INSTITUTE OF PHYSICS PUBLISHING NONLINEARITY

Nonlinearity **17** (2004) 1317–1336 PII: S0951-7715(04)70723-9

Dynamics and computation in functional shifts

Jun Namikawa and Takashi Hashimoto

School of Knowledge Science, Japan Advanced Institute of Science and Technology, Tatsunokuchi, Ishikawa 923-1292, Japan

E-mail: jnamika@jaist.ac.jp

Received 22 October 2003, in final form 19 March 2004
Published 13 April 2004
Online at stacks.iop.org/Non/17/1317
DOI: 10.1088/0951-7715/17/4/009

Recommended by J P Keating

Abstract

We introduce a new type of shift dynamics as an extended model of symbolic dynamics, and investigate the characteristics of shift spaces from the viewpoints of both dynamics and computation. This shift dynamics is called a functional shift, which is defined by a set of bi-infinite sequences of some functions on a set of symbols. To analyse the complexity of functional shifts, we measure them in terms of topological entropy, and locate their languages in the Chomsky hierarchy. Through this study, we argue that considering functional shifts from the viewpoints of both dynamics and computation gives us opposite results about the complexity of systems. We also describe a new class of shift spaces whose languages are not recursively enumerable.

Mathematics Subject Classification: 03D05, 03D35, 37B10, 37B40, 68Q15, 68Q45, 70K99

1. Introduction

We propose a new framework of shift dynamics, called functional shifts, which extends symbolic dynamics. A functional shift is defined as a shift space that is a set of bi-infinite sequences of some functions on a set of symbols, while symbolic dynamics is usually defined as a set of bi-infinite sequences of finite symbols. This functional shift generates another shift space, called a generated shift, as follows. Consider a sequence of functions $(f_i)_{i\in\mathbb{Z}}$ contained in the functional shift. The sequence of functions generates a sequence of symbols $(x_i)_{i\in\mathbb{Z}}$ determined by $x_{i+1} = f_i(x_i)$. A set of such bi-infinite sequences of symbols is also a shift space. Thus, this framework gives us a way of analysing the relationship between classes of shift spaces using the generative operation.

Introducing the framework of functional shifts allows us to consider the dynamic change of functions. In traditional dynamical systems theory, the time dependence of a function

governing change of states is not serious. The reason for this is that time in a dynamical system can be treated as an additional phase space variable, so that for any dynamical system we can describe it with a time-independent function. However, the time dependence of functions, i.e. the dynamic change of functions, has recently become a subject of interest in the investigation of dynamical systems, because there are many phenomena in which functions should be regarded as variable. For example, in a population dynamics with species extinction and speciation, the degree of freedom dynamically changes [1, 2]. In studying such a phenomenon, since it is difficult to get an immutable evolution rule, we would like to treat functions of the system as dynamic. Therefore, considering dynamic changes of a function is an important perspective for understanding complex systems. We often call such dynamic changes meta-dynamics.

Several models have been proposed to study the dynamic change of functions [3–9]. Sato and Ikegami [3] introduced switching map systems, in which maps to govern the evolution of the systems are dynamically switched with other maps in the system. Kataoka and Kaneko [4–6] investigated the evolution of a one-dimensional function f_n defined by $f_{n+1} = (1-\epsilon) f_n + \epsilon f_n \circ f_n$. Studying dynamics in which functions vary in time using meta-dynamics can be important when considering system evolution or learning. Fontana [7] studied abstract chemical systems that are defined by a loop in which objects encode the functions that act on them. For another instance, Tsuda [9] proposed a switching map system as a model of the brain. He has shown that a skew-product transformation can be considered as a framework describing meta-dynamics. Functional shifts also can be represented by skew-product transformations.

The framework of functional shift has two major advantages. One is the ability to directly compare dynamics with meta-dynamics, since both are represented by shift spaces. The other is to be able to analyse both dynamical and computational characteristics, because this framework is an extension of symbolic dynamics. Moreover, as a kind of meta-dynamics, we can discuss self-modifying systems in terms of functional shifts, in which the functions governing the dynamics of the system are used to change the functions themselves.

In this paper, we study functional shifts from the viewpoints of both dynamics and computation. In recent years, several studies have focused on the relationship between dynamics and computation [10–17]. The central idea in these studies is to regard the time evolution of dynamics as a computational process. Based on a correspondence between the unpredictability of dynamical systems and the halting problem, Moore [10] insists on the existence of dynamics that are more complex than chaos. In addition, computational complexity for continuous time analogue computation has been studied [12–15], in which the convergence of ordinary differential equations is interpreted as a process of computation. The relationship between the complexity of dynamics and computation is, however, still unclear. In this work, we discuss the complexity of the dynamics of functional shifts in terms of topological entropy, which measures the diversity of the orbits of dynamical systems, and also investigate the complexity of their computation in terms of the Chomsky hierarchy. Through this study, we argue that dynamics and computation give us opposite results concerning the complexity of systems.

In an analysis of the complexity of computation in functional shifts, we prove that there exists a shift space whose language is not recursively enumerable (r.e.), even though the language of the functional shift that generates it is r.e. Computational classes of sets to be beyond r.e. are related to analogue computation in the interest of both dynamics and computation. While Siegelmann [11] introduced analogue shifts as a model of analogue computation which is more powerful than the universal Turing machine, the dynamical features of such a powerful computation system are an open problem. When we study dynamical systems modified by meta-dynamics, we discuss analogue computation with functional shifts, and argue that the

existence of dynamical behaviour that is more complex than the universal Turing machine should be taken into account.

This paper is organized as follows. We first review some basic definitions of shift spaces, and define functional shifts in section 2. Next, in section 3, we investigate the property of entropy in functional shifts. We show that the entropy of a functional shift gives the upper limit for that of a generated shift given by it, in order to study the relationship between dynamics and meta-dynamics (see theorem 3.1). In section 4, we compare functional shifts with generated shifts, by focusing on how the language of a shift space belongs to the Chomsky hierarchy of formal languages. We prove that any class of the languages of functional shifts is contained in that of the generated shifts given by them (see corollary 4.2 and theorems 4.4, 4.7, and 4.12). One of the most important results in this section is that there is a shift space whose language is not r.e., even though the language of a functional shift to generate the shift space is r.e. (see theorem 4.12). Finally, our results are discussed from the standpoints of dynamics, computation, meta-dynamics, and self-modifying systems.

2. Definition

Since we will study shift dynamics, we first give some definitions for shift spaces [18]. Let $\mathcal{A}$ be a non-empty finite set of symbols called an alphabet. The full $\mathcal{A}$-shift (simply the full shift) is the collection of all bi-infinite sequences of symbols from $\mathcal{A}$. Here, such a sequence is denoted by $x = (x_i)_{i\in\mathbb{Z}}$ and the full $\mathcal{A}$-shift is denoted by

$$\mathcal{A}^{\mathbb{Z}} = \{x = (x_i)_{i\in\mathbb{Z}} \mid \forall i \in \mathbb{Z} x_i \in \mathcal{A}\}. \tag{1}$$

A block over $\mathcal{A}$ is a finite sequence of symbols from $\mathcal{A}$. An n-block is simply a block of length n. We write blocks without separating their symbols by commas or other punctuation, so that a typical block over $\mathcal{A} = \{a, b\}$ looks like $aababbabbb$. A sequence of no symbols is called an empty block and denoted by ϵ. For any alphabet $\mathcal{A}$, we write $\mathcal{A}^*$ to denote the set of all blocks over $\mathcal{A}$. If $x \in \mathcal{A}^{\mathbb{Z}}$ and $i \leqslant j$, then we denote a block of coordinates in x from position i to position j by $x_{[i,j]} = x_i x_{i+1} \cdots x_j$.

Let σ be a shift map on a full shift $\mathcal{A}^{\mathbb{Z}}$ defined by $\sigma((a_i)_{i\in\mathbb{Z}}) = (a_{i+1})_{i\in\mathbb{Z}}$ for any $a \in \mathcal{A}^{\mathbb{Z}}$. A subset X of $\mathcal{A}^{\mathbb{Z}}$ is called shift-invariant iff $\sigma(X) = X$. Let $\mathtt{F}$, which we call the forbidden blocks, be a collection of blocks over $\mathcal{A}$. For any such $\mathtt{F}$, define $X_{\mathtt{F}}$ to be the subset of sequences in $\mathcal{A}^{\mathbb{Z}}$ in which no block in $\mathtt{F}$ occurs. A shift space is a subset X of $\mathcal{A}^{\mathbb{Z}}$ such that $X = X_{\mathtt{F}}$ for some collection $\mathtt{F}$. Note that if X is a shift space then X is shift-invariant, but a shift-invariant set is not necessarily a shift space (some examples are given in [18]). The set of all n-blocks that occur in points in X is denoted by $\mathcal{B}_n(X)$, and the language of X is the collection $\mathcal{B}(X) = \bigcup_{n=0}^{\infty} \mathcal{B}_n(X)$. The language of a shift space determines the shift space. Thus, two shift spaces are equal iff they have the same language.

Suppose that X is a shift space and $\mathcal{A}$ is an alphabet. An $(m + n + 1)$-block map $\Phi : \mathcal{B}_{m+n+1}(X) \to \mathcal{A}$ maps from allowed $(m + n + 1)$-blocks in X to symbols in $\mathcal{A}$. A map $\phi : X \to \mathcal{A}^{\mathbb{Z}}$ defined by $y = \phi(x)$ with $y_i = \Phi(x_{i-m} x_{i-m+1} \cdots x_{i+n})$ is called a sliding block code induced by Φ. If Y is a shift space over $\mathcal{A}$ and $\phi(X) \subset Y$, then we write $\phi : X \to Y$. If a sliding block code $\phi : X \to Y$ is onto, ϕ is called a factor code. A shift space Y is a factor of X if there is a factor code from X onto Y. A sliding block code $\phi : X \to Y$ is a conjugacy from X to Y if it is bijective. Two shift spaces X and Y are conjugate (written $X \cong Y$) if there is a conjugacy between X and Y.

Next, we define functional shifts and generated shifts, and explain the basic property of each.

Definition 2.1. *Let $\mathcal{A}$ be a non-empty finite set, and F be a set of maps on $\mathcal{A}$. A functional shift $\mathcal{F}$ is a shift space which is a subset of the full shift $F^{\mathbb{Z}}$.*

A generated shift $X_{\mathcal{F}}$ given by $\mathcal{F}$ is defined by

$$X_{\mathcal{F}} = \{(x_i)_{i\in\mathbb{Z}} \in \mathcal{A}^{\mathbb{Z}} \mid \exists (f_i)_{i\in\mathbb{Z}} \in \mathcal{F}\ \forall i \in \mathbb{Z}\ x_{i+1} = f_i(x_i)\}. \tag{2}$$

Although a generated shift is not required by our definition to be a shift space, it is always a shift space.

Theorem 2.2. *If $\mathcal{F}$ is a functional shift, $X_{\mathcal{F}}$ is a shift space.*

Proof. Let

$$Y_n = \{x \in \mathcal{A}^n \mid \forall f \in \mathcal{B}_n(\mathcal{F})\ \exists i\ x_{i+1} \neq f_i(x_i)\} \tag{3}$$

and $\mathtt{F} = \bigcup_{n\in\mathbb{N}} Y_n$. Suppose that X is a shift space which can be described by the collection $\mathtt{F}$ of forbidden blocks. If $x \in X_{\mathcal{F}}$, then $x \in X$ because every block in $\mathtt{F}$ does not occur in x. Thus, $X_{\mathcal{F}} \subset X$.

Conversely, if $x \in X$, then $x_{[-n,n]} \notin \mathtt{F}$ for all n because $\mathtt{F}$ is a set of blocks never occurring in points in X. Therefore,

$$\forall n \in \mathbb{N}\ \exists f \in \mathcal{B}_{2n}(\mathcal{F})\ \forall i\ x_{[-n,n]i+1} = f_i(x_{[-n,n]i}), \tag{4}$$

then

$$\exists f \in \mathcal{F}\ \forall i \in \mathbb{Z}\ x_{i+1} = f_i(x_i), \tag{5}$$

so that $x \in X_{\mathcal{F}}$. Accordingly, $X \subset X_{\mathcal{F}}$. Hence, $X = X_{\mathcal{F}}$ and $X_{\mathcal{F}}$ is a shift space. ∎

By theorem 2.2, a functional shift is regarded as a rule to generate a shift space. Since any functional shift is also a shift space, we can compare functional shifts with generated shifts by using the properties of shift spaces.

The following examples are instances of functional shifts which generate shift spaces.

Example 2.3 (full shifts). Let $\mathcal{A} = \{0, 1\}$ and $F = \{f, g\}$ be a set of maps such that

x	$f(x)$	$g(x)$
0	1	0
1	0	1

If a functional shift $\mathcal{F}$ is equal to $F^{\mathbb{Z}}$, then $X_{\mathcal{F}}$ is the full shift $\mathcal{A}^{\mathbb{Z}}$.

Example 2.4 (golden mean shift). Let X be the set of all binary sequences with no two 1s next to each other, so that $X = X_{\mathtt{F}}$, where $\mathtt{F} = \{11\}$. This is called the golden mean shift.

Let $\mathcal{A} = \{0, 1\}$ and $F = \{f, g\}$ be a set of maps such that

x	$f(x)$	$g(x)$
0	1	0
1	0	0

If a functional shift $\mathcal{F}$ is equal to $F^{\mathbb{Z}}$, then $X_{\mathcal{F}}$ is the set of all binary sequences not containing contiguous 1s. Thus, $X_{\mathcal{F}}$ is equal to the golden mean shift X.

Example 2.5 (Sturmian shifts). Consider the circle map

$$T(x) = x + \alpha \bmod 1 \tag{6}$$

with irrational $\alpha \in [0, 1]$. Let $S \subset \{0, 1\}^{\mathbb{Z}}$ be a set

$$S = \left\{ s \in \{0, 1\}^{\mathbb{Z}} \mid x \in [0, 1),\ \forall n \in \mathbb{Z}\ s_n = \left\lfloor \frac{T^n(x)}{\alpha} \right\rfloor \right\}, \tag{7}$$

where $\lfloor x \rfloor$ is the integer part of x. S is not necessarily closed, but it is shift-invariant, and so its closure $X_\alpha = \mathrm{Cl}(S)$ is a shift space, called a Sturmian shift.

Let $F = \{f, g\}$ be a set of maps given by

x	$f(x)$	$g(x)$
0	1	$1 - \lfloor 2\alpha \rfloor$
1	0	$1 - \lfloor 2\alpha \rfloor$

and X_α a Sturmian shift with irrational α. Here, $\mathcal{F} = \phi(X_\alpha)$, where a 2-block map $\Phi : \{0, 1\}^2 \to F$ is defined by

$$\Phi(x) = \begin{cases} f, & \text{if } x = 01 \text{ or } x = 10, \\ g, & \text{otherwise} \end{cases} \tag{8}$$

and a map $\phi : X_\alpha \to F^{\mathbb{Z}}$ is a sliding block code induced by Φ. Since α is irrational, 00 or 11 must appear in x, so that for any $h \in \mathcal{F}$ there is a unique sequence $x \in X_\alpha$ such that $\phi(x) = h$ and $x_{n+1} = h_n(x_n)$ for all $n \in \mathbb{Z}$. Therefore, this functional shift $\mathcal{F}$ satisfies $\mathcal{F} \cong X_\alpha$ and $X_{\mathcal{F}} = X_\alpha$.

3. Entropy

This section will describe the properties of the relationship between functional shifts and generated shifts by analysing the entropy for those shifts.

The entropy of a shift space X is defined by

$$h(X) = \lim_{n \to \infty} \frac{1}{n} \log_2 |\mathcal{B}_n(X)|. \tag{9}$$

Note that $\log_2 |\mathcal{A}|$ is the upper limit of $h(X)$ for any $X \subset \mathcal{A}^{\mathbb{Z}}$. The entropy of X is a measure of the growth rate of the number of n-blocks occurring in points in X. Furthermore, if a distance function d of X is determined by

$$d(x, y) = \begin{cases} 2^{-|k|}, & \text{if } x_k \neq y_k \text{ and } x_i = y_i \quad \text{for } -|k| < i < |k|, \\ 0, & \text{if } x = y, \end{cases} \tag{10}$$

then the entropy of X is equal to the topological entropy of the shift map on the metric space (X, d) [18]. Hence, we regard the entropy of a shift space as the topological entropy.

It is known that the topological entropy is an indicator of the complexity of the dynamics and that it is invariant under topological conjugacy. The existence of positive topological entropy implies that a system is chaotic, because the topological entropy measures the mixing rate of the global orbit structure of the system.

The following theorem is an important result of the entropy of functional shifts.

Theorem 3.1. *If $\mathcal{F}$ is a functional shift, then $h(X_{\mathcal{F}}) \leqslant h(\mathcal{F})$.*

Proof. Let $\varphi_n : \mathcal{B}_n(\mathcal{F}) \to 2^{\mathcal{A}^n}$ be defined by

$$\varphi_n(f) = \{x \in \mathcal{B}_n(X_\mathcal{F}) \mid \exists a \in \mathcal{A}\ x_0 = f_0(a) \wedge \forall i\ x_{i+1} = f_{i+1}(x_i)\}, \tag{11}$$

where 2^X denotes the power set of X. It is clear that $|\varphi_n(f)| \leqslant |\mathcal{A}|$ for all $f \in \mathcal{B}_n(\mathcal{F})$ and

$$\mathcal{B}_n(X_\mathcal{F}) \subset \bigcup_{f \in \mathcal{B}_n(\mathcal{F})} \varphi_n(f). \tag{12}$$

Thus, $|\mathcal{B}_n(X_\mathcal{F})| \leqslant |\mathcal{B}_n(\mathcal{F})||\mathcal{A}|$. Accordingly,

$$\begin{aligned} h(X_\mathcal{F}) &= \lim_{n\to\infty} \frac{1}{n} \log_2 |\mathcal{B}_n(X_\mathcal{F})| \\ &\leqslant \lim_{n\to\infty} \frac{1}{n} \log_2 (|\mathcal{B}_n(\mathcal{F})||\mathcal{A}|) \\ &= \lim_{n\to\infty} \frac{1}{n} \log_2 |\mathcal{B}_n(\mathcal{F})| \\ &= h(\mathcal{F}). \end{aligned} \tag{13}$$

Hence, $h(X_\mathcal{F}) \leqslant h(\mathcal{F})$. ∎

Example 3.2. Let $\mathcal{F}$ be a functional shift given in example 2.4. Here, $h(\mathcal{F}) = \log_2 2$ and $h(X_\mathcal{F}) = \log_2 ((1 + \sqrt{5})/2)$ (this derivation is given in [18]). Thus, $h(X_\mathcal{F}) < h(\mathcal{F})$.

The key point in the proof of theorem 3.1 is to satisfy a condition $|\mathcal{B}_n(X_\mathcal{F})| \leqslant |\mathcal{B}_n(\mathcal{F})||\mathcal{A}|$ for any $n \in \mathbb{N}$. The condition implies that for any sequence of symbols $x \in X_\mathcal{F}$ there exists one or more sequences of functions $f \in \mathcal{F}$ that are restricted by $x_{n+1} = f_n(x_n)$ for all $n \in \mathbb{Z}$. Since the plural sequences (in some cases, it is infinitely so) in $\mathcal{F}$ can correspond to a sequence in $X_\mathcal{F}$, the entropy of $\mathcal{F}$ is the same as or larger than that of $X_\mathcal{F}$.

If F is a set of maps on $\mathcal{A}$ such that

$$\forall f, g \in F\ f \neq g \ \Rightarrow\ \forall x \in \mathcal{A}\ f(x) \neq g(x), \tag{14}$$

then every functional shift $\mathcal{F} \subset F^\mathbb{Z}$ satisfies $h(X_\mathcal{F}) = h(\mathcal{F})$, because $|\mathcal{B}_n(X_\mathcal{F})| \geqslant |\mathcal{B}_n(\mathcal{F})|$ for all $n \in \mathbb{N}$. Hence,

$$\exists f, g \in F\ \exists x \in \mathcal{A}\ f \neq g \wedge f(x) = g(x) \tag{15}$$

is a necessary condition in order to realize $h(X_\mathcal{F}) < h(\mathcal{F})$. For example, if $|\mathcal{A}| < |F|$ and $\mathcal{F} = F^\mathbb{Z}$, then F satisfies equation (15) and $h(X_\mathcal{F}) < h(\mathcal{F})$. However, there exists a case in which F satisfies equation (15) and $h(X_\mathcal{F}) = h(\mathcal{F})$. The example 2.5 is one such instance, because $\mathcal{F}$ and $X_\mathcal{F}$ are conjugate and the entropy is invariant under conjugacy.

From theorem 3.1, we may consider that the degree of complexity of a functional shift $\mathcal{F}$ is greater than that of $X_\mathcal{F}$ from the viewpoint of dynamics. However, satisfying such a relationship is not necessarily required in the computational point of view. In the next section we turn to the computational power of shift spaces, and compare the languages of functional shifts with those of generated shifts.

4. Computation in functional shifts

In section 3, we compared the complexity of functional shifts with generated shifts based on entropy. Entropy measures the exponential growth rate of the number of orbits distinguished in limited accuracy, i.e. it represents sensitivity to the initial conditions of a dynamical system. On the other hand, there is the complexity of languages given by the Chomsky hierarchy which differs from that of entropy. The complexity described by the Chomsky hierarchy is

based on the memory size of the automata that recognize languages (see the appendix). In this sense, languages are classified into four classes: regular languages, which do not need any memory; context free languages, which have just a stack; context sensitive languages, which have a storage capacity proportional to the input word length; and type-0 languages, which have unlimited memory.

The memory size of an automaton is deeply related to the long-range correlation and unpredictability of a dynamical system. Badii and Politi [19] have discussed the relationship between memory size and the properties of dynamical systems with some examples of physical systems corresponding to formal languages in the Chomsky hierarchy. For example, random walks with two reflecting barriers are dynamics whose languages are regular, because the domain surrounded by two barriers can be considered to express a finite state. Those dynamics can be described by the Markov graph, so they are stationary and ergodic. Random walks with one reflecting barrier are dynamics whose languages are context free, because the domain on the semi-lattice that makes one barrier the starting point can be considered to express a stack. The fact that there is no restriction about distance from a barrier brings long-range correlation to those random walks. For another example, self-avoiding random walks are dynamics whose languages are more complex than context free languages. Some dynamics whose languages are not context free have strong unpredictability. In section 5, we discuss this property in detail.

In this section, we compare functional shifts with generated shifts, by focusing on how the language of a shift space belongs to the Chomsky hierarchy of formal languages. Note that not every collection of blocks is the language of a shift space. Namely, if X is a shift space and $w \in \mathcal{B}(X)$, then

- every sub-block of w belongs to $\mathcal{B}(X)$, and
- there are non-empty blocks u and v in $\mathcal{B}(X)$ such that $uwv \in \mathcal{B}(X)$.

Given a class of languages of functional shifts $\mathscr{F}$, a class of languages of generated shifts $\mathscr{G}$ is given by $\mathscr{G} = \{\mathcal{B}(X_{\mathcal{F}}) \mid \mathcal{B}(\mathcal{F}) \in \mathscr{F}, \mathcal{F} \text{ is a functional shift}\}$. We now consider the inclusion relation between $\mathscr{F}$ and $\mathscr{G}$. From the properties of entropy in functional shifts, we may consider that $\mathscr{F}$ is at least as complex as $\mathscr{G}$ from the dynamical viewpoint, because for any shift space X whose language is in $\mathcal{G}$ there exists a functional shift $\mathcal{F}$ whose language is in $\mathscr{F}$ satisfying $X_{\mathcal{F}} = X$, so that $h(X) \leqslant h(\mathcal{F})$. Thus, if the complexity of entropy corresponds to that of computation, we shall expect $\mathscr{G}$ as the subclass of $\mathscr{F}$. It is, however, known that the complexities of entropy and computation generally do not correspond. For instance, consider the language of a periodic shift space, which only contains periodic sequences, and that of a full shift. Both are contained in a class of regular languages, which is the lowest computation class in the Chomsky hierarchy. However, the former has minimal entropy, and the latter has maximal entropy. The fact that shift spaces with different entropy belong to the same computational class makes it generally difficult to clarify the relationship between entropy and the complexity of computation. Hence, it is worthwhile to investigate this relationship using functional shifts. We will show the inclusion relation between $\mathscr{F}$ and $\mathscr{G}$ in the case where $\mathscr{F}$ and $\mathscr{G}$ belong to the Chomsky hierarchy, and discuss the complexity of dynamics and computation with these results.

We first prove the following theorem to be a basic principle of functional shifts.

Theorem 4.1. *For any shift space X over $\mathcal{A}$, there is a functional shift $\mathcal{F}$ over F and a* 1*-block map $\Phi : \mathcal{A} \to F$ such that*

- $X_{\mathcal{F}} = X$,
- Φ *is a one-to-one mapping,*
- ϕ *induced by* Φ *satisfies* $\phi(X) = \mathcal{F}$.

Proof. For any $a \in \mathcal{A}$, we define a function $f_a : \mathcal{A} \to \mathcal{A}$ by

$$\forall x \in \mathcal{A}\ f_a(x) = a. \tag{16}$$

Let $F = \{f_a \mid a \in \mathcal{A}\}$, and Φ be defined by

$$\forall a \in \mathcal{A}\ \Phi(a) = f_a. \tag{17}$$

Clearly Φ is a one-to-one mapping, so that $\phi(X)$ is a shift space. Now, a functional shift $\mathcal{F}$ is defined by $\mathcal{F} = \phi(X)$. Then,

$$\begin{aligned} x \in X_{\mathcal{F}} &\Leftrightarrow \exists f \in \mathcal{F} \forall i \in \mathbb{Z} x_{i+1} = f_i(x_i) \\ &\Leftrightarrow \exists f \in \mathcal{F} \forall i \in \mathbb{Z} \Phi(x_{i+1}) = f_i \\ &\Leftrightarrow x \in X. \end{aligned} \tag{18}$$

Thus, $X_{\mathcal{F}} = X$. ∎

From theorem 4.1 we can get the next corollary.

Corollary 4.2. *Let $\mathscr{F}$ be a class of languages of shift spaces. Suppose that $\mathscr{G}$ is a class of languages of generated shifts given by the functional shifts whose languages belong to $\mathscr{F}$. Then, $\mathscr{F} \subset \mathscr{G}$.*

Proof. Let L be a language in $\mathscr{F}$, and X be a shift space defined by $\mathcal{B}(X) = L$. By theorem 4.1, there is a functional shift $\mathcal{F}$ such that $\mathcal{B}(X) = \mathcal{B}(\mathcal{F})$ and $X_{\mathcal{F}} = X$. Then, $L \in \mathscr{G}$. ∎

From corollary 4.2, we obtain that any class of the languages of functional shifts is contained in that of generated shifts given by them. However, there is still the open problem of whether there exists a class of languages of functional shifts which is a proper subset of languages of shift spaces generated by the functional shifts. We can study the relationship between the languages of functional shifts and those of generated shifts to bring this problem into focus.

Hereafter, the next lemma is the key ingredient in each proof.

Lemma 4.3. *Let*

$$\mathcal{D}_{\mathcal{F}}(n, x) = \{y = ax \mid a \in \mathcal{A}^n, \exists f \in \mathcal{B}_{|y|-1}(\mathcal{F})\ \forall i\ y_{i+1} = f_i(y_i)\}. \tag{19}$$

Then, $x \in \mathcal{B}(X_{\mathcal{F}})$ iff $\lim_{n\to\infty} \mathcal{D}_{\mathcal{F}}(n, x) \neq \emptyset$.

Proof. Suppose that $x \in \mathcal{B}(X_{\mathcal{F}})$, and $y \in X_{\mathcal{F}}$ having sub-block x. There is a bi-infinite sequence $f \in \mathcal{F}$ such that $y_{i+1} = f_i(y_i)$ for any $i \in \mathbb{Z}$. Since x is a sub-block of y, an integer k such as $x_1 = y_k$ exists. Hence, $y_{k-n} \cdots y_{k-1} x_1 \cdots x_{|x|} \in \mathcal{D}_{\mathcal{F}}(n, x)$ for any $n \in \mathbb{N}$, so that $\lim_{n\to\infty} \mathcal{D}_{\mathcal{F}}(n, x) \neq \emptyset$.

Conversely, suppose that $\lim_{n\to\infty} \mathcal{D}_{\mathcal{F}}(n, x) \neq \emptyset$. Then, there is an infinite sequence $y = \cdots a_{-1} a_0 x_1 \cdots x_{|x|}$ (by Koenig's lemma) and a bi-infinite sequence $f \in \mathcal{F}$ such that $y_{i+1} = f_i(y_i)$ for $-\infty < i < |x|$. If a bi-infinite sequence z is defined by

$$z_i = \begin{cases} y_i, & \text{if } i \leqslant |x|, \\ f_{i-1}(z_{i-1}), & \text{otherwise,} \end{cases} \tag{20}$$

then $z \in X_{\mathcal{F}}$ because $z_{i+1} = f_i(z_i)$ for any $i \in \mathbb{Z}$. Since x is a sub-block of z, $x \in \mathcal{B}(X_{\mathcal{F}})$. ∎

4.1. Shifts of finite type and sofic shifts

Here, we study the case in which a functional shift is a shift of finite type or a sofic shift. We first define shifts of finite type and sofic shifts.

A shift of finite type is a shift space that can be described by a finite set of forbidden blocks. Although shifts of finite type are the simplest shifts, they are significant in the dynamical systems theory. If a dynamical system is hyperbolic, then the system has a Markov partition and a topological conjugacy to a shift of finite type.

Sofic shifts are defined by using graphs, called labelled graphs, whose edges are assigned labels. A graph G consists of a finite set $\mathcal{V} = \mathcal{V}(G)$ of vertices (or states) together with a finite set $\mathcal{E} = \mathcal{E}(G)$ of edges. Each edge $e \in \mathcal{E}$ starts at a vertex denoted by $i(e) \in \mathcal{V}(G)$ and terminates at a vertex $t(e) \in \mathcal{V}(G)$ (which can be the same as $i(e)$). Equivalently, the edge e has an initial state $i(e)$ and a terminal state $t(e)$. A labelled graph $\mathcal{G}$ is a pair $(G, \mathcal{L})$, where G is a graph with edge set $\mathcal{E}$, and the labelling $\mathcal{L} : \mathcal{E} \to \mathcal{A}$ assigns each edge e of G to a label $\mathcal{L}(e)$ in $\mathcal{A}$. Let $X_{\mathcal{G}}$ be denoted by

$$X_{\mathcal{G}} = \{x \in \mathcal{A}^{\mathbb{Z}} \mid \exists e \in \mathcal{E}^{\mathbb{Z}}\ \forall i \in \mathbb{Z}\ t(e_i) = i(e_{i+1}) \wedge x_i = \mathcal{L}(e_i)\}. \tag{21}$$

A subset X of the full shift $\mathcal{A}^{\mathbb{Z}}$ is a sofic shift if $X = X_{\mathcal{G}}$ for some labelled graph $\mathcal{G}$. Since a labelled graph is regarded as a state diagram of a finite state automaton, the language of a sofic shift is regular.

It is known that a shift space is sofic iff it is a factor of a shift of finite type [18]. Since an identity function on a shift space is a factor code, shifts of finite type are sofic. Moreover, the class of sofic shifts is larger than that of shifts of finite type, because not all sofic shifts have finite type. For example, the even shift, which can be described by the collection $\{10^{2n+1}1 \mid n \geqslant 0\}$ of forbidden blocks, is a sofic shift that does not have finite type.

Let us prove the following theorems as the case in which functional shifts are shifts of finite type or sofic shifts.

Theorem 4.4. *(1) If X is a sofic shift, then there is a functional shift $\mathcal{F}$ such that $\mathcal{F}$ has finite type and $X = X_{\mathcal{F}}$.*

(2) If a functional shift $\mathcal{F}$ is sofic, then $X_{\mathcal{F}}$ is sofic (so that if a functional shift $\mathcal{F}$ has finite type, $X_{\mathcal{F}}$ is sofic).

Proof. (1) Suppose that X is a sofic shift over $\mathcal{A}$, and $\mathcal{G} = (G, \mathcal{L})$ is a labelled graph such that $X = X_{\mathcal{G}}$. If a is an edge of G, then $f_a : \mathcal{A} \cup \mathcal{E} \cup \{\delta\} \to \mathcal{A} \cup \mathcal{E} \cup \{\delta\}$ (where $\mathcal{A} \cap \mathcal{E} = \emptyset$ and $\delta \notin \mathcal{A} \cup \mathcal{E}$) is defined by

$$f_a(x) = \begin{cases} \delta, & \text{if } x = a, \\ \mathcal{L}(a), & \text{otherwise.} \end{cases} \tag{22}$$

Let $F = \{f_a \mid a \in \mathcal{E}\}$ and $\mathtt{F} = \{f_a f_b \in F^2 \mid t(a) \neq i(b)\}$. Recall that $i(a)$ is an initial state and $t(a)$ is a terminal state of a. If a functional shift $\mathcal{F}$ over F can be described by $\mathtt{F}$, then $X_{\mathcal{F}} = X$. Since $\mathtt{F}$ is a finite set, $\mathcal{F}$ is a shift of finite type.

(2) Suppose that $\mathcal{F}$ is a sofic shift over F which is a set of maps on $\mathcal{A}$, and F' is a collection of maps on F. By (1), there is a functional shift $\mathcal{F}'$ over F' such that $\mathcal{F}'$ has finite type and $X_{\mathcal{F}'} = \mathcal{F}$. Let

$$X = \{\langle x, f, g\rangle \in (\mathcal{A} \times F \times F')^{\mathbb{Z}} \mid g \in \mathcal{F}',\ \forall i \in \mathbb{Z}\ x_{i+1} = f_i(x_i) \wedge f_{i+1} = g_i(f_i)\}, \tag{23}$$

be a set of elements, which are sequences of 3-tuples $(\cdots\langle x_0, f_0, g_0\rangle\langle x_1, f_1, g_1\rangle\cdots)$, and $\mathtt{F}$ be a finite set of forbidden blocks such that $X_{\mathtt{F}} = \mathcal{F}'$. Then,

$$\begin{aligned} \tilde{\mathtt{F}} = {} & \{\langle x, f, g\rangle \in (\mathcal{A} \times F \times F')^2 \mid x_1 \neq f_0(x_0) \vee f_1 \neq g_0(f_0)\} \\ & \cup \{\langle x, f, g\rangle \in (\mathcal{A} \times F \times F')^* \mid g \in \mathtt{F}\} \end{aligned} \tag{24}$$

is a set of forbidden blocks such that $X_{\tilde{\mathtt{F}}} = X$. Here, $\tilde{\mathtt{F}}$ is a finite set because $\mathtt{F}$ is finite. Suppose that $\Phi : (\mathcal{A} \times F \times F') \to \mathcal{A}$ is a 1-block map such that $\Phi(\langle x, f, g\rangle) = x$. Since a

sliding block code $\phi : X \to X_{\mathcal{F}}$ induced by Φ is onto, ϕ is a factor code. If a shift space is a factor of a shift of finite type, then it is sofic. Thus, $X_{\mathcal{F}}$ is a sofic shift. ■

The next is an instance satisfying theorem 4.4 (1).

Example 4.5. Let $\mathcal{A} = \{0, 1\}$, and $F = \{f_a, f_b, f_c\}$ be a set of functions on $\mathcal{A}$ such that

x	$f_a(x)$	$f_b(x)$	$f_c(x)$
0	1	0	0
1	1	0	1

If a functional shift $\mathcal{F}$ can be described by a set of forbidden blocks $\mathtt{F} = \{f_a f_c, f_b f_a, f_b f_b, f_c f_c\}$, then $\mathcal{F}$ is a shift of finite type and $X_{\mathcal{F}}$ is the even shift.

4.2. Context free languages

This subsection studies the case in which the language of a functional shift is a context free language. The shift dynamics on some shift spaces with languages that are context free have long-range correlations, because stacks can hold memories infinitely. Moreover, if the shift spaces are probability measure spaces, the phase transition often appears in this dynamics [19].

We begin by proving that if the language of a functional shift $\mathcal{F}$ is context free, then there is a number $p \in \mathbb{N}$ such that for any $x \in \mathcal{A}^*$ $\mathcal{D}_{\mathcal{F}}(p, x) \neq \emptyset$ iff $\lim_{n\to\infty} \mathcal{D}_{\mathcal{F}}(n, x) \neq \emptyset$, by using the pumping lemma[1]. Next, we prove that if the language of $\mathcal{F}$ is context free, then that of $X_{\mathcal{F}}$ is so.

Lemma 4.6. *Suppose that $\mathcal{F}$ is a functional shift and $\mathcal{B}(\mathcal{F})$ is a context free language. There is a natural number p such that*

$$\forall x \in \mathcal{A}^* \qquad \mathcal{D}_{\mathcal{F}}(p, x) \neq \emptyset \;\Leftrightarrow\; \lim_{n\to\infty} \mathcal{D}_{\mathcal{F}}(n, x) \neq \emptyset. \tag{25}$$

Proof. Let $G = \{N, F, P, S\}$ be a context free grammar such that $\mathcal{B}(\mathcal{F}) = L(G)$, where $L(G)$ denotes a formal language generated by G. Suppose, without loss of generality, that G is in Chomsky normal form[2]. Now a formal grammar $G' = \{N', \mathcal{A}, P', S'\}$ is defined as follows. A set N' of nonterminal symbols is equal to $\{A_{ab} \mid A \in N, a, b \in \mathcal{A}\}$. P' is a set of productions determined by the following rules:

- $S' \to aS_{ab} \in P'$ for any $a, b \in \mathcal{A}$;
- $A_{ab} \to B_{ac}C_{cb} \in P'$ iff $A \to BC \in P$, where $A, B, C \in N$ and $a, b, c \in \mathcal{A}$;
- $A_{ab} \to b \in P'$ iff $A \to f \in P$ and $f(a) = b$, where $A \in N$, $f \in F$, and $a, b \in \mathcal{A}$.

Clearly, G' is a context free grammar; furthermore,

$$L(G') = \bigcup_{x\in\mathcal{A}^*} \mathcal{D}_{\mathcal{F}}(0, x) = \bigcup_{x\in\mathcal{A}^*} \bigcup_{n\in\mathbb{N}} \mathcal{D}_{\mathcal{F}}(n, x) \tag{26}$$

because $x \in L(G')$ iff there is a block $f \in L(G) = \mathcal{B}(\mathcal{F})$ such that $|f| = |x| - 1$ and $x_{i+1} = f_i(x_i)$ for $1 \leqslant i < |x|$.

[1] In the theory of formal languages, the pumping lemma provides the necessary conditions for languages to be context free. The pumping lemma for context free languages is as follows: if language L is context free, then there is a natural number p such that if $r = uvwxy \in L$ and $|r| > p$, then $|vx| \geqslant 1$, $|vwx| \leqslant p$, and for any $i \geqslant 0$, $uv^i wx^i y \in L$.

[2] A formal grammar $G = (V_N, V_T, P, S)$ is in Chomsky normal form iff all productions are of the form $A \to BC$ or $A \to a$, where $A, B, C \in V_N$ and $a \in V_T$. Every formal grammar in Chomsky normal is context free, and conversely, every context free grammar that does not generate an empty string can be transformed into an equivalent one which is in Chomsky normal form.

From the pumping lemma, there is a natural number p, such that, if $r = uvwxy \in L(G')$ and $|r| > p$, then,

- $|vx| \geqslant 1$,
- $|vwx| \leqslant p$,
- for any $i \geqslant 0$, $uv^i wx^i y \in L(G')$.

Hence, if $rs \in L(G')$, i.e. $\mathcal{D}_{\mathcal{F}}(p, s) \neq \emptyset$, then $\mathcal{D}_{\mathcal{F}}(p + |vx|n, s) \neq \emptyset$ for all $n \in \mathbb{N}$. Note that if $\mathcal{D}_{\mathcal{F}}(n, s) \neq \emptyset$ and $m \leqslant n$ then $\mathcal{D}_{\mathcal{F}}(m, s) \neq \emptyset$. Therefore, $\mathcal{D}_{\mathcal{F}}(p, s) \neq \emptyset$ iff $\lim_{n\to\infty} \mathcal{D}_{\mathcal{F}}(n, s) \neq \emptyset$. ∎

Since we use a nondeterministic pushdown automaton (NPDA) to prove the next theorem, we give the formal definition of NPDA. A NPDA is a 6-tuple $M = \{Q, \Sigma, \Gamma, \delta, q_0, Z, E\}$, where Q is a finite set of states, Σ is an alphabet (defining what set of input strings the automaton operates on), Γ is a stack alphabet (specifying the set of symbols that can be pushed onto the stack), $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to 2^{Q\times\Gamma^*}$ is a transition function, $q_0 \in Q$ is a starting state, $Z \in \Gamma$ is a starting stack symbol, and $E \subset Q$ is a set of final (or accepting) states.

Given a NPDA $M = \{Q, \Sigma, \Gamma, \delta, q_0, Z, E\}$, the relation $\vdash_M \subset Q\times\Sigma^*\times\Gamma^*\times Q\times\Sigma^*\times\Gamma^*$ is defined by

$$(p, \beta) \in \delta(q, a, A) \quad \Leftrightarrow \quad (q, aw, A\gamma) \underset{M}{\vdash} (p, w, \beta\gamma) \tag{27}$$

and the reflexive and transitive closure $\vdash_M$ is denoted by $\vdash_M^*$. M accepts an input string x if there are $p \in E$ and $\gamma \in \Gamma^*$ such that $(q_0, x, Z) \vdash_M^* (p, \epsilon, \gamma)$. The language recognized by M is the set

$$L(M) = \{x \in \Sigma^* \mid \exists p \in E \exists \gamma \in \Gamma^* (q_0, x, Z) \overset{*}{\underset{M}{\vdash}} (p, \epsilon, \gamma)\}. \tag{28}$$

It is known that a language L is a context free language iff $L = L(M)$ for some NPDA M.

Theorem 4.7. *If $\mathcal{F}$ is a functional shift and $\mathcal{B}(\mathcal{F})$ is a context free language, then $\mathcal{B}(X_{\mathcal{F}})$ is also a context free language.*

Proof. We will construct a NPDA which can recognize the language $\mathcal{B}(X_{\mathcal{F}})$.

Since $\mathcal{B}(\mathcal{F})$ is a context free language, $\mathcal{B}(\mathcal{F})^{\mathbf{R}} = \{x^{\mathbf{R}} \mid x \in \mathcal{B}(\mathcal{F})\}$ is also context free, where $x^{\mathbf{R}}$ denotes the reversal of block x. Thus, a NPDA $M = \{Q, F, \Gamma, \delta, q_0, Z, E\}$ to recognize $\mathcal{B}(\mathcal{F})^{\mathbf{R}}$ exists. Here, a NPDA $M' = \{Q', \mathcal{A}, \Gamma', \delta', q_0', Z', E'\}$ is defined as follows: let $N = \{0, 1, \ldots, p\}$ and

- $Q' = (Q \times \mathcal{A} \times N) \cup \{q_0'\}$,
- $\Gamma' = \Gamma \cup \{Z'\}$,
- $E' = \{(q, a, i) \in Q' \mid q \in E \wedge i = p\}$;

next, δ' is determined by the following:

- $\delta'(q_0', a, Z') = \{(\langle q_0, a, 0\rangle, Z)\}$ for any $a \in \mathcal{A}$;
- $\delta'(q_0', \epsilon, Z') = \{(\langle q_0, a, 1\rangle, Z) \mid a \in \mathcal{A}\}$;
- suppose that $q \in Q$, $a \in \mathcal{A}$, and $A \in \Gamma$; if $b \in \mathcal{A}$ then

$$\delta'(\langle q, a, 0\rangle, b, A) = \{(\langle r, b, 0\rangle, g) \mid \exists f \in F(r, g) \in \delta(q, f, A) \wedge f(b) = a\}, \tag{29}$$

else if $b = \epsilon$, then

$$\delta'(\langle q, a, 0\rangle, \epsilon, A) = \{(\langle r, a, 0\rangle, g) \mid (r, g) \in \delta(q, \epsilon, A)\} \cup \{(\langle q, a, 1\rangle, A)\}; \tag{30}$$

- for any $1 \leqslant i \leqslant p$,

$$\begin{aligned}\delta'(\langle q, a, i\rangle, \epsilon, A) = {} & \{(\langle r, b, i + 1\rangle, g) \mid b \in \mathcal{A}, \exists f \in F(r, g) \in \delta(q, f, A) \wedge f(b) = a\} \\ & \cup \{(\langle r, a, i\rangle, g) \mid (r, g) \in \delta(q, \epsilon, A)\}.\end{aligned} \tag{31}$$

Given an input $x \in \mathcal{A}^*$, M' accepts x iff there are $y = xa \in \mathcal{A}^{|x|+p}$ and $f \in F^{|x|+p-1}$ such that M accepts f and $y_{i-1} = f_i(y_i)$ for $1 < i \leqslant |y|$. Thus, $y^{\mathbf{R}} \in \mathcal{D}_{\mathcal{F}}(p, x^{\mathbf{R}})$, so that M' accepts x iff $\mathcal{D}_{\mathcal{F}}(p, x^{\mathbf{R}}) \neq \emptyset$. By lemma 4.6, there is a natural number p such that $\mathcal{D}_{\mathcal{F}}(p, x^{\mathbf{R}}) \neq \emptyset$ iff $\lim_{n\to\infty} \mathcal{D}_{\mathcal{F}}(n, x^{\mathbf{R}}) \neq \emptyset$. Hence, there is a NPDA M' such that M' accepts x iff $x^{\mathbf{R}} \in \mathcal{B}(X_{\mathcal{F}})$ by lemma 4.3. Accordingly, $\mathcal{B}(X_{\mathcal{F}})$ is a context free language. ■

4.3. Context sensitive, recursive, and r.e. sets

Let us consider the case in which the language of a functional shift $\mathcal{F}$ is not context free. A class of dynamical systems, called generalized shifts, has been proposed by Moore [10], as corresponding to the class of languages more complicated than context free languages. The class of generalized shifts is equivalent to that of Turing machines, and they can be embedded in smooth maps in $\mathbb{R}^2$ or smooth flows in $\mathbb{R}^3$.

In this case, the problem of determining the predicate $x \in \mathcal{B}(X_{\mathcal{F}})$ is more difficult than in the above subsection. This is because there does not necessarily exist a number p to satisfy that for any $x \in \mathcal{A}^*$ $\mathcal{D}_{\mathcal{F}}(q, x) \neq \emptyset$ iff $\lim_{n\to\infty} \mathcal{D}_{\mathcal{F}}(n, x) \neq \emptyset$, in the case in which the language of a functional shift $\mathcal{F}$ is not context free, while it always exists in context free cases.

We suppose that F is a set of bijections and $\mathcal{F}$ is a functional shift over F. From the definition of $\mathcal{D}_{\mathcal{F}}$, it is clear that $\mathcal{D}_{\mathcal{F}}(0, x) \neq \emptyset$ is a sufficient condition for $\lim_{n\to\infty} \mathcal{D}_{\mathcal{F}}(n, x) \neq \emptyset$. Then, we can prove the following theorems.

Theorem 4.8. *Let F be a set of bijections on $\mathcal{A}$, and $\mathcal{F}$ be a functional shift over F. If $\mathcal{B}(\mathcal{F})$ is a context sensitive language, then $\mathcal{B}(X_{\mathcal{F}})$ is context sensitive.*

Proof. Let $G = \{N, F, P, S\}$ be a context sensitive grammar to generate $\mathcal{B}(\mathcal{F})$. $G' = \{N', \mathcal{A}, P', S'\}$ is determined by the following conditions:

- $N' = N \cup F \cup \{S'\}$, where $S' \notin N \cup F$.
- P' contains only productions satisfying the following restrictions:
  - (a) for any $a \in \mathcal{A}$, $S' \to aS \in P'$ and $S' \to a \in P'$;
  - (b) for any $\alpha, \beta \in (N \cup F)^*$, if $\alpha \to \beta \in P$ then $\alpha \to \beta \in P'$;
  - (c) if $f \in F$ and $a \in \mathcal{A}$, then $af \to af(a) \in P'$.

Then, G' is a context sensitive grammar because if $\alpha \to \beta \in P'$ then $|\alpha| \leqslant |\beta|$.

Let $x \in L(G')$. Since there is a block $f \in L(G) = \mathcal{B}(\mathcal{F})$ such that $|f| = |x| - 1$ and $x_{i+1} = f_i(x_i)$ for $1 \leqslant i < |x|$, then $\mathcal{D}_{\mathcal{F}}(0, x) \neq \emptyset$. Thus, $\lim_{n\to\infty} \mathcal{D}_{\mathcal{F}}(n, x) \neq \emptyset$ because every function in F is a bijection, so that $x \in \mathcal{B}(X_{\mathcal{F}})$ by lemma 4.3. Hence $L(G') \subset \mathcal{B}(X_{\mathcal{F}})$.

Conversely, let $x \in \mathcal{B}(X_{\mathcal{F}})$. Clearly, there is a block $f \in \mathcal{B}_{|x|-1}(\mathcal{F})$ such that $x_{i+1} = f_i(x_i)$ for any $1 \leqslant i < |x|$. Thus, f can be derived from S in G. Then,

$$\begin{aligned} S' &\underset{G'}{\Longrightarrow} x_1 S \\ &\underset{G'}{\overset{*}{\Longrightarrow}} x_1 f \\ &\underset{G'}{\overset{*}{\Longrightarrow}} x, \end{aligned} \tag{32}$$

so that $x \in L(G')$. Hence, $\mathcal{B}(X_{\mathcal{F}}) \subset L(G')$. Accordingly, $L(G') = \mathcal{B}(X_{\mathcal{F}})$ and $\mathcal{B}(X_{\mathcal{F}})$ is a context sensitive language. ■

Theorem 4.9. *Let F be a set of bijections on $\mathcal{A}$, and $\mathcal{F}$ be a functional shift over F. If $\mathcal{B}(\mathcal{F})$ is r.e., then $\mathcal{B}(X_{\mathcal{F}})$ is r.e.*

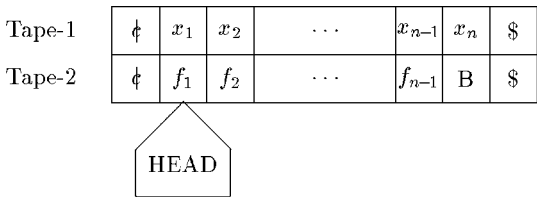

Figure 1. Illustration of the LBA M_2.

Proof. Suppose that G is a type-0 grammar to generate $\mathcal{B}(\mathcal{F})$, and G' is defined in a manner similar to that in the proof of theorem 4.8. As discussed in that proof, G' is also a type-0 grammar and $L(G') = \mathcal{B}(X_{\mathcal{F}})$. ∎

In the general case in which some functions in F may be not bijections, it is difficult to determine where the languages of generated shifts are located in the Chomsky hierarchy. To locate a collection of forbidden blocks is, however, an easier task than studying this problem.

Theorem 4.10. *Suppose that $\mathcal{F}$ is a functional shift and $\mathcal{B}(\mathcal{F})$ is context sensitive. Then, there is a context sensitive language* F *of forbidden blocks such that $X_{\mathtt{F}} = X_{\mathcal{F}}$.*

Proof. Let

$$\mathtt{F} = \{x \in \mathcal{A}^* \mid \mathcal{D}_{\mathcal{F}}(0, x) = \emptyset\}. \tag{33}$$

We now show that F is a collection of forbidden blocks such that $X_{\mathtt{F}} = X_{\mathcal{F}}$. Suppose that x is a bi-infinite sequence in $\mathcal{A}^{\mathbb{Z}}$ such as $x \notin X_{\mathcal{F}}$. Then, there is a block y, which is not contained in $\mathcal{B}(X_{\mathcal{F}})$, occurring in x. Thus, by lemma 4.3, a natural number n such as $\mathcal{D}_{\mathcal{F}}(n, y) = \emptyset$ exists. For any $a \in \mathcal{A}^n$, $ay \in \mathtt{F}$ because $\mathcal{D}_{\mathcal{F}}(0, ay) = \emptyset$. Since there is a block $a \in \mathcal{A}^n$ such that x contains ay, some blocks in F occur in x. Conversely, suppose that $x \in \mathcal{A}^{\mathbb{Z}}$ contains a block $y \in \mathtt{F}$. Then, it is clear that $x \notin X_{\mathcal{F}}$. Accordingly, $x \notin X_{\mathcal{F}}$ iff there is a block in F which occurs in x. Hence $X_{\mathtt{F}} = X_{\mathcal{F}}$.

Next, we will explain the existence of a linear bounded automaton (LBA) which can recognize F. Let M_1 be a LBA to compute the following function

$$\varphi(x) = \begin{cases} 1, & \text{if } x \in \mathcal{B}(\mathcal{F}), \\ 0, & \text{if } x \notin \mathcal{B}(\mathcal{F}). \end{cases} \tag{34}$$

We construct a LBA M_2 with two separate tapes as follows. First, a block $x \in \mathcal{A}^*$ is inscribed on tape-1, and $f \in F^{|x|-1}$ on tape-2 (see figure 1). Then, M_2 carries out the following operations.

(1) The machine M_2 begins with the head resting in anticipation on the left-most cell. The machine repeatedly moves right and reads the cell value beneath the head until the right-most cell. When the machine finds i such that $x_{i+1} \neq f_i(x_i)$, M_2 accepts $\langle x, f \rangle$ and halts.
(2) The machine M_2 calls the subroutine M_1 with the block f on the tape-2, which returns the answer '0' or '1' as appropriate. If the answer is '0', that is, M_1 does not accept f, then M_2 accepts $\langle x, f \rangle$. In the other case, M_2 does not accept $\langle x, f \rangle$.

We now consider a machine M such that, for any input x, if M_2 accepts $\langle x, f \rangle$ for all $f \in F^{|x|-1}$ then M accepts x. Since to construct a LBA to enumerate $F^{|x|-1}$ is an easy task, from this explicit definition we can get the LBA M. For any x, M accepts x iff $\mathcal{D}_{\mathcal{F}}(0, x) = \emptyset$ because there is no $f \in \mathcal{B}_{|x|-1}(\mathcal{F})$ such that $x_{i+1} = f_i(x_i)$ for any $1 \leqslant i < |x|$. Hence M is a LBA to recognize F. ∎

Moreover, by using a proof similar to theorem 4.10, we can prove that if the language of a functional shift is a recursive set, then F is also recursive. A language is a recursive set if there exists a Turing machine which recognizes the language and always halts.

Theorem 4.11. *Suppose that $\mathcal{F}$ is a functional shift and $\mathcal{B}(\mathcal{F})$ is recursive. Then, there is a recursive set* F *of forbidden blocks such that $X_{\mathtt{F}} = X_{\mathcal{F}}$.*

Proof. Since $\mathcal{B}(\mathcal{F})$ is a recursive set, there is a Turing machine M_3 to compute the function φ in equation (34). Suppose that M_2 in the proof of theorem 4.10 calls subroutine M_3 instead of M_1. Then, M_2 and M are Turing machines that always halt after a finite amount of time, and M recognizes F. Thus, F is a recursive set. ■

4.4. The language of a generated shift beyond r.e.

In this subsection, we prove that there is a functional shift $\mathcal{F}$ satisfying that $\mathcal{B}(\mathcal{F})$ is r.e. and $\mathcal{B}(X_{\mathcal{F}})$ is not r.e. Here, a set A is r.e. iff the predicate $x \in A$ is partially decidable, i.e. the partial characteristic function

$$f(x) = \begin{cases} 1, & \text{if } x \in A, \\ \text{undefined}, & \text{if } x \notin A \end{cases} \tag{35}$$

is computable. Note that if a predicate $P(x)$ is partially decidable and undecidable, then $\neg P(x)$ is not partially decidable. For example, the following predicate

'a Turing machine of the Gödel number x eventually stops on input x'

is partially decidable and undecidable. In the proof of the next theorem, we show that $x \in \mathcal{B}(X_{\mathcal{F}})$ iff a Turing machine of the Gödel number x never halts on input x, in order to prove $x \in \mathcal{B}(X_{\mathcal{F}})$ is not partially decidable.

Theorem 4.12. *There is a function shift $\mathcal{F}$ such that $\mathcal{B}(\mathcal{F})$ is r.e. and $\mathcal{B}(X_{\mathcal{F}})$ is not r.e.*

Proof. For any $x \in \{01^n\delta \mid n \in \mathbb{N}\}$, let

$$\mathrm{num}(x) = \text{the number of 1s occurring in } x \tag{36}$$

and a Turing machine of the Gödel number n be denoted by T_n. We will show a functional shift $\mathcal{F}$ such that $\mathcal{B}(\mathcal{F})$ is r.e. and

$$x \in \mathcal{B}(X_{\mathcal{F}}) \quad \Leftrightarrow \quad T_{\mathrm{num}(x)} \text{ never halts on input } x. \tag{37}$$

Let $\mathcal{A} = \{0, 1, \delta\}$, $F = \{f, g, h\}$ be a set of maps defined by

x	$f(x)$	$g(x)$	$h(x)$
0	0	1	δ
1	δ	1	δ
δ	δ	δ	δ

and $\mathcal{F}$ be a functional shift over F such that $\mathcal{B}(\mathcal{F})$ is r.e. Suppose that a Turing machine M to recognize $\mathcal{B}(\mathcal{F})$ satisfies the following conditions:

- $T_{\mathrm{num}(x)}$ does not halt on input x before time t iff M accepts $f^t g^{\mathrm{num}(x)} f$,
- $T_{\mathrm{num}(x)}$ halts on input x at time t iff M accepts $hf^t g^{\mathrm{num}(x)} f$,

where $x \in \{01^n\delta \mid n \in \mathbb{N}\}$. Since it is clear that $\mathrm{num}(x)$ and the emulation of $T_{\mathrm{num}(x)}$ are in fact computable functions, M exists, by Church's thesis.

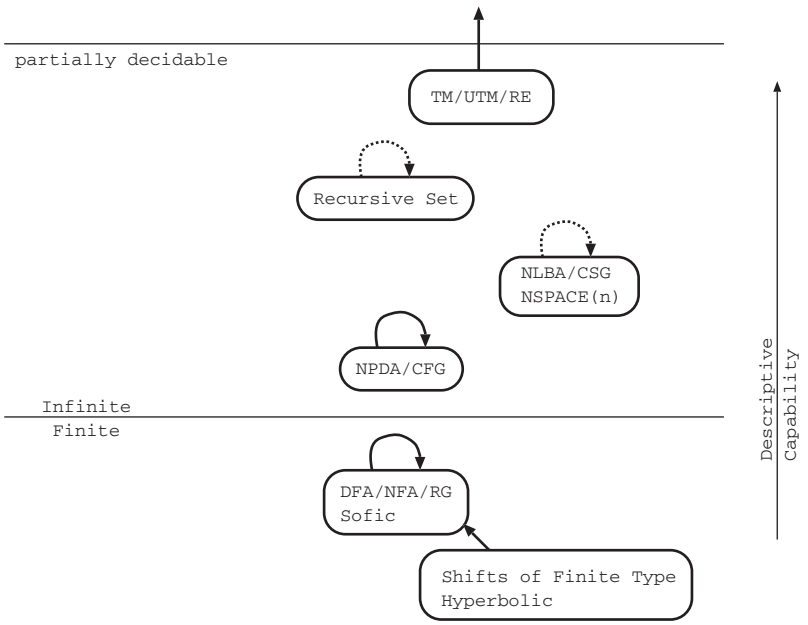

Figure 2. The computational hierarchy of the languages of functional shifts and generated shifts. Every class of the languages of functional shifts is the same as or smaller than that of generated shifts. For functional shifts with r.e. language, we have generated shifts whose language is beyond r.e. The solid arrow from A to B denotes that A is a class of the languages of functional shifts and B is that of shift spaces generated by the functional shifts. The broken arrow from A to B denotes that A is the same as the case of the solid arrow and B is a class of sets of forbidden blocks which can describe generated shifts given by the functional shifts. D = deterministic, N = nondeterministic, U = universal, G = grammar, A = automata, TM = Turing machine, RE = recursively enumerable, LBA = linear bounded A, PDA = pushdown A, FA = Finite A, CSG = context sensitive G, CFG = context free G, RG = regular G.

For any $x \in \{01^n\delta \mid n \in \mathbb{N}\}$, $\mathcal{D}_{\mathcal{F}}(t, x) \neq \emptyset$ iff $T_{\mathrm{num}(x)}$ does not halt on input x before time $t + 1$. Accordingly,

$$\begin{aligned} T_{\mathrm{num}(x)} \text{ never halts on input } x &\Leftrightarrow \lim_{t\to\infty} \mathcal{D}_{\mathcal{F}}(t, x) \neq \emptyset \\ &\Leftrightarrow x \in \mathcal{B}(X_{\mathcal{F}}), \end{aligned} \tag{38}$$

by lemma 4.3. Hence, the predicate $x \in \mathcal{B}(X_{\mathcal{F}})$ is not partially decidable, so that $\mathcal{B}(X_{\mathcal{F}})$ is not r.e. ■

5. Discussion

To study the relationship between dynamics and meta-dynamics, we have compared functional shifts with generated shifts, in both dynamical systems and computational terms. In section 3, we have proved that the entropy of a functional shift gives the upper limit for that of a generated shift given by the functional shift. In other words, every functional shift is at least as complex as the generated shift from the standpoint of dynamics. In section 4, considering the language of a shift space as a formal language, we have shown that there is a case in which the language of a functional shift is simpler than that of a shift space generated by the functional shift. Figure 2 shows the summary of results, in which the languages of functional shifts and generated shifts are located in the Chomsky hierarchy. It shows clearly that the class of the languages of

functional shifts is the same as or smaller than that of the generated shifts given by them. From those results, we may consider that the complexity of dynamics does not correspond to that of computation under the generative operation introduced here. Moreover, the viewpoints of both dynamics and computation give us opposite results concerning the complexity of systems. This means that an analysis of the complexity of systems surely depends on how we select a measure of complexity. Thus, it is important to study dynamical systems from several viewpoints, for example, those of both dynamics and computation, when we analyse the complexity of the systems.

It is interesting that the class of languages of functional shifts is equal to that of generated shifts, in the case in which the languages of the functional shifts are regular or context free, while there is no equivalence in the r.e. cases. The cause is conjectured to be that the equivalence depends on whether there exists a natural number n such that $\mathcal{D}_{\mathcal{F}}(n, x) \neq \emptyset$ iff $x \in \mathcal{B}(X_{\mathcal{F}})$ for any $x \in \mathcal{A}^*$. Lemma 4.6 and theorem 4.7 confirm our presumption. Theorems 4.8 and 4.9 also support it, because if any functions are bijections then $\mathcal{D}_{\mathcal{F}}(0, x) \neq \emptyset$ iff $x \in \mathcal{B}(X_{\mathcal{F}})$.

Let us discuss the existence of systems in which some predicates of dynamical systems theory are not partially decidable. In theorem 4.12, we have proved that there is a functional shift $\mathcal{F}$ satisfying that the language of $\mathcal{F}$ is r.e. and that of $X_{\mathcal{F}}$ is not r.e. Note that the predicate $x \in \mathcal{B}(X)$ means that the subset $\{y \in \mathcal{A}^{\mathbb{Z}} \mid y_{[1,|x|]} = x\}$ of $\mathcal{A}^{\mathbb{Z}}$ contains part of an invariant set X. Many problems involving dynamical systems can be resolved into this predicate. For example, we can consider such problems as follows:

(1) Given open sets $Y, Z \subset X$, is there a point $z \in Y$ that falls into Z under the shift map on X?
(2) Is the shift map on X topologically transitive?

Problem (1) is considered to be a prediction problem of orbits in a dynamical system. The reason why problem (1) resolves itself into the predicate $x \in \mathcal{B}(X)$ is that if $u, v \in \mathcal{B}(X)$, $Y = \{y \in X \mid y_{[1,|u|]} = u\}$ and $Z = \{y \in X \mid y_{[1,|v|]} = v\}$, then there is a block w such that $uwv \in \mathcal{B}(X)$ iff (1) is true. Since problem (2) can be reduced to (1), problem (2) contains the predicate as a subproblem. Thus, a shift space $X_{\mathcal{F}}$ is so complex that those predicates are not necessarily partially decidable, even if such predicates of $\mathcal{F}$ are partially decidable. As a dynamical system in which those problems are not partially decidable, we may consider the system with riddled basins [20, 21]. In fact, probably no algorithm exists which is able to assess problem (1) in a finite number of steps if Z is a riddled basin [19, 25].

Generally, the automaton to recognize a set not to be r.e. is called a super-Turing machine [22, 23]. Every super-Turing machine recognizes a set to be beyond r.e. by using infiniteness, for example, the property of the real number, which usual Turing machines do not have. Since the languages of shift spaces generated by functional shifts with r.e. languages are not r.e., those shift spaces have relevance to super-Turing machines. Some classes of sets to be beyond r.e. have been discussed in the field of analogue computation [11–15, 24]. Hamkins and Lewis [24], for instance, analyse classes of computations with infinitely many steps, and investigate computability and decidability on the reals. Note that shift spaces are usually continuums, because every shift space is defined as a collection of bi-infinite sequences. Hence, constructing a generated shift from a functional shift is an operation on a continuum, including tasks such as infinite mapping $x_{i+1} = f_i(x_i)$ for all $i \in \mathbb{Z}$. This operation, which is implicitly involved in the definition of functional shifts and generated shifts, often affects properties of the shifts themselves. For example, by using such an operation, we show that the predicate $x \in \mathcal{B}(X_{\mathcal{F}})$ is not partially decidable in the proof of theorem 4.12. Therefore, our framework could be related to super-Turing machines and analogue computation. Further analysis from these viewpoints is a future research topic.

We have introduced the framework of functional shifts as a model of dynamic change of functions. Since we can consider that a bi-infinite sequence of function $(f_i)_{i\in\mathbb{Z}}$ denotes the evolution of maps, functional shifts represent the dynamics of functions. Generated shifts also represent the dynamics determined by $x_{i+1} = f_i(x_i)$. Considering the shift map on $X_{\mathcal{F}}$ as dynamics, we can regard that on $\mathcal{F}$ as meta-dynamics. Let us focus on the operation to generate shift spaces from functional shifts, i.e. to construct dynamics from meta-dynamics. By theorem 4.12, such an operation includes a task to generate sets not to be r.e. in spite of the fact that any functions in F are computable. Thus, as we discussed, when we study dynamical systems modified by meta-dynamics, the existence of complex dynamics should be taken into account.

Finally, we consider self-modifying systems, in which rules governing a system are used to change the rules themselves. We have difficulty in achieving a direct representation of the dynamics of a self-modifying type. The cause of the difficulty is that functions and states cannot be perfectly separated in self-modifying systems. Research into the function dynamics of the self-modifying type has recently become a subject of special interest in the study of these systems. Functional dynamics is an example of a self-modifying system, because the change in function f is determined by a self-reference term $f \circ f$ [4–6]. As another example, objects in algorithmic chemistry encode functions which change the objects themselves [7,8]. To describe self-modifying systems, we must take the self-referential nature of a dynamical system into account. We can represent this characteristic using functional shifts as follows [26]. Let us consider that a conjugacy from a functional shift to a generated shift given by the functional shift is a 'self-reference code' between functions and states. Considering that each sequence of functions is equal to a sequence of symbols corresponding to itself under the code, we can regard the functional shift as self-modifying. For instance, the functional shift $\mathcal{F}$ given in examples 2.3 or 2.5 has conjugacy to $X_{\mathcal{F}}$, so that the functional shifts in these examples are regarded as self-modifying systems. However, the relationship between the systems described by functional shifts and other systems introduced by [4–8] is not clear.

6. Conclusion

We have investigated shift dynamics called functional shifts within both dynamics and computational frameworks. From the dynamical viewpoint, we have proved that the entropy of a functional shift is not less than that of a shift space generated by the functional shift. This means that functional shifts generate less complex shift spaces than themselves. On the other hand, we have compared functional shifts with generated shifts in terms of the Chomsky hierarchy (see figure 2). We have proved that any class of the languages of shift spaces is at least as large as that of the functional shifts that generate the shift spaces. Furthermore, we have shown that there is a class of the languages of functional shifts, which is strictly smaller than that of generated shifts given by the functional shifts. From those results, we have argued that the viewpoints of dynamics and computation give us opposite results about the complexity of systems.

We have shown a new class of shift spaces, generated shifts whose languages are not r.e. if the languages of functional shifts to give the generated shifts are r.e. The shift map over some shift spaces in the class has very unpredictable dynamics. This new class gives us a method of studying dynamical systems from the viewpoint of analogue computation.

Acknowledgments

The authors wish to thank Yuzuru Sato, Ichiro Tsuda, Hiroakira Ono, and Naoto Kataoka for helpful discussions and comments. We are grateful to the anonymous reviewers for

their critical reading of the manuscript and significant comments. We would like to thank Judith Anne Steeh for her assistance in English editing. This work is partly supported by the Research Fellowship Program of Canon Foundation in Europe, and a Grant-in-Aid for Scientific Research (No 12780269) from the Ministry of Education, Culture, Sports, Science and Technology of Japan and by the Japan Society for the Promotion of Science.

Appendix. Review of the Chomsky hierarchy

The Chomsky hierarchy is a containment hierarchy of classes of formal grammars that generate formal languages. This hierarchy was described by Noam Chomsky [27].

A *formal grammar* (or *type-0 grammar*) is a 4-tuple (V_N, V_T, P, S), where

- V_N is a finite set of *nonterminal symbols*;
- V_T is a finite set of *terminal symbols* such as $V_N \cap V_T = \emptyset$, and $V = V_N \cup V_T$ is called the set of *grammar symbols*;
- P is a finite collection of *productions* which are of the form $\alpha \to \beta$ with $\alpha \in V^+$ and $\beta \in V^*$,
- $S \in V_N$ is a designated symbol called the *start symbol*.

Given a formal grammar $G = (V_N, V_T, P, S)$, the *derivation relation* $\underset{G}{\Longrightarrow} \subset V^* \times V^*$ is defined by

$$\gamma\alpha\delta \underset{G}{\Longrightarrow} \gamma\beta\delta \qquad \text{iff } \alpha \to \beta \in P, \tag{A1}$$

where $\alpha, \beta, \gamma, \delta \in V^*$. The transitive closure of $\underset{G}{\Longrightarrow}$ is denoted by $\overset{+}{\underset{G}{\Longrightarrow}}$, and the reflexive and transitive closure of $\underset{G}{\Longrightarrow}$ is denoted by $\overset{*}{\underset{G}{\Longrightarrow}}$. The language generated by G is the set

$$L(G) = \{w \in V_T^* \mid S \overset{*}{\underset{G}{\Longrightarrow}} w\}. \tag{A2}$$

A language $L \subset V_T^*$ is a *formal language* (or *type-0 language*) iff $L = L(G)$ for some formal grammar G.

The Chomsky hierarchy consists of classes of regular grammars, context free grammars, context sensitive grammars, and type-0 grammars. Regular, context free, and context sensitive grammars are more restrictive than formal grammars.

- A regular grammar (type-3 grammar) is a formal grammar $G = (V_N, V_T, P, S)$, such that the productions are of the form $\alpha \to \beta$ with $\alpha \in V_N$ and $\beta \in V_T \cup V_T \times V_N$. A language $L \subset V_T^*$ is a regular language iff $L = L(G)$ for some regular grammar G.
- A context free grammar (type-2 grammar) is a formal grammar $G = (V_N, V_T, P, S)$, such that the productions are of the form $\alpha \to \beta$ with $\alpha \in V_N$ and $\beta \in V^+$. A language $L \subset V_T^*$ is a context free language iff $L = L(G)$ for some context free grammar G.
- A context sensitive grammar (type-1 grammar) is a formal grammar $G = (V_N, V_T, P, S)$, such that the productions are of the form $\alpha \to \beta$ satisfying $|\alpha| \leqslant |\beta|$. A language $L \subset V_T^*$ is a context sensitive language iff $L = L(G)$ for some context sensitive grammar G.

From the above definition, every regular grammar is context free, every context free grammar is context sensitive and every context sensitive grammar is type-0. Moreover, these are all proper inclusions.

It is known that there exists an automaton corresponding to each formal language belonging to the Chomsky hierarchy. Every type-0 language can be recognized by a Turing machine, where a Turing machine is a finite state machine moving left and right on a tape, on which a string of symbols in some finite alphabet is written. The language recognized by a Turing

Table 1. Summarize each of the Chomsky's four types of grammars, languages, and automata.

Grammar	Language	Automaton
Type-0	Recursively enumerable	Turing machine
Type-1	Context sensitive	LBA
Type-2	Context free	NPDA
Type-3	Regular	Finite state automaton

machine is defined as all the strings on which it halts. These languages are also known as the recursively enumerable (r.e.) languages. Every context sensitive language can be recognized by a LBA which is a nondeterministic Turing machine whose tape is bounded by the length of the input string. Every context free language can be recognized by a NPDA, which is a finite state automaton having a stack. Every regular language can be recognized by a finite state automaton, which has no memory. Thus, the complexity from the Chomsky hierarchy is based on the memory size of automata to recognize languages. Table 1 summarizes each of Chomsky's four types of grammars, the class of languages each grammar generates, and the type of automaton that recognizes each language.

References

[1] Tokita K and Yasutomi A 1999 Mass extinction in a dynamical system of evolution with variable dimension *Phys. Rev.* E **60** 842–7

[2] Tokita K and Yasutomi A 2003 Emergence of a complex and stable network in a model ecosystem with extinction and mutation *Theor. Population Biol.* **63** 131–46

[3] Sato Y and Ikegami T 2000 Nonlinear computation with switching map systems *J. Universal Comput. Sci.* **6** 881–905

[4] Kataoka N and Kaneko K 2000 Functional dynamics I: articulation process *Physica* D **138** 225–50

[5] Kataoka N and Kaneko K 2001 Functional dynamics II: syntactic structure *Physica* D **149** 174–96

[6] Kataoka N and Kaneko K 2003 Dynamical networks in function dynamics *Physica* D **181** 235–51

[7] Fontana W 1992 Algorithmic chemistry *Artificial Life II* SFI Studied in the Sciences of Complexity **X** 159–209

[8] Fontana W and Buss L W 1994 The arrival of the fittest: toward a theory of biological organization *Bull. Math. Biol.* **56** 1–64

[9] Tsuda I 1994 Can stochastic renewal of maps be a model for cerebral cortex? *Physica* D **75** 165–78

[10] Moore C 1991 Generalized shifts: unpredictability and undecidability in dynamical systems *Nonlinearity* **4** 199–230

[11] Siegelmann H T 1995 Computation beyond the Turing limit *Science* **268** 545–8

[12] Siegelmann H T and Fishman S 1998 analog computation with dynamical systems *Physica* D **120** 214–35

[13] Siegelmann H T and Ben-Hur A 1999 Computational complexity for continuous time dynamics *Phys. Rev. Lett.* **83** 1463–6

[14] Ben-Hur A, Siegelmann H T and Fishman S 2002 A theory of complexity for continuous time systems *J. Complexity* **18** 51–86

[15] Ben-Hur A, Feinberg J, Fishman S and Siegelmann H T 2003 Probabilistic analysis of a differential equation for linear programming *J. Complexity* **19** 474–510

[16] Crutchfield J P and Young K 1990 Computation at the onset of chaos *Complexity, Entropy and the Physics of Information* (Reading, MA: Addison Wesley) pp 223–69

[17] Lakdawala P 1996 Computational complexity of symbolic dynamics at the onset of chaos *Phys. Rev.* E **53** 4477–85

[18] Lind D and Marcus B 1995 *An Introduction to Symbolic Dynamics and Coding* (Cambridge: Cambridge University Press)

[19] Badii R and Politi A 1997 *Complexity: Hierarchical Structures and Scaling in Physics* (Cambridge: Cambridge University Press)

[20] Alexander J C, Yorke J A, You Z and Kan I 1992 Riddled Basins *Int. J. Bifurc. Chaos* **2** 795–813

[21] Ott E 2002 *Chaos in Dynamical Systems* (Cambridge: Cambridge University Press)

[22] Stannett M 1990 X-machines and the halting problem: building a super-Turing machine *Formal Aspects Comput.* **2** 331–41
[23] Copeland B J 1998 Super Turing-machines *Complexity* **4** 30–2
[24] Hamkins J D and Lewis A 2000 Infinite time Turing machines *J. Symbolic Logic* **65** 567–604
[25] Blum L and Smale S 1993 The Gödel incompleteness theorem and decidability over a ring *From Topology to Computation* ed M W Hirsch *et al* (New York: Springer) pp 321–39
[26] Namikawa J and Hashimoto T 2002 Functional shifts: hierarchy and self-modification of rules in dynamics *Discrete Dynamics in Nature and Society* at press
[27] Chomsky N 1959 On certain formal properties of grammars *Inform. Control* **2** 393–5