerical trajectory starting from the first iterate of C is called a *kneading sequence*. Kneading sequences serve as "topological" or universal parameters of the map that do not depend on the concrete shape of the unimodal mapping function. Among the new results obtained in the 1990s we indicate:

Tab. 1.2 Forbidden words in regular languages of unimodal maps.

Kneading Sequence	**S**et of Forbidden Words L''	**R**emark
RL^∞	Empty set $\emptyset$	A surjective map
L^∞	$\{R\}$	Fixed point only
R^∞	$\{RL\}$	Period 2 appears
$(RL)^\infty$	$\{RLL, RLRR\}$	Period 4 appears
$(RLR)^\infty$	$\{RLL\}$	Period 3 appears
RLR^∞	$\{RL(RR)^n L\}_{n\geq 0}$	First band-merging point

1. In the dynamical languages of the unimodal maps the class of regular languages contains only periodic kneading sequences ρ^∞ and eventually periodic kneading sequences $\rho\lambda^\infty$ [12], where ρ and λ are finite strings made of R and L. Examples of these sequences are given in Table 1.2.
2. Since the above result [12] closes the problem of regular languages in unimodal maps we draw a solid circle around RGL in Fig. 1.2.
3. All attempts to construct context-free languages associated with unimodal maps, including various Fibonacci sequences [13] have led to context-sensitive languages which are not context-free. Hence the as yet open conjecture: there is no context-free language which is not regular in unimodal maps [5], a fact represented by the three question marks in Fig. 1.2.
4. The kneading sequence of the infinite limit of the Feigenbaum period-doubling cascade may be obtained by infinitely repeated application of the homomorphism $h = (R \to RL, L \to RR)$ to the single letter R. It was the only known context-sensitive language in the unimodal maps in the beginning of 1990s, first proved to be a non-context-free language [14], in which the language consists of the set of all substrings of the kneading sequence only. Now the class of context-sensitive language has split into layers as shown in Fig. 1.2. All these layers are not empty and may be characterized precisely in mathematical terms [15].

1.4 Sequences Generated from Cellular Automata

The research of cellular automata were originated from Von Neumann's study of formalizing the self-reproduction feature of life in 1950s [16], and popularized by The Game of Life invented by Conway in 1970s [17, ch.25]. The systematic study of cellular automata, however, was begun in 1980s in a series of works by S. Wolfram et al, in which many new points of view and tools were introduced. It is evident that the ideas and results from nonlinear science play an important role in cellular automata, including numerical experiment by computer, statistical method, algebraic method and the method of formal languages and automata from the theoretical computer science.

Two collections of papers are good references for an introduction into the area of cellular automata [18,19]. For the arguments caused by Wolfram's new book [20] in 2002 many materials can be found from Website, for example, `http://www.math.usf.edu/~eclark/ANKOS_reviews.html`.

In this section we will consider the study about sequences generated by one-dimensional cellular automata. In Subsection 1.4.1 an introduction is given, including some definitions and simple facts about cellular automata. Subsections 1.4.2 and 1.4.3 are respectively devoted to two kinds of complexity of sequences generated by cellular automata, namely the limit complexity and the evolution complexity. The languages involved in these studies are always factorizable languages.

1.4.1 Encounter the Cellular Automata

The complex systems in Nature are often composed by coupling many simple systems, and *Cellular Automata (CA)* are one of ideal mathematical models for those complex systems.

The definition of one-dimensional CA can be given as follows.[2]

Assume that there are infinite cells, namely automata, situated on all integer points of the number axis, each cell can only take finite states, and the time variable is discrete, namely, $t = 0, 1, 2, \dots$. It seems true that these discrete features of states and time are similar with those features of many automata in theoretical computer science [4]. The distinct feature of CA, however, is that here there are infinitely many cells, and they change their states simultaneously at each discrete time. In addition, the state of each cell in the next time is determined by the states of itself and some cells nearby through some definite rules specified by a given CA, and these rules are all the same for each cells and each time thenceforth.

Hence the evolutionary rules of CA are homogeneous both for space and time. From the point of view of computer, CA belong to a class of parallel computers.

Now the above definition can be rephrased by mathematical language as follows. Assume that each cell has $k > 1$ states, and denote these states by the symbols $0, 1, \dots, k-1$, hence the alphabet set is $S = \{0, 1, \dots, k-1\}$. let $a_i \in S$ be the state of the cell posited at the integer $i \in \mathbb{Z}$, the set of all integers, and the state of CA at each time be a bisequence over S:

$$a = \cdots a_{-n} \cdots a_{-2} a_{-1} a_0 a_1 a_2 \cdots a_n \cdots , \tag{1.3}$$

2) Both CA in [16,17] are two-dimensional.

which is called a *configuration* of CA, and $S^{\mathbb{Z}}$, the set of all bisequence, the configuration space of CA.

Furthermore, if the states of cells at time t are denoted by a_i^t, $i \in \mathbb{Z}$, then the evolutionary rule of states can be written in the form of

$$a_i^{t+1} = f(a_{i-r}^t, \cdots, a_i^t, \cdots, a_{i+r}^t), \ \forall\, i \in \mathbb{Z}, \tag{1.4}$$

in which the number r is called the radius of neighborhood. The existence of such r reflects the finiteness of information transmission velocity in CA.

The simplest CA are those CA with the number of states $k = 2$ and the radius of neighborhood $r = 1$, which are conventionally called the *elementary cellular automata (ECA)*.

Since for ECA the alphabet set being $S = \{0, 1\}$, the rule f in the definition of (1.4) is a mapping from S^3 to S. As the arguments of this mapping have only 8 possibilities, namely, $000, 001, \cdots, 111$, hence an ECA is completely given as long as the values of f are given on these 8 arguments. A consequence of this consideration is that there are altogether $2^{2^3} = 256$ possible ECA.

A popular coding scheme for ECA can be explained below (see [18]).

For example, let an ECA be given by the rules

$$011, 100, 101 \longrightarrow 1; \quad 000, 001, 010, 110, 111 \longrightarrow 0, \tag{1.5}$$

namely $f(011) = 1$, $f(100) = 1$, and so forth. Rearranging the rules (1.5) as follows

111	110	101	100	011	010	001	000
0	0	1	1	1	0	0	0

and then to convert the binary number 00111000 in the second line to the decimal number 56, and call the ECA thus given the ECA of rule 56.

It is convenient to see the space-time behavior of one-dimensional CA on computer's screen. Figure 1.3 is the experimental results performed for the space-time behavior of 8 different ECA.

In this Figure the rule numbers of each ECA are put on the top of each subfigure. Every subfigure is obtained by the same procedure as follows. Let the small white square be the symbol 0, and the small black square the symbol 1. The first line is (a part of) the starting configuration at the time $t = 0$, which is generated by a pseudorandom binary generator. The direction of time is from top to bottom. Each of the other lines is obtained from the previous line by the rule of ECA. For each subfigure of Figure 1.3 the number of time steps is 100, and the length of the part of starting configuration is 300. Since only the center part of width 100 is seen, the trouble of appropriate boundary effect is removed. (Another experiment method not discussed here for one-dimensional CA is to use the circular boundary condition, and it can be seen as if all cells are arranged on a circle.)

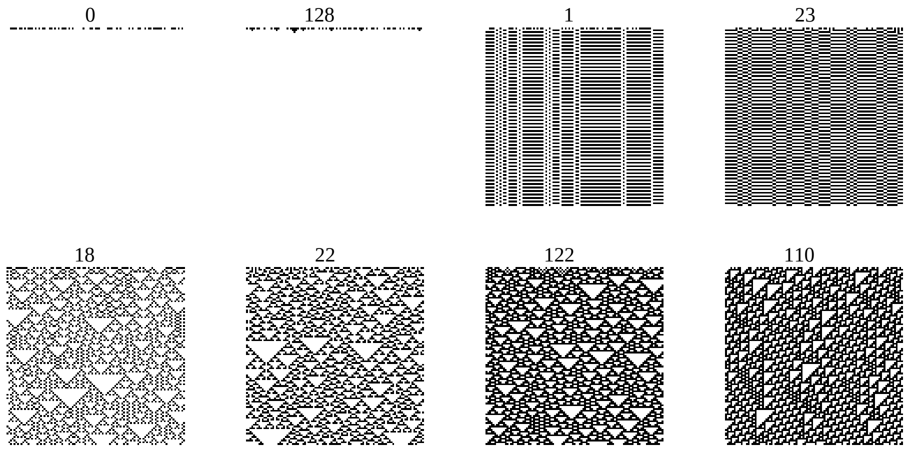

Fig. 1.3 The space-time behaviors of 8 ECA of different rules

Wolfram did many computer experiments for one-dimensional and two-dimensional CA, including all 256 ECA, and proposed a classification scheme for all CA as follows:

1. all configurations converge to an invariant homogeneous configuration composed of all the same symbols,

2. converge to some simple stable or periodic structures,

3. chaotic and non-periodic behaviors appeared,

4. complex local structures appeared, and some of them can irregularly propagate.

As pointed by Wolfram that all these 4 kinds of behaviors can be seen in ECA.

Of course this classification scheme is only a phenomenological one, and more rigorous analysis is absolutely needed. In Figure 1.3 all four kind of behaviors are shown, but there exist many questions which cannot be removed by pure experiment on computer.

For example, the first ECA of rule 0 in this Figure, namely the mapping from S^3 to the state 0, belongs to the class 1, since each configuration from $t = 1$ composed by the same symbol 0s, which will be denoted by $\overline{0}$ hereafter. But it is more difficult for the next one, the ECA of rule 128, since nearly any experiment with this ECA shows that the configuration will rapidly converge to $\overline{0}$ as for the ECA of rule 0, but can we say definitely that the ECA of rule 128 belongs to class 1 in Wolfram's classification scheme? We will come back to this question in Subsection 1.4.2.

Another factor is that different starting configuration can lead to different space-time behaviors. An example is shown in Figure 1.4. The left subfigure is obtained from a random starting configuration, and the behavior looks like those in the class 2. The starting configuration for the middle subfigure is a random sequence of 00 and 11, and the behavior looks like those in the class 3. The right subfigure is a combination of those of the two previous subfigures.

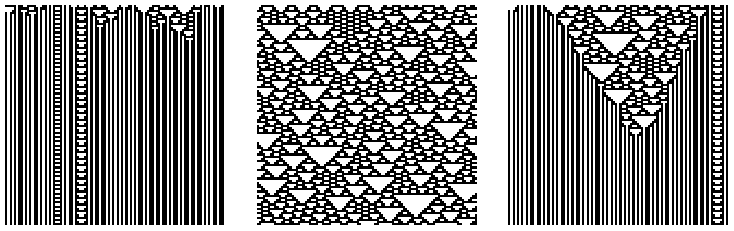

Fig. 1.4 Space-time behaviors of ECA of rule 94 for different starting configurations

There are many studies for the ECA, but still it cannot say that any complete theory has been established. An example is the ECA of rule 110 in Figure 1.3. It has been believed that it belongs to the class 4, and only recently an important result is obtained that the ECA of rule 110 has the power of universal Turing machine [20, 21].

The remaining part of this Section is devoted to a review of research of sequences generated from CA by the method of formal languages.

1.4.2 Limit Complexity of Cellular Automata

The introduction of the method of formal languages to CA began mainly from Wolfram's paper [9], in which many conjectures were made based on the experiment on computer. After that some theoretical results are obtained by Hurd [22, 23], in which and also in other papers the concept of limit set play an important role. The complexity analyzed by this method will be called the *Limit Complexity of CA*.

First define both the limit set of CA and the limit language generated from the limit set.

The rule f in (1.4) is also called the *local mapping* of CA. Using f onto a configuration will generate a new configuration, and hence establish a *global mapping* F from the configuration space into itself. We also use the notation, for example, f_{56} and F_{56} to represent the local and global mapping for the ECA of rule 56.

Using F iteratively generates a dynamical system, and the configuration space, $S^{\mathbb{Z}}$, becomes the phase space for this dynamical system. The difference between this dynamical system generated by the global mapping of CA and the low-dimensional discrete dynamical systems is that the former one is an infinite-dimensional dynamical systems. Nevertheless, many mature concepts from dynamical systems can be borrowed for study of CA, for example, invariant sets, periodic points, trajectories, and so forth. An important feature of CA is that we have to consider its space structure with its time structure simultaneously, namely, the space-time behaviors.

Consider the phase space $S^{\mathbb{Z}}$ more closely. Since $S^{\mathbb{Z}}$ is a product space generated from the alphabet set S, considering the discrete topology to S, and using the Tychonoff theorem, $S^{\mathbb{Z}}$ is a compact space. Moreover, this compact topology can also be metrizable, for example, by introducing the distance between $x, y \in S^{\mathbb{Z}}$ as follows: let $d(x,y) = 0$ if $x = y$, and

$$d(x,y) = \frac{1}{k+1},$$

where the number k is the minimal non-negative integer such that $x_k \neq y_k$ or $x_{-k} \neq y_{-k}$. Since it can be shown that the global mapping F is continuous on the space $S^{\mathbb{Z}}$, the image set of $F(S^{\mathbb{Z}})$ is also a compact set.

From the inclusion relation $F(S^{\mathbb{Z}}) \subset S^{\mathbb{Z}}$ it is evident that $F^{n+1}(S^{\mathbb{Z}}) \subset F^n(S^{\mathbb{Z}})$ holds for every $n \geq 0$. By the theorem about the nonemptyness of the intersection of non-increasing sequence of compact sets, the set

$$\Lambda(F) = \bigcap_{n=0}^{\infty} F^n(S^{\mathbb{Z}}) \tag{1.6}$$

is non-empty, and called the *limit set* of CA hereafter. It can be seen that the limit set contains all periodic points and non-wandering points of F.

From the point of view of dynamical systems, it is evident that the limit set is just the largest invariant set of the global mapping F. But the limit set of CA still has its special features as shown by the following theorem [24, p.373]:

Theorem 1.3 *If the limit set of a CA has more than one element, then it must be an infinite set.*

The simplest case is the limit set which has only one element: $\Lambda(F) = \{c\}$. Since any shift of element in limit set is still an element of it, the configuration c must be a space homogeneous one, namely composed by a single symbol q. Considering that the configuration c must also be an invariant point of F, then it is certain that a rule of form as $qq \cdots q \to q$ must be in f of (1.4). We call F a *nilpotent CA*, if its limit set contains only one element. It corresponds the class 1 in Wolfram's classification scheme. But it turns out that even here some difficulty is unavoidable. It is proved already that the problem of deciding a given CA being nilpotent is an undecidable problem, that is to say there exists no algorithm to give the answer of "yes" or "no" for the problem of deciding whether a given CA being nilpotent or not. This result has been proved for some time for CA with dimension more than 1, and for one-dimensional CA lately in [25].

In order to reduce the study of bisequences to the study of finite sequences, a language will be generated from the limit set.

For a subset A of the compact space $S^{\mathbb{Z}}$, taking all finite subsequences for each bisequence of A gives a formal language over S denoted by the notation $\mathcal{L}(A)$. It can be shown that, if the set A is shift-invariant and closed, then it can be determined by the language $\mathcal{L}(A)$ completely [5].

The rest of this Subsection is devoted to the study of limit language of CA, and its complexity in the Chomsky hierarchy [4].

The first result in this aspect is the following theorem. we will give its proof, and referred the reader to the textbook [4] for the concepts and tools used below.

Theorem 1.4 *If F is the global mapping of a CA, then for each $n \geq 0$, the language $\mathcal{L}(F^n(S^{\mathbb{Z}}))$ is regular.*

Proof. Since for each n the mapping F^n is also a CA, it suffices to prove the conclusion for the case of $n = 1$. The method used below is to construct the finite automaton accepting the language $\mathcal{L}(F(S^{\mathbb{Z}}))$ exactly.

Let the radius of neighborhood of F be r, and the number of states be k. Taking all words of length $2r$ as the accepting states of finite automaton, then there are k^{2r} states altogether.

The transition rule of states can be determined as follows. Observing the local mapping f as shown in (1.4), every rule

$$f(s_1 s_2 \cdots s_{2r+1}) = s_0$$

can be interpreted as a transition from the state $s_1 s_2 \cdots s_{2r}$ to the state $s_2 s_3 \cdots s_{2r+1}$ if the symbol s_0 is read. In other words, between these two states there exists an arc labeling by s_0 as shown by

$$\boxed{s_1 s_2 \cdots s_{2r}} \xrightarrow{s_0} \boxed{s_2 s_3 \cdots s_{2r+1}}\,.$$

It is easy to see that this finite automaton accepts exactly the language $\mathcal{L}(F(S^{\mathbb{Z}}))$. □

The limit language has the similar expression as (1.6) that

$$\mathcal{L}(\Lambda(F)) = \bigcap_{n=0}^{\infty} \mathcal{L}(F^n(S^{\mathbb{Z}})). \tag{1.7}$$

Hence from Theorem 1.4 it is known that the limit language is the intersection of countable regular languages, but it cannot tell us that how complex the limit language is. In Hurd's papers [22,23] many CA are constructed such that their limit languages can have all possible grammatical complexity in Chomsky hierarchy.

From the expression of (1.7) it is a consequence that a sequence $s \in \mathcal{L}(\Lambda(F))$ if and only if $s \in \mathcal{L}(F^n(S^{\mathbb{Z}}))\ \forall\, n = 1, 2, \cdots$, and verifying this is usually a hard work, if not impossible.

Now consider the limit languages of ECA. As discussed above that all nilpotent CA belong to the class 1 in Wolfram's scheme, and as shown in figure 1.3 the ECA of rule 0 is one of them. But what about the ECA of rule 128 in this Figure? Here the experiment on computer can easily lead to the conjecture that this ECA is also one of class 1, since each experiment shows that it converges to all 0's rapidly. Examing its local mapping

111	110	101	100	011	010	001	000
1	0	0	0	0	0	0	0

means that the configuration of all 1's also belongs to the limit set, and from Theorem 1.3 the latter must be an infinite set.

It is not hard to tackle directly the limit language of ECA of rule 128, and obtain the result that

$$\mathcal{L}(\Lambda(F_{128})) \ = 0^*1^*0^* = \{0^m1^n0^p \mid m, n, p \geq 0\}, \tag{1.8}$$

a regular language. Since it is a factorizable language, the concept of forbidden words (or distinct excluded blocks) [5,26] can be used to it and it is easy to find out all forbidden words for this language (1.8) being $\{10^n1 \mid n \geq 1\}$.

In [9] a measure of complexity for regular languages is proposed and applied to characterize the behavior of CA. For a regular language L, define the *regular language complexity* of L as the number of states of *the minimal deterministic finite automaton* (minDFA) which accepts L. In [18] the regular language complexity of $\mathcal{L}(F^n(S^{\mathbb{Z}}))$ for $n = 1 \sim 5$ and 256 ECA are computed and listed as the table 10 of it.

It turns out that for most ECA belonging to the class 1 or 2, their regular language complexity are either constant or increasing slowly; but for ECA belonging to the class 3 or 4, their regular language complexity are increasing rapidly such that in many cases the computation of regular language complexity cannot be performed even for $n = 4$ or 5.

But this method is still not a rigorous and reliable approach to decide the complexity level of limit languages. In Table 1.3 these results are listed for some of ECA, whose complexity is already known.

The limit language of the ECA of rule 94 is proved to be non-regular [27] and this result coincides with the data in Table 1.3. However, we can compare these conclusion with the space-time behavior of ECA of rule 94 shown in Figure 1.4, in which the left subfigure is the behavior obtained from a random starting configuration, and it seems that the configuration converges to

Tab. 1.3 Regular language complexity of some ECA for $n = 1 \sim 5$

ECA	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$
94	15	230	3904		
22	15	280	4506		
122	15	179	5088		
104	15	265	2340	1394	1542
164	15	116	667	1214	
110	5	20	160	1035	

a period 6 attractor. This comparison reveals that sometimes the experiment method is not satisfactory and reliable.

The conclusion about the ECA of rule 22 is similar and proved in [28], but its space-time behavior as shown in Figure 1.3 is more complex than that of ECA of rule 94.

As it can be seen from Figure 1.3 that the space-time behavior of the ECA of rule 122 is similar to that of ECA of rule 22. Here the theoretical analysis gives better result that its limit language, $\mathcal{L}(\Lambda(F_{122}))$, is not a context-free language [29].

If it is true that the data in Table 1.3 give the correct evidence of complexity for the first three ECA, those of rules 94, 22, 122, then it is not so for the next two ECA, those of rules 104 and 164. As proved in Jiang's doctoral thesis, both limit languages of these two ECA are regular, although their regular language complexity are very high indeed. However, seeing their data in Table 1.3, it seems that both ECA of rules 104 and 164 may be more complex than the ECA of rule 110, but it is known that the latter is the most complex ECA as proved in [20,21].

The most unsuccessful cases of the application of limit languages happen for the surjective CA, in which it is evident that

$$F(S^{\mathbb{Z}}) = S^{\mathbb{Z}}, \quad \Lambda(F) = S^{\mathbb{Z}}, \quad \mathcal{L}(\Lambda(F)) = S^*. \tag{1.9}$$

That is to say the configuration space $S^{\mathbb{Z}}$ itself, namely, the phase space, is invariant under the global mapping F, and the limit language is the largest language over the alphabet S. Its regular language complexity is simply 1, since there is only one accepting state as it can accept every sequence over S.

There are 30 ECA belonging to this category, and some of them are shown in Figure 1.5 below.

It is evident that the 4 surjective ECA in the first line of Figure 1.5 are very simple. The ECA of rule 204 is the identity mapping, each configuration is a fixed point of it; and the ECA of rule 51 is the flip-flop mapping, its f_{51} maps 0 to 1 and vice versa, each configuration is a period-2 point of it. The real radius

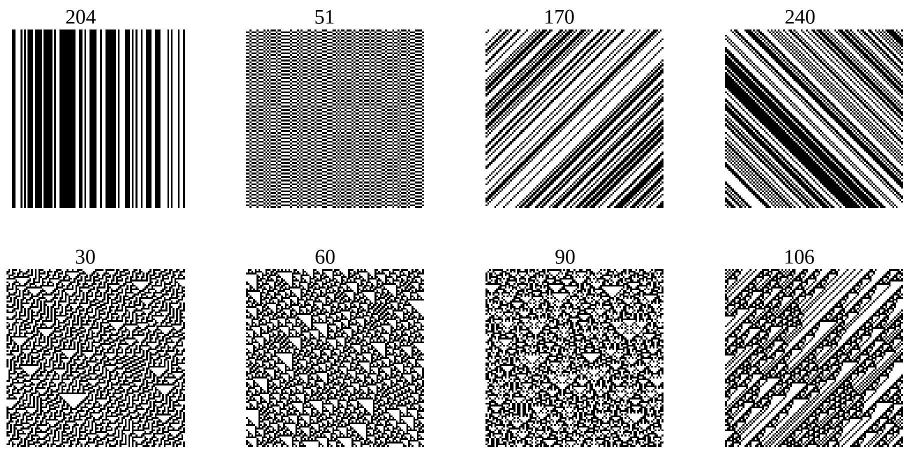

Fig. 1.5 The space-time behaviors of 8 surjective ECA

of neighborhood of these two ECA is $r = 0$. It is true that both of them belong to the class 2 in Wolfram's classification scheme. The ECA of rules 170 and 240 are respectively the left shift and right shift.

It is also evident that the 4 surjective ECA in the second line of Figure 1.5 show much more complex behaviors than those ECA of first line there, and belong to the class 3 in Wolfram's classification scheme. The ECA of rule 30, among them, is used as a pseudorandom number generator in the software of Mathematica, and there are many discussion about the behaviors of ECA of rules 60 and 90. The ECA of rule 106 contains the leftshift ECA of rule 170 as its factor system, but is much more complex than it.

Summary. Although there exist many results and discussion about the limit set and limit language of CA, but no universal method or result have been found. Computer experiment method is useful for one-dimensional CA, but its result need careful exploration and theoretical research.

The classification problem of CA has drawn much attention after Wolfram's scheme, and several rigorous classification schemes are obtained [30, 31]. But there are often drawbacks in these schemes that (1) most complex CA are classified into one class, and not much help is provided for their analysis; (2) there is no effective criteria to determine which class a given CA belongs to.

It seems that there are still many open problems in the area of CA. It is also true even of ECA, the simplest class of CA.

1.4.3 Evolution Complexity of Cellular Automata

From the content of Subsection 1.4.2 it seems that the approach of limit language of CA has its restriction, and is especially unsatisfactory for the surjective CA.

Many examples, which include the above-mentioned surjective CA, show that there exist no direct relationship between the limit set (or limit language) and the space-time behavior of CA. It is not surprising about it if recalling the fact that the limit set is just the largest invariant set of the dynamical system of CA.

Another approach for analyzing the complexity of CA using formal languages is proposed by Gilman [32]. The complexity analyzed by this approach will be called the *Evolution Complexity of CA* hereafter.

The evolution complexity consider the evolution of one cell, a_i, or some cells, $a_i, a_{i+1}, \ldots, a_{i+k-1}$, in which case we say its width is k. From the space homogeneous feature of CA, the index i can be fixed as 0 or other number. Therefore, this approach is very similar to look at the computer screen for CA's evolution with a fixed width of cells, that is a fixed width of window, and it can be expected that the evolution complexity of CA should have more direct relationship with the space-time behavior of CA obtained by computer experiment.

Considering that the configuration of CA is a bisequence, which is infinite in both directions, it is natural that people can only see its finite part and take the evolution of this finite strip as the basis for analyzing its complexity. This idea is another example of coarse-graining mentioned before.

Assuming the width of observing window is k, then the object of our study can be seen as a sequence over the alphabet set S^k. Taking all possible finite sequences together gives the *evolution language of width k* of the CA, which will be denoted by the notation $\mathcal{E}_k$. For the case of $k = 1$ the name of *time series* of CA is also used to describe the sequence over S, which is a record of one cell's evolution over some time interval.

For a given CA and a given sequence s over S^k, it can be determined whether s belonging to $\mathcal{E}_k$ by using the local rule (1.4) for a fixed number of cells of CA, and this number is a linear function of k. Therefore, this is a problem which can be solved by a restricted kind of Turing machines, namely, the linear bounded automata. Using the relation between automata and Chomsky hierarchy a theorem is obtained that the grammatical complexity of evolution language will not beyond the *context-sensitive languages* [4,32,33].

An example of CA is given by Gilman, in which the number of states is $k = 2$, and the radius of neighborhood is $r = 2$, hence it is not an ECA. The local rule of this CA is given by

$$a_i^{t+1} = f(a_{i-2}^t a_{i-1}^t a_i^t a_{i+1}^t a_{i+2}^t) \; = a_{i+1}^t a_{i+2}^t \quad \forall\, i \in \mathbb{Z}, \tag{1.10}$$

where the right-hand side is the multiplication of states a_{i+1}^t and a_{i+2}^t which are binary numbers as well as symbols[3].

The space-time behavior of Gilman's CA is shown in Figure 1.6:

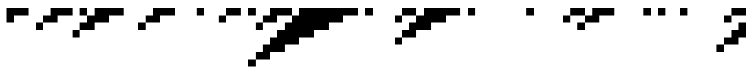

Fig. 1.6 The space-time behavior of Gilman's CA

It has been proved that the evolution language $\mathcal{E}_1$ of Gilman's CA is a context-sensitive language, but not a context-free language (see [32, p.99] and [33, p.429]).

The time series also appears in [34,35], in which the problem of periodicity of time series of ECA is discussed, in which $f(000) = 0$ is satisfied and there exist only finite symbol 1 in the starting configurations.

For the ECA of rule 18, the complexity of its evolution languages of each width has been completely solved. In [36] it is proved that for this ECA the language $\mathcal{E}_1$ is regular, and all $\mathcal{E}_k$ ($k \geq 2$) are context-sensitive, but not context-free. Similar results are obtained recently for the ECA of rule 146 [37].

For the ECA of rule 22, it is proved that all $\mathcal{E}_k$ ($k \geq 2$) are not regular [38], but it is open for $\mathcal{E}_1$.

Comparing the limit complexity and the evolution complexity of the same CA, there exists no direct relationship between them. For example, the evolution complexity of Gilman's CA is the highest level possible in Chomsky hierarchy, but its limit language is very simple, since it can be proved that for Gilman's CA, $\mathcal{L}(\Lambda(F)) = 0^*1^*0^*$, the same language as that of ECA of rule 128 mentioned before.

Another example in this aspect is the ECA of rule 90. As pointed out above that it is a surjective CA, which has been studied extensively before. We know that its limit language is $\{0,1\}^*$, its regular language complexity is 1, and cannot reflect its rather complex space-time behavior as shown in Figure 1.5. Generally, the ECA of rule 90 is considered a typical CA in class 3 of Wolfram's classification scheme.

It is easy to show that for the ECA of rule 90, each $\mathcal{E}_k$ ($k \geq 1$) is regular. Let their regular language complexity be $C(\mathcal{E}_k)$, then it can be verified that

$$C(\mathcal{E}_1) = C(\mathcal{E}_2) = 1,\; C(\mathcal{E}_k) = 2^{k-2} + 2\,\forall\, k \geq 3, \tag{1.11}$$

3) Although the CA of Gilman has the radius of neighborhood $r = 2$, but its a_i^{t+1} is determined only by a_{i+1}^t and a_{i+2}^t, and, therefore, this CA can be obtained by the composition of the ECA of rule 136 and rule of 170, the left-shift operator.

in which the first two results are equivalent to say that

$$\mathcal{E}_1 = \{0,1\}^*, \mathcal{E}_2 = \{00,01,10,11\}^*.$$

As an example of the results in (1.11), the minDFA accepting the evolution language $C(\mathcal{E}_3)$ of the ECA of rule 90 will be constructed below.

In Figure 1.7 it is shown a DFA which contains 3 accepting states q_0, q_1, q_2, and 1 non-accepting state q_3.

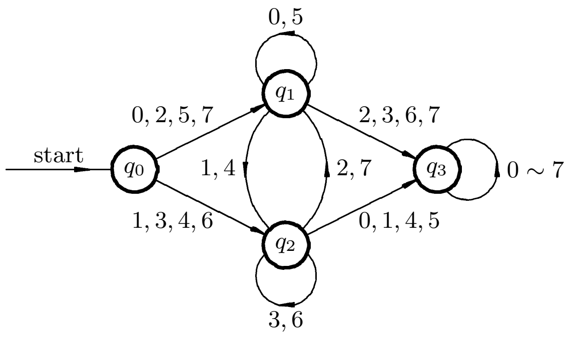

Fig. 1.7 A minDFA accepting the evolution language of width 3 of the ECA of rule 90

The alphabet set of the language $C(\mathcal{E}_3)$ is the set of all words of length 3, namely,

$$S = \{000,001,010,011,100,101,110,111\},$$

and its elements are denoted by $0 \sim 7$ in Figure 1.7. The local mapping of ECA of rule 90 is

$$f(a_{-1}, a_0, a_1) \equiv a_{-1} + a_1 \pmod 2,$$

and it is a straightforward verification that the automaton in Figure 1.7 is the required one accepting $C(\mathcal{E}_3)$.

In the sequel a recent work is reviewed in which the evolution complexity of width 1 for all 256 EAC are explored and some interesting results obtained [39,40].

Since the evolution language $\mathcal{E}_1$ is factorizable, hence it can be characterized by its forbidden words [5,26]. Using a C++ program as a searching tool, we can use computer to find all forbidden words whose length are not beyond a certain limitation, then a theoretical analysis is followed to see whether the whole set of forbidden words can be determined.

Since the complexity of space-time behavior is not influenced by the exchange of 0, 1 and the mirror image of the neighborhood, the 256 of ECA can be reduced to the 88 classes of ECA, and each class thus formed is represented by the ECA in the class with the minimal rule number [18]. Hence we need only to consider 88 ECA.

The result is shown in Table 1.4, in which all 88 ECA are classified into 4 classes according the complexity of $\mathcal{E}_1$ for each ECA[4].

Tab. 1.4 The classification of ECA on the basis of complexity of evolution languages $\mathcal{E}_1$

Class	No.	Rule number of ECA	No. of FW	Complexity
I.1	10	15, 30, 45, 60, 90, 105, 106, 150, 154, 170	none	$\{0,1\}^*$
I.2	4	94, 122, 126, 184	none	$\{0,1\}^*$
II	53	0, 1, 2, 3, 4, 5, 6, 8, 10, 11, 12, 13, 14, 18, 19 23, 24, 29, 32, 34, 35, 36, 38, 40, 42, 43, 44 46, 50, 51, 57, 58, 72, 76, 77, 108, 128, 130 132, 136, 138, 140, 142, 146, 152, 160, 162 168, 172, 178, 200, 204, 232	≤ 5	FCR
III	8	27, 28, 33, 78, 104, 134, 156, 164	infinite	ICR
IV	13	7, 9, 22, 25, 26, 37, 41, 54, 56, 62, 73, 74, 110	infinite	nonRGL

Class I includes all ECA whose $\mathcal{E}_1$ has no forbidden words at all, namely, their $\mathcal{E}_1 = S^*$, and the above-mentioned ECA of rule 90 belongs to this class. Moreover, Class I is divided into two subClass I.1 and I.2, the difference is that every ECA in I.1 is surjective, but the ECA in I.2 is not.

The ECA in Class II has only finite forbidden words, and the number of them is in the range of $1 \sim 5$. Hence their evolution language $\mathcal{E}_1$ is finite complement regular, and it can be pointed that the symbolic flow associated with them is of the *subshift of finite type (SFT)* [5, p.23].

In Class III the evolution language of each ECA has infinite forbidden words, but still is regular, and, therefore, their $\mathcal{E}_1$ is infinite complement regular. The associated symbolic flow is the so-called *sofic system* [5, p.25].

Finally, the theoretical study of the ECA in Class IV has not been finished yet, for some of them the evolution language $\mathcal{E}_1$ is proved to be context-free, but not regular, and for others we have only some feeling that their evolution languages $\mathcal{E}_1$ are non-regular, and maybe much more complex than those ECA which are known already.

This conjecture is supported by the data in Table 1.5, in which the numbers of forbidden words whose length are not beyond 17 are listed there.

Finally, a theorem about the evolution language of the ECA of rule 56 in Class IV is cited below, which shows that an interesting structure appears in the time series generated by this ECA [40].

4) The meaning of some abbreviations in Table 1.4 are as follows. FW, forbidden words; FCR, finite complement regular language; ICR, infinite complement regular language; nonRGL, non-regular language.

Tab. 1.5 The numbers of forbidden words of length $K = 2 \sim 17$ for ECA in Class IV

Rule	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Sum
7	0	0	3	2	0	2	0	2	0	2	0	2	0	2	0	2	17
9	0	1	1	2	0	2	3	3	3	3	3	4	7	6	14	10	62
22	0	0	1	1	0	0	1	1	2	5	1	7	26	54	78	153	330
25	0	1	1	1	0	3	2	3	2	2	4	3	5	2	14	15	58
26	0	1	0	0	1	0	0	0	2	2	4	7	16	19	35	44	131
37	0	0	0	3	4	5	3	10	9	23	20	22	32	36	37	66	270
41	0	1	0	0	0	1	1	1	6	2	14	30	51	90	146	253	596
54	0	1	1	0	0	1	4	2	0	4	6	15	18	34	32	62	180
56	0	1	1	0	0	1	0	0	2	0	0	5	0	0	14	0	24
62	0	1	0	1	2	4	4	3	6	5	6	6	10	3	16	12	79
73	0	0	2	2	1	2	2	1	2	3	3	8	11	13	18	27	95
74	0	0	3	0	1	2	1	0	0	0	1	2	4	7	11	16	48
110	0	0	0	1	3	3	5	6	12	16	17	38	42	73	112	198	526

Theorem 1.5 *Let $\mathcal{E}_1$ be the evolution language of width 1 of the ECA of rule 56, then (1) the languages $\mathcal{E}_1$ and $\mathcal{E}_1''$, the set of its forbidden words, are both context-free, but not regular; (2) the set $\mathcal{E}_1''$ can be expressed explicitly by*

$$\mathcal{E}_1'' = \{111\} \cup 0D01,$$

in which D is the Dyck language generated by the strings 0 and 01; (3) the number of forbidden words whose length is $3n+1$ ($n \geq 2$) is C_{n-1}, the $n-1$-th Catalan number[5].

1.5 Avoidance Signature of Bacterial Complete Genomes

DNAs are one-dimensional, directed, non-branching heteropolymers made of four kinds of monomers – the nucleotides adenine (a), cytosine (c), guanine (g), and thymine (t). In 1995 the first two complete genomes of free living bacteria were published. By the end of 2006 more than 430 bacterial genomes were available in public databases. Having a genome at one's disposal many global questions may be asked. A biochemist would wish to infer all possible

5) The Dyck language is one of the most important context-free languages [4, p.142], and the Catalan numbers appear in many interesting problems of combinatorics, their expression is given by

$$C_n = \frac{1}{n+1}\binom{2n}{n}, \; n \geq 1.$$

The first Catalan numbers are $1, 2, 5, 14, 42, 132, 429, \cdots$ [41].

metabolic pathways underlying the life of a particular bacterium. A physicist without sound biological knowledge might ask the simplest global question concerns the distribution of short nucleotide strings of a fixed length K, in particular, whether some strings are absent at a given K.

1.5.1 Visualization of Long DNA Sequences

The length of a typical bacteria genome is of a few millions nucleotides. In order to visualize the K-string composition of a genome we apply a simple counting algorithm. To count the number of K-strings we allocate the 4^K counters on the computer screen as a direct product of K copies of 2×2 matrix M:

$$M \otimes M \otimes \cdots \otimes M,$$

where

$$M = \begin{pmatrix} g & c \\ a & t \end{pmatrix}.$$

We use 16 colors to represent the counts. If a string is absent the corresponding cell is shown in white. The bright colors are assigned to small counts. If the counts are greater than a certain threshold, say, 40, the cell is shown in black. This is also a kind of coarse-graining. A program entitled SEEDNA has been put in the public domain [42]. We call the output of this program a "portrait" of the bacterium.

1.5.2 Avoided K-Strings in Complete Genomes

A portrait of the harmless laboratory strain K12 of *E. coli* is shown in Fig. 1.8 for $K = 8$. The almost regular pattern seen in Fig. 1.8 tells the under-representation of strings containing *ctag* as substring. In fact, closely related species have similar pattern of under-represented strings and bacteria from different taxa show some characteristic “avoidance signature”. For more discussion see [43]. A portrait is nothing but a two-dimensional histogram of the string counts. The string counts may be visualized by using one-dimensional histogram as well. The latter may show some peculiar fine structure for a few randomized genomes. The explanation requires combination of simple combinatorics with statistics [44].

1.5.3 True and Redundant Avoided Strings

By inspection of Fig. 1.8 or, to be more precise, by direct counting, we see that at $K = 8$ there are 173 string types missing in the *E. coli* K12 genome. At $K = 7$

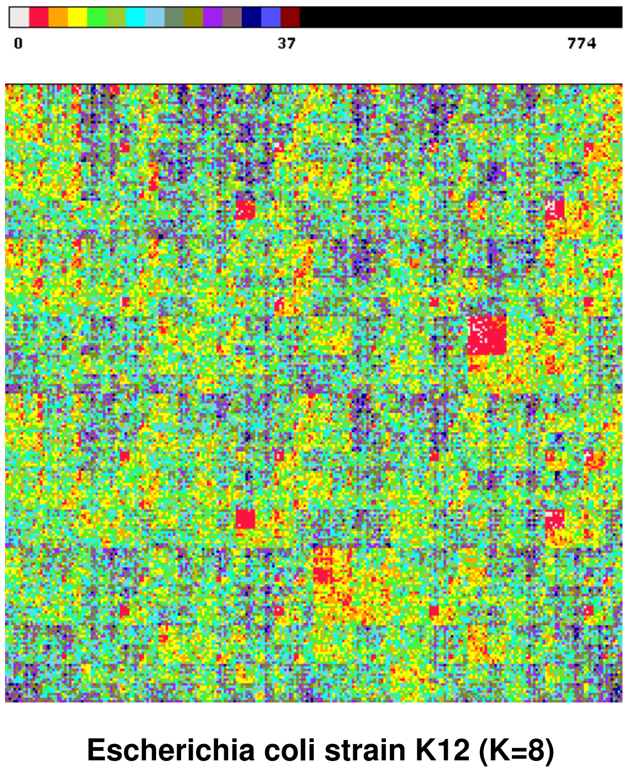

Fig. 1.8 A "portrait" of *E. coli* strain K12 at $K = 8$.

there is only one missing string, namely, *gcctagg*. This simple fact raises a question. Among the 173 missing strings at $K = 8$ eight strings must be the consequence of the string *gcctagg* being absent at $K = 7$, because one may add a letter in front or at the end of the string to get 8 strings that cannot appear at $K = 8$. We say that at $K = 8$ there are 8 redundant and 165 true missing strings in the genome. Given that at length K there is one missing string, one would like to know how many redundant missing strings it produces at $K + i$. By mathematical induction one gets a simple formula as the answer:

$$N_i = 4^i(i+1), \quad i = 0, 1, \cdots \tag{1.12}$$

However, this formula only gives an approximate answer to the question as it has not taken into account the fact that the first and the last letter in the string *gcctagg* happens to be the same. In this particular case the above formula works for all $i < 13$, but at $i \geq 13$ it fails because the 13-string *gcctaggcctagg* contains the missing 7-string twice, a fact not reflected in the inductive derivation of Eq. (1.12).

The situation becomes more formidable when at certain K there are several true missing strings and among the missing ones there exist overlaps among their prefixes and suffixes. A prominent example is provided by the hyperthermophilic bacterium *Aquifex aeolicus* genome [45]. In this 1 551 335 letter sequence four true missing strings are identified at $K = 7$:

$$B = \{gcgcgcg, gcgcgca, cgcgcgc, tgcgcgc\}. \tag{1.13}$$

The overlapping among these forbidden words is apparent. Denote by a_K the number of words of length K within Σ^* that do not contain any element of the subset B and define a generating function

$$f(s) = \sum_{K=0}^{\infty} a_K s^K, \tag{1.14}$$

where s is an auxiliary variable. An explicit expression of $f(s)$ may be obtained [46, 47] by invoking the Goulden-Jackson cluster method [48] in combinatorics. The Goulden-Jackson method is capable to determine the number of strings including or excluding designated substrings with overlapping prefixes and suffixes among the later. It works even for letters with non-equal probabilities of appearance [49]. Not going into the details of the combinatorics, we turn to the language theory solution of the same problem.

1.5.4
Factorizable Language Defined by a Genome

Given a complete genome G one may define a language as follows. Take enough copies of the same genome G and cut them in all possible ways, from single nucleotides, dinucleotides, trinucleotides, up to the uncut sequences themselves in G. The collection of all these strings plus an empty string ϵ defines a language $L(G) \subset \Sigma^*$ over the alphabet $\Sigma = \{a, c, g, t\}$. Clearly, the language $L(G)$ is factorizable by construction.

Now we show how formal language theory may provide a framework to yield concrete solution to an appropriate problem in numbers. First of all, any language $L \subset \Sigma^*$ introduces an *Equivalence Relation* R_L in Σ^* with respect to L: any two elements $x, y \in \Sigma^*$ are equivalent and denoted as xR_Ly if and only if for every $z \in \Sigma^*$ both xz and yz either belong to L or not belong to L. As usual, the index of R_L is the number of equivalent classes in Σ^* with respect to L. An equivalent class may be represented by any element $x \in L$ of that class and we will denote this equivalent class by $[x]$. The importance of the equivalent relation R_L comes from the Myhill-Nerode Theorem in language theory, see, for example, reference [4]: L is regular if and only if the index of R_L is finite and the number of states in the minDFA that accepts L is given by the index.

Tab. 1.6 The transfer function for the minDFA accepting the *A. aeolicus* genome.

Class	a	c	g	t
$[\epsilon]$	$[\epsilon]$	$[c]$	$[g]$	$[c]$
$[g]$	$[\epsilon]$	$[gc]$	$[g]$	$[c]$
$[gc]$	$[\epsilon]$	$[c]$	$[gcg]$	$[c]$
$[gcg]$	$[\epsilon]$	$[gcgc]$	$[g]$	$[c]$
$[gcgc]$	$[\epsilon]$	$[c]$	$[gcgcg]$	$[c]$
$[gcgcg]$	$[\epsilon]$	$[gcgcgc]$	$[g]$	$[c]$
$[gcgcgc]$	$[L']$	$[c]$	$[L']$	$[c]$
$[c]$	$[\epsilon]$	$[c]$	$[cg]$	$[c]$
$[cg]$	$[\epsilon]$	$[cgc]$	$[g]$	$[c]$
$[cgc]$	$[\epsilon]$	$[c]$	$[cgcg]$	$[c]$
$[cgcg]$	$[\epsilon]$	$[cgcgc]$	$[g]$	$[c]$
$[cgcgc]$	$[\epsilon]$	$[c]$	$[cgcgcg]$	$[c]$
$[cgcgcg]$	$[\epsilon]$	$[L']$	$[g]$	$[c]$

Taking the four true missing strings in the *A. aeolicus* genome as forbidden words, that is, let $L'' = B$, we undertake to construct a finite state automaton that accepts $L(G)$. We define a set V:

$$V = \{v | v \text{ is a proper prefix of some } y \in L''\}.$$

Then for each word $x \in L$ there exists a string $v \in V$ such that it is equivalent to x, or using our notations xR_Lv. In other words, all equivalent classes of Σ^* with respect to L are represented in V. Therefore, in order to find all equivalent classes of Σ^* with respect to L it is enough to work with L''. By the way, $[\epsilon]$ and L' are always two equivalent classes among others.

Collecting all proper suffixes of the avoided strings in B, we get

$$\begin{aligned} V \quad = \quad & \{g, gc, gcg, gcgc, gcgcg, gcgcgc, c, cg, cgc, cgcg, \\ & cgcgc, cgcgcg, t, tg, tgc, tgcg, tgcgc, tgcgcg\}. \end{aligned}$$

By checking the equivalence relations, 13 out of 18 elements in V are kept as representatives of the equivalent classes. Adding the class $[L'] \subset \Sigma^*$ we get all 14 equivalent classes of Σ^*:

$$[\epsilon]\ [g]\ [gc]\ [gcg]\ [gcgc]\ [gcgcg]\ [gcgcgc]\ [c]\ [cg]\ [cgc]\ [cgcg]\ [cgcgc]\ [cgcgcg]\ [L'].$$

At first glance the requirement of checking the equivalence relations for every $z \in \Sigma^*$ may seem formidable as it deals with an infinite set. However, a little practice shows that this may be done effectively without too much work.

Treating each class as a state, we define the discrete transfer function by

$$\delta([x_i], s) = [x_i s] \quad \text{for} \;\; x_i \in V \;\; \text{and} \;\; s \in \Sigma.$$

The result is shown in Table 1.6. The special class $[L']$ is a “dead end”, that is, an unacceptable state. The minDFA defined by the transfer function is drawn in Fig. 1.9.

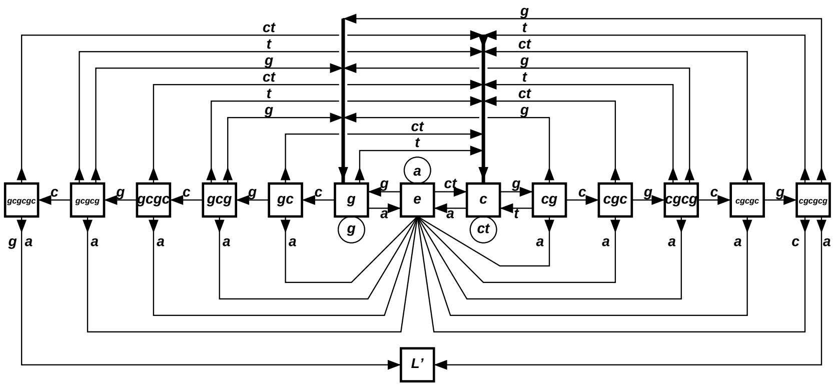

Fig. 1.9 A minDFA accepting the *A. aeolicus* genome with the four forbidden 7-strings given in (1.13).

Counting the number of lines leading from a node (state) to another, we write down the following *incidence* matrix:

$$M = \begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}, \tag{1.15}$$

To make connection with the generating function (1.14) we note that the characteristic polynomial of M is related to $f(\frac{1}{\lambda})$:

$$\det(\lambda I - M) = \lambda^{13} f(\frac{1}{\lambda}).$$

Moreover, the sum of elements in the first row of the K-th power of M is nothing but a_K [9]:

$$a_K = \sum_{j=1}^{13} (M^K)_{1j}.$$

In order not to cause any confusion we note that many true avoided strings longer than 7 may be identified in the *A. aeolicus* genome, we have only used the $K = 7$ ones to construct the minDFA shown in Fig. 1.9. In fact, this DFA accepts a greater language of which $L(G)$ is a subset. In principle, one may invoke more forbidden words to construct a more complex DFA that accepts a smaller language still having $L(G)$ as a subset. Since our goal consists in calculating the number of redundant avoided strings of longer length caused by a given set of true avoided strings, it is enough to restrict the forbidden set to the true avoided strings up to a certain length.

1.6 Decomposition and Reconstruction of Protein Sequences

In recent years we developed a composition vector tree (CVTree) approach [50–53] to infer phylogenetic relationships of bacteria from their complete genomes without making sequence alignment. The justification of CVTree method has led to, among other things, the problem of uniqueness of reconstruction of a protein sequence from its constituent K-peptides. This problem has a natural relation to the number of Eulerian loops in a graph, a well-developed chapter of graph theory (see, for example, [54]). It turns out that in order to tell whether a given sequence has a unique reconstruction at a fixed K the notion of factorizable languages again comes to our help.

1.6.1 A Few Notions of Graph Theory

We need a few notions from graph theory. Look at a connected directed graph with a certain number of labeled nodes. If node i is connected to node j by one directed arc, then we say $a_{ij} = 1$, and so forth. From a beginning node v_b we go through a number of arcs to an ending node v_f in such a way that each arc has been traversed once and only once; such a path is called an *Eulerian path*. If $v_b = v_f$ the path becomes an *Eulerian loop*. A graph in which an Eulerian loop exists is called an *Euler graph*. An Eulerian path may be transformed into an Eulerian loop by drawing an auxiliary arc from v_f back to v_b.

From a given node, there may be d_{out} arcs going out to other nodes; d_{out} is called the outdegree of the node. Likewise, there may be d_{in} arcs coming into a node, d_{in} defines the indegree of the node. The condition for an undirected graph to be Eulerian was indicated by Euler in 1736, the year that has been considered the beginning of graph theory. In our case of directed graph it may be formulated as

$$d_{\text{in}}(i) = d_{\text{out}}(i) = d_i$$

for all nodes numbered in a certain way from $i = 1$ to m. The numbers d_i are simply called degrees. We define a diagonal matrix

$$M = \mathrm{diag}(d_1, d_2, \cdots, d_m).$$

The connectivity of the nodes is described by an adjacent matrix $A = \{a_{ij}\}$, where a_{ij} is the number of arcs leading from node i to j. From the matrices M and A we form the Kirchhoff matrix

$$C = M - A.$$

The Kirchhoff matrix has the peculiar property that its elements along any row or column sum to zero. Furthermore, for any $m \times m$ Kirchhoff matrix all $(m-1) \times (m-1)$ minors are equal and this common minor is denoted by Δ.

A graph is called *simple* if (1) there are no parallel arcs between nodes, that is, $a_{ij} = 0$ or 1 $\forall i, j$; and (2) there are no rings at any node, that is, $a_{ii} = 0$ $\forall i$. The number R of Eulerian loops in a simple Euler graph is given by the **BEST formula** (BEST stands for N. G. de **B**ruijn, T. van Aardenne-**E**hrenfest, C. A. B. **S**mith, and W. T. **T**utte) [54]:

$$R = \Delta \prod_i (d_i - 1)!.$$

This formula gives the number of Eulerian loops in an Euler graph without specifying a starting node. If a node k is specified as the beginning (hence ending) of the loop, then the number of loops starting from k is [55]

$$R = \Delta d_k \prod_i (d_i - 1)!, \tag{1.16}$$

where d_k is the degree of the node k.

In a general Euler graph there may be parallel arcs between certain pairs of nodes ($a_{ij} > 1$) and rings at some nodes ($a_{ii} \neq 0$). One may put auxiliary nodes on these arcs and rings to make the graph simple. By applying elementary operations to the larger Kirchhoff matrix thus obtained, one can reduce it to the original size with some $a_{ii} \neq 0$ and $a_{ij} > 1$. Since the parallel arcs and rings are unlabeled we must eliminate the redundancy in the counting result. Therefore, the BEST formula is modified to

$$R = \frac{\Delta d_k \prod_i (d_i - 1)!}{\prod_{ij} a_{ij}!}. \tag{1.17}$$

As $0! = 1! = 1$ this formula reduces to the previous one in case of simple graphs. The modified BEST formula (1.17) first appeared in [56] where Eulerian loops from a fixed starting node were considered.

1.6.2 The K-peptide Representation of Proteins

In the CVTree method, instead of using a primary protein sequence made of M amino acid letters, we decompose the protein into $M - K + 1$ overlapping K-peptides. Take, for example, the human centromere protein B (*CENB_HUMAN* in the SwissProt database), consisting of 599 amino acids:

```
MGPKRRQLTF REKSRIIQEV EENPDLRKGE IARRFNIPPS TLSTILKNKR AILASERKYG
VASTCRKTNK LSPYDKLEGL LIAWFQQIRA AGLPVKGIIL KEKALRIAEE LGMDDFTASN
GWLDRFRRRH GVVSCSGVAR ARARNAAPRT PAAPASPAAV PSEGSGGSTT GWRAREEQPP
SVAEGYASQD VFSATETSLW YDFLPDQAAG LCGGDGRPRQ ATQRLSVLLC ANADGSEKLP
PLVAGKSAKP RAGQAGLPCD YTANSKGGVT TQALAKYLKA LDTRMAAESR RVLLLAGRLA
AQSLDTSGLR HVQLAFFPPG TVHPLERGVV QQVKGHYRQA MLLKAMAALE GQDPSGLQLG
LTEALHFVAA AWQAVEPSDI AACFREAGFG GGPNATITTS LKSEGEEEEE EEEEEEEEEG
EGEEEEEEGE EEEEEGGEGE ELGEEEEVEE EGDVDSDEEE EEDEESSSEG LEAEDWAQGV
VEAGGSFGAY GAQEEAQCPT LHFLEGGEDS DSDSEEEDDE EEDDEDEDDD DDEEDGDEVP
VPSFGEAMAY FAMVKRYLTS FPIDDRVQSH ILHLEHDLVH VTRKNHARQA GVRGLGHQS
```

Decomposing the above sequence into a collection of overlapping 5-peptides `MGPKR`, `GPKRR`, and so forth., and considering each 5-peptide as a transition from its 4-letter prefix to the 4-letter suffix, one easily transforms the sequence to an Eulerian path, using the 4-strings as node labels. Since we are interested in the number of Eulerian loops, we can replace a node with degree 1 by an arc without affecting the number of loops. Thus, only nodes with $d_i > 1$ matter. Finally, by drawing an auxiliary arc from the last node to the first we get an Eulerian loop and this loop defines an Euler graph, see Fig. 1.10. Since we are interested only in the number of different Eulerian loops in the graph, we can add a degree 1 node v_0 on the auxiliary arc and count the number of loops starting from this node. Therefore, in practical calculations we always take $d_k = 1$ in Eqs. (1.16) and (1.17).

From Fig. 1.10 we infer the diagonal matrix

$$M = \mathrm{diag}(1, 2, 2, 2, 2, 2, 4, 4, 20, 4, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2)$$

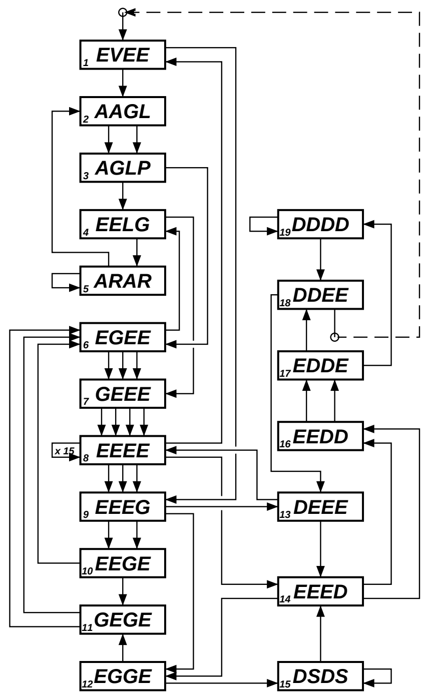

Fig. 1.10 An Euler graph generated by the protein sequence CENB_HUMAN with 599 amino acids.

and the adjacent matrix

$$A = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}.$$

(1.18)

The common minor of the Kirchhoff matrix $C = M - A$ is $\Delta = 168\,960$. Denoting by $R(K)$ the number of reconstructions at K, we get from the modified BEST formula $R(5) = 491\,166\,720$. We see that the protein $CENB\_HUMAN$ belongs to the few sequences that have a huge number of reconstructions at moderate value of K. Most of the naturally occurring proteins have unique reconstruction at $K = 5 \sim 6$ [57], a fact speaking in favor of the composition vector approach.

In principle, one can write a program to generate all distinct reconstructions from a given set of K-tuples obtained by decomposing an original protein sequence. In so doing a cut-off reconstruction number must be set because there might be proteins with a huge number of reconstructions as shown by the above example. Alternatively, by implementing the modified BEST formula (1.17) one may get the number of reconstructions without producing all the sequences. It is curious to note that equipped with these programs one can fish out a set of proteins that have a huge number of reconstructions without any biological knowledge as prerequisite [58].

However, if we are only interested in whether a given symbolic sequence over a finite alphabet Σ has a unique reconstruction at a given K or not, factorizable language may help us to construct finite state automata to yield an YES/NO answer.

1.6.3
Uniquely Reconstructible Sequences and Factorizable Language

We put the sequence decomposition and reconstruction problem in a more general setting and ask a more specific question. Consider a finite alphabet Σ and an arbitrary sequence $s \in \Sigma^*$ of length N. Decompose the sequence s into $N - K + 1$ overlapping K-tuples and then reconstruct sequences by using each of the K-tuples once and only once. Collect all the sequences that have an unique reconstruction into a subset $L \subset \Sigma^*$. The subset L defines the language of uniquely reconstructible sequences. The language L is a factorizable language by definition, because any substring in a word $s \in L$ must be uniquely reconstructible otherwise the whole word s cannot be in L. To the contrary, the language $L' = \Sigma^* - L$ consists of sequences that must have two or more reconstructions. Among words in L' there is a set L'' of forbidden words.

Question: given a sequence $s \in \Sigma^*$, judge whether s is in L or not; if not, s must be in L'. A deterministic finite state automaton may be built to answer this YES/NO question.

It was first conjectured in [59] and then proved in [60] that the non-uniqueness in sequences reconstruction comes from two kinds of transformations. Applied to our problem of loops with starting and ending nodes connected by an auxiliary arc, we can take all the $(K-1)$-tuples that label the nodes as symbols in a new alphabet. Then it is sufficient to consider only the case of $K = 2$. The two types of transformations are: *Transposition*

$$\begin{aligned} \cdots xwz \cdots xuz \cdots \quad &\Longleftrightarrow \quad \cdots xuz \cdots xwz \cdots , \\ \cdots xwxux \cdots \quad &\Longleftrightarrow \quad \cdots xuxwx \cdots , \end{aligned}$$

and *Rotation*

$$xwzux \quad \Longleftrightarrow \quad zuxwz,$$

where $x, z \in \Sigma$ and $w, u \in \Sigma^*$. These conditions are necessary but not sufficient for non-uniqueness of reconstruction. Since the starting and ending symbols are fixed in our construction the *rotation* drops out of consideration.

A recent paper [61] discussed the reconstruction problem in an entirely different context and obtained the necessary and sufficient conditions for an Eulerian loop to be unique in an Euler graph. In particular, Kontorovich [61] proved

Theorem 1.6 *The set of all uniquely reconstructible sequences is a regular language.*

Unfortunately, the abstract proof of the theorem did not provide a way to build a finite state automaton that would accept an uniquely reconstructible language.

By invoking the notion of factorizable language and using the set of forbidden words we construct such an automaton explicitly and thus provide a constructive proof of the theorem.

We begin with

Theorem 1.7 *Under the condition that $K = 2$ and the starting and ending symbols are identical, an uniquely reconstructible language $L \subset \Sigma^*$ only possesses two types of forbidden words:*

$$\begin{aligned} (i) \quad & x\alpha y\gamma x\beta y, \\ (ii) \quad & x\alpha x\beta x, \end{aligned} \tag{1.19}$$

where $x, y \in \Sigma$ and $x \neq y$, $\alpha, \beta, \gamma \in L$, and they satisfy the following conditions

1. *In (i): at least one of α and β is not empty, all α, β and γ do not contain neither the symbols x, y nor identical symbols;*
2. *In (ii): at least one of α and β is not empty, they do not contain neither the symbol x nor identical symbols.*

We skip the simple but somewhat lengthy proof and go directly to the construction of the finite state automaton.

1.6.4
Automaton that Accepts an Uniquely Reconstructible Language

We first describe a deterministic finite state automaton M that accepts an uniquely reconstructible language L [62]. The automaton M consists of five elements:

$$M = \{Q, \Sigma, \delta, q_0, F\}, \tag{1.20}$$

where Σ is the finite alphabet, $Q = \{q\}$ is the set of states, $q_0 \in Q$ is the initial state, $F \in Q$ are the states that accept L, and δ is the transfer function from $Q \times \Sigma$ to Q. The automaton M reads in symbols in a sequence $s \in \Sigma^*$ from the left to right, one symbol a at a time, and changes its state from q to a new state q' according to the transfer function $\delta(q, a) = q'$. We explain these elements one by one.

1. Σ is an alphabet of m symbols $\{a_1, a_2, \cdots, a_m\}$ which may be denoted by $\{1, 2, \cdots, m\}$ as well. The language $L \subset \Sigma^*$ is defined over Σ and the automaton M reads a sequence $s \in \Sigma^*$.
2. Each state in Q consists of three components $(p; n; c)$: $Q = P \times N \times C = \{(p; n; c)\}$, where
   - p records the most recently read symbol $a_p \in \Sigma$. For the initial state q_0 when no input symbol has been read yet we introduce a symbol a_0 (or 0) which does not belong to Σ. Thus we may write $P = \Sigma \bigcup 0$.
   - n is a list of $m+1$ symbols $(n_0, n_1, n_2, \cdots, n_m)$ that updates the next symbol read after p. For the initial state $n = (\epsilon, \epsilon, \epsilon, \cdots, \epsilon) \equiv \epsilon^{m+1}$, where ϵ means empty or non-existence. Thus $N = (\Sigma \bigcup \epsilon)^{m+1}$.
   - c is a list of m toggle switches: $c = (c_1, \cdots, c_m)$. We denote the two states of a toggle as $WHITE$ and $BLACK$. Initially $c = WHITE^m$. Whenever a forbidden word in the sequence s has been read c becomes $BLACK^m$. As long as c is not all-black the state q is acceptable. Once c becomes all-black it remains so for ever and the automaton recognizes s as a non-uniquely reconstructible sequence.
3. The initial state $q_0 = (0; \epsilon^{m+1}; WHITE^m)$.
4. The acceptable states

   $$F = (p; n; c \neq BLACK^m). \tag{1.21}$$

5. The key element of M is the transfer function $\delta(q, a)$. A program to implement δ is written down below using a simple meta-language (i is a working variable in the program):

```
1  procedure $\delta((p, n, c), a)$
2    if $(n_p \neq \epsilon)$ & $(n_p \neq a)$ then
3      $i \leftarrow p$
4      repeat
5        $c_i \leftarrow BLACK$
6        $i \leftarrow n_i$
7      until $i = p$
8    endif
9    if $c_a = BLACK$ then
10     $c \leftarrow BLACK^m$
11   endif
12   $p \leftarrow n_p \leftarrow a$
13 end procedure
```

All technical terms involved in describing the automaton may be found in [4]. A C++ implementation of the transfer function δ is available upon request to the authors of [62]. As usual, the best way to understand the workings of this program is to take a couple of uniquely and non-uniquely reconstructible sequences over some alphabet and follow the transitions among states according to the program.

The proof of the following theorem may help to grasp the essence of the transfer function δ.

Theorem 1.8 *The language $L(M)$ accepted by the finite state automaton M defined in Eq.* (1.20) *is the uniquely reconstructible language $L \subset \Sigma^*$ over an alphabet Σ of m symbols.*

Proof. The proof of $L(M) = L$ goes in two parts.

First part. We prove $L(M) \subset L$ by showing that all sequences in the complementary language L' cannot be accepted by the automaton M.

Suppose $t \notin L$ then t contains forbidden words. Consider the first forbidden word encountered when feeding t to M. If this forbidden word belongs to type (i) in Theorem 1.7, that is, it is of type $x\alpha y\gamma x\beta y$, where $x, y \in \Sigma$, $x \neq y$, $\alpha, \beta, \gamma \in L$, α and β are not empty at the same time, α, β and γ do not contain neither x, y nor identical symbols. Now look at Lines 2 $\sim$ 8 in procedure δ. Suppose that M has reached a state where p equals the second x, then its next symbol is the a in $\delta(q, a)$. (If β is not empty then a is the first symbol in β, otherwise a is the last y.) Since the symbol after the first x cannot be a, the condition in Line 2 of the procedure holds. Therefore, Line 5 makes $c_x = BLACK$. Now the loop from Line 4 to Line 7 makes every toggle c_i corresponding to symbols in $x\alpha y\gamma$ becomes $BLACK$, including $c_y = BLACK$. As blackened toggles cannot become $WHITE$ again, when reading in the last y the execution of Line 9 and 10 turns c to all-$BLACK$, that is, M enters a non-acceptable state.

When the first forbidden word in t belong to type (ii) in Theorem 1.7, that is, it is of type $x\alpha x\beta x$, the situation is simpler and we ignore the discussion.

Second part. In order to prove $L \subset L(M)$ it is enough to show that after reading in the sequence $t \in L$ the state $(p; n; c)$ belongs to the acceptable states F given in Eq. (1.21), namely, the toggles c are not all-$BLACK$.

We prove this by mathematical induction with respect to the length $|t| = N$ of the sequence $t \in L$. From the definition of the automaton M we know that when $N = 0$ the initial state $q_0 \in F$, the statement holds true. Now suppose the statement holds for $N - 1$ and discuss the situation of $|t| = N$.

Since L is a factorizable language, we denote by a the last symbol of $t \in L$ and write t as $t = sa$, the prefix s of t is uniquely reconstructible and it is of length $N - 1$. Therefore, c is not all-$BLACK$ after reading in s.

Now we prove that upon reading a the toggles c do not become all-$BLACK$ by reduction to absurdity. Suppose the opposite is true, that is, upon reading in a the toggles c become all $BLACK$. An inspection of the procedure δ shows that this may happen in two cases:

(1). The toggle c_a has become $BLACK$ before reading in a. Therefore, after reading a the Line 10 in procedure δ is executed that makes c all-$BLACK$. However, in order to enable Line 4 to be executed the two conditions in Line 1 must be satisfied. This means the sequence s must have a suffix like $b\alpha b\beta$ where the symbols following the two symbols b must be different and at the same time $a \in b\alpha$. Then $t = sa$ must contain a forbidden word as suffix no matter whether a equals b or not. Therefore, $t \notin L$, a contradiction.

(2). The toggle c_a was $WHITE$ and it turns $BLACK$ only after reading a. Now we must consider the last symbol in s. Denote it by p. The fact that changing c_a to $BLACK$ takes place in Line 2 to Line 7 shows that t must have a suffix $p\alpha pa$ with non-empty α; the first symbol of α is not a but a appears in $p\alpha$. Therefore, $p\alpha pa$ is a forbidden word independent on whether p equals a or not, a contradiction to $t \in L$.

Thus we have proved $L \subset L(M)$. Combination of the two parts completes the proof. □

Clearly, M is a deterministic finite state automaton. Although how to build the corresponding minDFA from the above DFA is known in principle [4], the explicit construction remains lacking for the time being. In particular, we do not know the size of the minDFA which is given by the index of the equivalence relation R_L generated by L in Σ^*.

In principle, one can build an automaton that accepts the complementary L' of the language L. The right-linear grammar to implement such an NDFA may be found in [62].

1.6.5 Other Applications of the Sequence Reconstruction Problem

The sequence unique reconstruction problem occurred in so-called *sequencing by hybridization* (SBH), one of the earliest proposed application of DNA arrays. It has been analyzed from the viewpoint of the number of Eulerian loops, see [60] and references therein. However, no connection with language theory was mentioned in these studies.

Another possible application is associated with sequence randomization under constraints. In order to tell the statistical significance of certain "signals" revealed in some symbolic sequences one must compare them with what might be observed in a background model [44]. A frequent choice of the background model is full randomization of the original sequence that only keeps the number of single letters unchanged. However, under certain circumstances it is more appropriate to perform the randomization under some further constraints, for example, keeping a designated number of short strings fixed and having the rest of the sequence "randomized". In this setting we encounter the opposite of the unique reconstruction problem, namely, only when there exists a huge number of reconstructions under the given constraints it makes sense to speak about "randomization", otherwise it is just a choice among a finite number of shufflings. This problem has been studied before [63], again no reference to factorizable language was made.

We expect more applications of factorizable languages, especially the method of forbidden words, in various problems of dynamics and biology.

References

1 Hao, B.-L., in *On Self-Organization — An Interdisciplinary Search for a Unifying Principle,* Springer Series in Synergetics 61, Springer Verlag, **1994**, p. 197

2 Hao, B.-L., Zheng, W.-M., *Applied Symbolic Dynamics*, World Scientific, Singapore, **1998**

3 Shannon, C. E., A mathematical theory of communication, *Bell Systems Tech. J.*, 27 (**1948**), p. 379 and p. 623

4 Hopcroft, J and Ullman, J., *Introduction to Automata Theory, Languages and Computation,* Addison-Wesley, Reading, **1979**

5 Xie, H.-M., *Grammatical Complexity and One-Dimensional Dynamical Systems,* World Scientific, Singapore, **1996**

6 Rozenberg, G., Salomma, A., eds., *Handbook of Formal Languages*, vols. 1-3, Springer, **1997**

7 Shyr, H. J., *Free Monoids and Languages*, Hon Min Book Company, Taichung, **1991**

8 Rozenberg, G., Salomaa, A., *The Mathematical Theory of L-Systems*, Academic Press, **1980**

9 Wolfram, S., Computation theory of cellular automata. *Commun. Math. Phys.* 96 (**1984**), p. 15

10 Morse, M., Hedlund, G. A., Symbolic dynamics, *Am. J. Math.* 60 (**1938**), p. 815; reprinted in *Collected Papers of M. Morse*, vol. 2, World Scientific, **1986**

11 Hao, B.-L., Xie, H.-M., Yu, Z.-G., Chen, G.-Y., Factorisable language: From dynamics to complete genomes, *Physica* A288 (**2000**) p. 10

12 Xie, H.-M., On formal languages of one-dimensional dynamical systems, *Nonlinearity*, 6, (**1993**), p. 997

13 Hao, B.-L., Symboic dynamics and characterization of complexity, *Physica* D51 (**1991**), p. 161

14 Crutchfield, J. P., Young, K., Computation at the onset of chaos, in *Complexity, Entropy, and Physics of Information*, ed. W. Zurek, Addison-Wesley, **1990**, p. 223

15 Wang, Y., Yang, L., Xie, H.-M., Complexity of unimodal maps with aperiodic kneading sequences, *Nonliearity*, 12 (**1999**), p. 1151

16 von Neumann, J., *Theory of Self-Reproducing Automata*, edited by Burks, A. W., Univ of Illinois Press, Champaign, **1966**

17 Berlekamp, E., Conway, J. H. and Guy, R., *Winning Ways,* vol 2, Academic Press, New York, **1982**

18 Wolfram, S., *Theory and Applications of Cellular Automata,* Singapore: World Scientific, **1986**

19 Wolfram, S., *Cellular Automata and Complexity,* Addison-Wesley, New York, **1994**

20 Wolfram, S., *A New Kind of Science,* Wolfram Media Inc, Champaign, **2002**

21 Cook, M., Universality of elementary cellular automata. *Complex Systems* 15 (**2004**), p. 1

22 Hurd, L. P., Recursive cellular automata invariant sets. *Complex Systems* 4 (**1990**), p. 119

23 Hurd, L. P., Nonrecursive cellular automata invariant sets. *Complex Systems* 4 (**1990**), p. 131

24 Culik, II K., Hurd, L. P. and Yu, S., Computation theoretic aspects of cellular automata. *Physica* D45 (**1990**), p. 357

25 Kari, J., The nilpotency problem of one-dimensional cellular automata. *SIAM J Comp* 21 (**1992**), p. 571

26 Xie, H. M., Distinct excluded blocks and grammatical complexity of dynamical systems, *Complex Systems* 9 (**1995**), p. 73

27 Xie, H.-M., The complexity of limit languages of cellular automaton: an example. *J. of Systems Science & Complexity* 14 (**2001**), p. 17

28 Jiang, Z.-S. and Wang, Y., Complexity of limit language of the elementary cellular automaton of rule 22. *Appl. Math. J. Chinese Univ, ser.B* 20 (**2005**), p. 268

29 Jiang, Z.-S., A complexity analysis of the elementary cellular automaton of rule 122. *Chinese Science Bulletin* 46(7) (**2001**), p. 600

30 Cattaneo, G., Dennunzio, A. and Margara, L., Chaotic subshifts and related languages—applications to one-dimensional cellular automata. *Fund Inform* 52 (**2002**), p. 39

31 Hurley, M., Attractors in restricted cellular automata. *Proc Amer Math Soc* 115 (**1990**), p. 563

32 Blanchard, F., Kůrka, P. and Maass, A., Topological and measure-theoretic properties of one-dimensional cellular automata. *Physica* D103 (**1997**), p. 86

33 Kůrka, P., Languages, equicontinuity and attractors in cellular automata. *Ergodic Theory & Dynamic Systems* 17 (**1997**), p. 417

34 Jen, E., Aperiodicity in one-dimensional cellular automata. *Physica* D45 (**1990**), p. 9

35 Jen, E., Exact solvability and quasiperiodicity of one-dimensional cellular automata. *Nonlinearity* 4 (**1990**), p. 251

36 Jiang, Z.-S. and Xie, H.-M., Evolution complexity of the elementary cellular automaton rule 18. *Complex Systems* 13 (**2001**), p. 271

37 Wang, Y. and Kenichi, M., Complexity of evolution languages of the elementary cellular automaton of rule 146. *Appl Math J Chinese Univ, Ser B* 21 (**2006**), p. 418

38 Wang, Y. and Jiang, Z.-S., Evolution complexity of the elementary cellular automaton of rule 22. *Appl Math J Chinese Univ, Ser B* 17 (**2002**), p. 404

39 Qin, D.-K. and Xie, H.-M., Complexity analysis of time series generated by elementary cellular automata. *App Math J Chinese Univ, Series B* 20 (**2005**), p. 253

40 Qin, D.-K. and Xie, H.-M., Catalan numbers, Dyck language and time series of elementary cellular automaton of rule 56. *J. of Systems Science & Complexity* 18 (**2005**), p. 404

41 Sloane, N. J., Sequence A000108 in *The On-Line Version of the Encyclopedia of Integer Sequences*: http://www.research.att.com/∼njas/sequences/eisonline.html

42 Shen, J.-J., Zhang, S.-Y., Lee, H.-C., Hao, B.-L., SeeDNA: Visualization of K-string content of long DNA sequences and their randomized counterparts, *Genomics, Proteomics & Bioinformatics* 2 (**2004**), p. 192

43 Hao, B.-L., Lee, H.-C., Zhang, S.-U., Fractals related to long DNA sequences and bacterial complete genomes, *Chaos, Solitons & Fractals*, 11 (**2000**) p. 825

44 Xie, H.-M., Hao, B.-L. Visualization of *k*-tuples distribution in prokaryote complete genomes and their randomized counterparts, in *Bioinformatics. CBS2002 Proceedings,* , IEEE Computer Society, Los Alamitos, California **2002**, p. 31

45 Deckert,G. *et al.*, The complete genome of the hyperthermophilic bacterium *Aquifex aeolicus*, *Nature* 392 (**1998**), p. 353

46 Hao, B.-L., Xie, H.-M., Yu, Z.-G., Chen, G.-Y., A combinatorical problem related to avoided strings in bacterial complete genomes, *Ann. Combin.* 4 (**2000**), p. 247

47 Hao, B.-L., Fractals from genomes: exact solutions of a biology-inspired problem, *Physica* A282 (**2000**) p. 225

48 Goulden, I., Jackson, D. M., *Combinatorial Enumeration*, John Wiley & Sons, New York, **1983**

49 Zhou, C., Xie, H.-M., Exact distribution of the occurrence number for K-tuples over an alphabet of non-equal probability letters, *Ann. Combinatorics* 8 (**2004**) p. 499

50 Qi, J., Wang, B., Hao, B.-L., Whole genome prokaryote phylogeny without sequence alignment: a K-string composition approach, *J. Mol. Evol.,* 58 (**2004**), p. 1

51 Qi, J., Luo, H., Hao, B.-L., CVTree: a phylogenetic tree reconstruction tool based on whole genomes, *Nucl. Acids Res.*, 32 (**2004**), Web Server Issue, p. W45

52 Hao, B.-L., Qi, J., Prokaryote phylogeny without sequence alignment: from avoidance signature to composition distance, *J. Bioinformatics & Computational Biology*, 2 (**2004**), p. 1

53 Gao, L., Qi, J., Sun, J.-D., Hao, B.-L., Prokaryote phylogeny meets taxonomy: an exhaustive comparison of the composition vector tree with the biologist's systematic bacteriology, *Science in China* Series C Life Sciences, to appear, (**2007**)

54 Fleischner, H., *Eulerian Graphs and Related Topics*, Part 1, vol. 2, **1991**, p. IX80

55 Bollobás, B., *Modern Graph Theory*, Springer-Verlag, New York, **1998**

56 Hutchinson, J. P., On words with prescribed overlapping subsequences, *Utilitas Mathematica*, 7, (**1975**), p. 241

57 Xia, L., Zhou, C., Phase transitions in sequence unique reconstruction, *J. Syst. Sci. & Complexity* 20 (**2007**), p. 18

58 Shi, X.-L., Xie, H.-M., Zhang, S.-Y., Hao, B.-L., Decomposition and reconstruction of protein sequences: the problem of uniqueness and factorizable language, *J. Korean Phys. Soc.* 58 (**2007**), p. 118

59 Ukkonen, E., Approximate string matching with q-grams and maximal matches, *Theor. Comput. Sci.*, 92, (**1992**), p. 191

60 Pevzner, P. A., *Computational Molecular Biology: An Algorithmic Approach*, The MIT Press, Cambridge, MA, **2000**

61 Kontorovich, L., Uniquely decodable n-gram embeddings, *Theor. Computer Sci.*, 329 (**2004**), p. 271

62 Li, Q., Xie, H.-M., Finite automata for testing uniqueness of Eulerian trails, **2005**, arXive: cs.CC/0507052 (20 July 2005); Finite automata for testing composition-based reconstructibility of sequences, submitted to *J. Computer & Systems Sci.*, in April **2007**

63 Kandel, D., Matias, Y., Unger, R., Winkler, P., Shuffling biological sequences, *Discrete Appl. Math.*, 71 (**1996**), p. 171