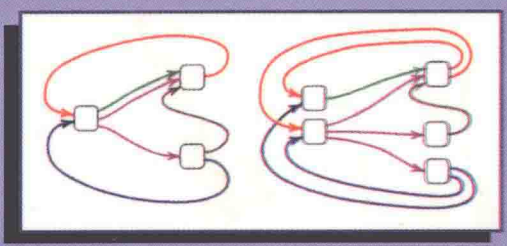
AN INTRODUCTION TO

Symbolic Dynamics and Coding



DOUGLAS LIND

and

BRIAN MARCUS

AN INTRODUCTION TO SYMBOLIC DYNAMICS AND CODING

Douglas Lind
University of Washington

Brian Marcus
IBM Almaden Research Center

CAMBRIDGE
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK http: //www.cup.cam.ac.uk
40 West 20th Street, New York, NY 10011-4211, USA http: //www.cup.org
10 Stamford Road, Oakleigh, Melbourne 3166, Australia

First published 1995
Reprinted 1999 (with corrections)

Lind, Douglas A.
An introduction to symbolic dynamics and coding / Douglas
Lind, Brian Marcus
p. cm.
1. Differential dynamical systems. 2. Coding theory.
I. Marcus, Brian, 1949– . II. Title.
QA614.8.L56 1995 95-10570
003'.85'01154 – dc20 CIP

A catalogue record for this book is available from the British Library

ISBN 0 521 55124 2 hardback
ISBN 0 521 55900 6 paperback

Transferred to digital printing 2003

Dedicated to the memory of Rufus Bowen,
our friend and teacher
1947–1978

PREFACE

Symbolic dynamics is a rapidly growing part of dynamical systems. Although it originated as a method to study general dynamical systems, the techniques and ideas have found significant applications in data storage and transmission as well as linear algebra. This is the first general textbook on symbolic dynamics and its applications to coding, and we hope that it will stimulate both engineers and mathematicians to learn and appreciate the subject.

Dynamical systems originally arose in the study of systems of differential equations used to model physical phenomena. The motions of the planets, or of mechanical systems, or of molecules in a gas can be modeled by such systems. One simplification in this study is to discretize time, so that the state of the system is observed only at discrete ticks of a clock, like a motion picture. This leads to the study of the iterates of a single transformation. One is interested in both quantitative behavior, such as the average time spent in a certain region, and also qualitative behavior, such as whether a state eventually becomes periodic or tends to infinity. Symbolic dynamics arose as an attempt to study such systems by means of discretizing space as well as time. The basic idea is to divide up the set of possible states into a finite number of pieces, and keep track of which piece the state of the system lies in at every tick of the clock. Each piece is associated with a "symbol," and in this way the evolution of the system is described by an infinite sequence of symbols. This leads to a "symbolic" dynamical system that mirrors and helps us to understand the dynamical behavior of the original system.

In their fundamental paper [MorH1] written over fifty years ago, Morse and Hedlund named the subject of symbolic dynamics and described its philosophy as follows.

> *The methods used in the study of recurrence and transitivity frequently combine classical differential analysis with a more abstract symbolic analysis. This involves a characterization of the ordinary dynamical trajectory by an unending sequence of symbols termed a symbolic trajectory such that the properties of recurrence and transitivity of the dynamical trajectory are reflected in analogous properties of its symbolic trajectory.*

One example of this idea that you are very familiar with is the decimal expansion of real numbers in the unit interval $[0,1)$. Here the transformation is given by multiplying a number $x \in [0,1)$ by 10 and keeping its fractional part $\{10x\}$. We partition $[0,1)$ into ten equal subintervals $[0,1/10)$, $[1/10,2/10)$, $\ldots$, $[9/10,1)$, and we use the "symbol" j for the interval $[j/10,(j+1)/10)$. Let x be a number in $[0,1)$. Using the transformation $x \mapsto \{10x\}$ and this partition, we write down an infinite sequence

of symbols corresponding to any given $x = .x_1x_2\ldots$, as follows. Since $x \in [x_1/10, (x_1+1)/10)$, the first symbol is just the first digit x_1 of its decimal expansion. Since $\{10x\} \in [x_2/10, (x_2+1)/10)$, the next symbol is the first digit of the decimal expansion of $\{10x\}$, which is simply the second digit x_2 of the decimal expansion of x. Continuing in this way, we see that the symbolic sequence derived from x is none other than the sequence of digits from its decimal expansion.

Symbolic dynamics began when Jacques Hadamard [Had] applied this idea in 1898 to more complicated systems called geodesic flows on surfaces of negative curvature. The main point of his work is that there is a simple description of the possible sequences that can arise this way. He showed that there is a finite set of forbidden pairs of symbols, and that the possible sequences are exactly those that do not contain any forbidden pair. This is an example of one of the fundamental objects of study in symbolic dynamics called a shift of finite type. Later discoveries of Morse, Hedlund, and others in the 1920's, 1930's, and 1940's showed that in many circumstances such a finite description of the dynamics is possible. These ideas led in the 1960's and 1970's to the development of powerful mathematical tools to investigate a class of extremely interesting mappings called hyperbolic diffeomorphisms. We will see in §6.5 some examples of this.

Another source of inspiration and questions in symbolic dynamics comes from data storage and transmission. As we discuss in §2.5, when information is stored as bits on a magnetic or optical disk, there are physical reasons why it is not stored verbatim. For example, the bits on the surface of a compact audio disk are written in a long sequence obeying the constraint that between successive 1's there are at least two 0's but no more than ten 0's. How can one efficiently transform arbitrary data (such as a computer program or a Beethoven symphony) into sequences that satisfy such kinds of constraints? What are the theoretical limits of this kind of transformation? We are again confronted with a space of sequences having a finite description, and we are asking questions about such spaces and ways to encode and decode data from one space (the space of arbitrary sequences) to another (the space of constrained sequences). One of the main results in this book, the Finite-State Coding Theorem of Chapter 5, tells us when such codes are possible, and gives us an algorithm for finding them. This has led to new codes and a deeper understanding of code constructions. In particular, it has yielded a new and useful technique for code construction called the state-splitting algorithm.

While symbolic dynamics has drawn heavily on other mathematical disciplines such as linear algebra for its tools, it has also contributed new tools for other areas. For example, some deep work of Boyle and Handelman in symbolic dynamics described in Chapter 11 has led to the complete solution of the problem of characterizing the possible sets of nonzero eigen-

values of real nonnegative matrices. This solved a variant of a problem which has been perplexing linear algebraists for decades.

This book is intended to serve as an introduction to symbolic dynamics, its basic ideas, techniques, problems, and spirit. We will focus on symbolic dynamical systems that use just a finite number of symbols, on sequences of such symbols that are infinite in both directions (unlike the decimal expansion above), and spaces of sequences that have finite memory properties. Our aim is to study coding problems for such systems. In particular, we concentrate on the following:

- Find complete sets of necessary and sufficient conditions for the existence of various types of codes from one symbolic dynamical system to another.
- Completely determine the values of certain properties, invariant under a natural notion of "conjugacy," such as entropy, numbers of periodic sequences, and so on.
- Explore interactions with information and coding theory, linear algebra, and general dynamical systems.

On the other hand, there are many important topics within symbolic dynamics that we do not treat in any depth in this book. Many of these topics are briefly discussed in Chapter 13.

The book is organized as follows. We start in Chapter 1 with the basic notions in symbolic dynamical systems, such as full shifts, shift spaces, irreducibility, and sliding block codes. In Chapter 2 we focus on a special class of shift spaces called shifts of finite type, and in Chapter 3 we study a generalization of these called sofic shifts. Entropy, an important numerical invariant, is treated in Chapter 4, which also includes an introduction to the fundamental Perron–Frobenius theory of nonnegative matrices. Building on the material in the first four chapters, we develop in Chapter 5 the state-splitting algorithm for code construction and prove the Finite-State Coding Theorem. Taken together, the first five chapters form a self-contained introduction to symbolic dynamics and its applications to practical coding problems.

Starting with Chapter 6 we switch gears. This chapter provides some background for the general theory of dynamical systems and the place occupied by symbolic dynamics. We show how certain combinatorial ideas like shift spaces and sliding block codes studied in earlier chapters are natural expressions of fundamental mathematical notions such as compactness and continuity. There is a natural notion of two symbolic systems being "the same," or conjugate, and Chapter 7 deals with the question of when two shifts of finite type are conjugate. This is followed in Chapters 8 and 9 with the classification of shifts of finite type and sofic shifts using weaker notions, namely finite equivalence and almost conjugacy. Chapters 7, 8, and 9 treat problems of coding between systems with equal entropy. In

Chapter 10 we treat the case of unequal entropy and, in particular, determine when one shift of finite type can be embedded in another. The set of numbers which can occur as entropies of shifts of finite type is determined in Chapter 11. Also in that chapter we draw on the results of Chapter 10 to prove a partial result towards characterizing the zeta functions of shifts of finite type (the zeta function is a function which determines the numbers of periodic sequences of all periods). Some of these results in turn are used in Chapter 12 to classify shifts of finite type and sofic shifts up to a notion that sits in between conjugacy and almost conjugacy. The main result of Chapter 12 may be regarded as a generalization of the Finite-State Coding Theorem, and tells us when one sofic shift can be encoded into another in a natural way. Chapter 13 contains a survey of more advanced results and recent literature on the subject. This is a starting point for students wishing to become involved in areas of current research.

The mathematical prerequisites for this book are relatively modest. A reader who has mastered undergraduate courses in linear algebra, abstract algebra, and calculus should be well prepared. This includes upper-division undergraduate mathematics majors, mathematics graduate students, and graduate students studying information and storage systems in computer science and electrical engineering. We have tried to start from scratch with basic material that requires little background. Thus in the first five chapters no more than basic linear algebra and one-variable calculus is needed: matrices, linear transformations, similarity, eigenvalues, eigenvectors, and notions of convergence and continuity (there are also a few references to basic abstract algebra, but these are not crucial). Starting with Chapter 6, the required level of mathematical sophistication increases. We hope that the foundation laid in the first five chapters will inspire students to continue reading. Here the reader should know something more of linear algebra, including the Jordan canonical form, as well as some abstract algebra including groups, rings, fields, and ideals, and some basic notions from the topology of Euclidean space such as open set, closed set, and compact set, which are quickly reviewed.

The book contains over 500 exercises, ranging from simple checks of understanding to much more difficult problems and projects. Those that seem harder have been marked with a *. There is a detailed index as well as a notation index.

Theorems, propositions, lemmas, and other items are numbered consecutively within sections using three parts: $k.m.n$ refers to item n in section m of Chapter k. For instance, Proposition 5.2.3 refers to the third item in Section 2 of Chapter 5. Equations are numbered separately with the same three-part system, but with a different style, so that (3-2-4) refers to the fourth equation in Section 2 of Chapter 3.

We use the standard notations of $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ for the sets of integers, rationals, reals, and complexes, and use $\mathbb{Z}^+$, $\mathbb{Q}^+$, and $\mathbb{R}^+$ to denote the

nonnegative elements of the corresponding sets. All matrices are assumed to be square unless stated (or context demands) otherwise.

The authors have taught courses based on preliminary versions of this material at the University of Washington, Stanford University and the University of California at Berkeley. They have also given lecture series at the Mathematical Sciences Research Institute in Berkeley and the IBM Almaden Research Center. Based on these experiences, here are some suggestions on how to organize the material in this book into courses of different lengths. Chapters 1–5 would constitute a solid one-quarter course introducing symbolic dynamics from scratch and concluding with applications to coding. Chapters 1–7 could form a one-semester course that also includes more theoretical material and finishes with a detailed discussion of the conjugacy problem. The entire book can be covered in a one-year course that would bring students to the frontiers of current research. Of course, the material can be covered more quickly for students with more than the minimum required background.

This book has benefited enormously from suggestions, comments, and corrections made by many people (students, faculty and industrial researchers) on preliminary versions of this manuscript, in lectures on these versions, and in answers to our queries. These include Paul Algoet. Jonathan Ashley, Leslie Badoian, Mignon Belongie, Mike Boyle, Karen Brucks, Elise Cawley, Phil Chou, Wu Chou, Ethan Coven, Jim Fitzpatrick, Dave Forney, Lisa Goldberg, Michael Gormish, Michael Hollander, Danrun Huang, Natasha Jonoska, Sampath Kannan, Bruce Kitchens, Wolfgang Krieger, David Larsen, Nelson Markley, Lee Neuwirth, Kyewon Koh Park, Karl Petersen, Ronny Roth, Paul Siegel, Sylvia Silberger, N. T. Sindhushayana, Serge Troubetzkoy, Paul Trow, Selim Tuncel, Jeff Von Limbach, Jack Wagoner, Peter Walters, Barak Weiss, Susan Williams, and Amy Wilkinson. We are happy to thank Roy Adler, who first suggested the possibility of a textbook on symbolic dynamics, especially one that requires only a minimum amount of mathematical background. Adler's 1984 set of unpublished lecture notes from Brown University and his notes on Markov partitions were valuable sources. We are also grateful to Elza Erkip, Amos Lapidoth, and Erik Ordentlich, who earned their just desserts by working so many of the exercises in the first five chapters. Special thanks are due to Zhe-xian Wan for very detailed and helpful comments on our manuscript and to Brian Hopkins for clarifying a multitude of murky passages and for eliminating an uncountable number of needless commas. This book is typeset using the LaMS-TeX extensions to TeX, and we thank Michael Spivak for his help with this. Finally, we wish to express our gratitude to the National Science Foundation (grants DMS-9004252 and DMS-9303240), the University of Washington, and the IBM Corporation for their support of this work.

Current information regarding this book, including selected corrections and suggestions from readers, may be found using the World Wide Web at

the address

`http://www.math.washington.edu/SymbolicDynamics`

This address can also be used to submit items for inclusion.

The first author is profoundly grateful to John Whittier Treat for many things, both spoken and unspoken, over many years.

The second author would like to thank Yvonne, Nathaniel, Mom and Dad for a life filled with love and support, without which this book could not have been written.

Notes on the Second Printing

The Second Printing has provided us the opportunity to fix some minor errors. In addition, we have made a few changes to reflect the recent solution by Kim and Roush [KimR12] of the Shift Equivalence Problem described in §7.3.

A complete list of changes made for the Second Printing is available at the Web site displayed above. We will continue to list further corrections and clarifications at this Web site.

CONTENTS

CHAPTER 1

SHIFT SPACES

Shift spaces are to symbolic dynamics what shapes like polygons and curves are to geometry. We begin by introducing these spaces, and describing a variety of examples to guide the reader's intuition. Later chapters will concentrate on special classes of shift spaces, much as geometry concentrates on triangles and circles. As the name might suggest, on each shift space there is a shift map from the space to itself. Together these form a "shift dynamical system." Our main focus will be on such dynamical systems, their interactions, and their applications.

In addition to discussing shift spaces, this chapter also connects them with formal languages, gives several methods to construct new shift spaces from old, and introduces a type of mapping from one shift space to another called a sliding block code. In the last section, we introduce a special class of shift spaces and sliding block codes which are of interest in coding theory.

§1.1. Full Shifts

Information is often represented as a sequence of discrete symbols drawn from a fixed finite set. This book, for example, is really a very long sequence of letters, punctuation, and other symbols from the typographer's usual stock. A real number is described by the infinite sequence of symbols in its decimal expansion. Computers store data as sequences of 0's and 1's. Compact audio disks use blocks of 0's and 1's, representing signal samples, to digitally record Beethoven symphonies.

In each of these examples, there is a finite set $\mathcal{A}$ of *symbols* which we will call the *alphabet*. Elements of $\mathcal{A}$ are also called *letters*, and they will typically be denoted by a, b, c, ..., or sometimes by digits like 0, 1, 2, ..., when this is more meaningful. Decimal expansions, for example, use the alphabet $\mathcal{A} = \{0, 1, \ldots, 9\}$.

Although in real life sequences of symbols are finite, it is often extremely useful to treat long sequences as infinite in both directions (or *bi-infinite*).

This is analogous to using real numbers, continuity, and other ideas from analysis to describe physical quantities which, in reality, can be measured only with finite accuracy.

Our principal objects of study will therefore be collections of bi-infinite sequences of symbols from a finite alphabet $\mathcal{A}$. Such a sequence is denoted by $x = (x_i)_{i \in \mathbb{Z}}$, or by

$$x = \ldots x_{-2}x_{-1}x_0x_1x_2\ldots,$$

where each $x_i \in \mathcal{A}$. The symbol x_i is the ith *coordinate* of x, and x can be thought of as being given by its coordinates, or as a sort of infinite "vector." When writing a specific sequence, you need to specify which is the 0th coordinate. This is conveniently done with a "decimal point" to separate the x_i with $i \geqslant 0$ from those with $i < 0$. For example,

$$x = \ldots 010.1101\ldots$$

means that $x_{-3} = 0$, $x_{-2} = 1$, $x_{-1} = 0$, $x_0 = 1$, $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, and so on.

Definition 1.1.1. If $\mathcal{A}$ is a finite alphabet, then the *full $\mathcal{A}$-shift* is the collection of all bi-infinite sequences of symbols from $\mathcal{A}$. The *full r-shift* (or simply *r-shift*) is the full shift over the alphabet $\{0, 1, \ldots, r-1\}$.

The full $\mathcal{A}$-shift is denoted by

$$\mathcal{A}^{\mathbb{Z}} = \{x = (x_i)_{i \in \mathbb{Z}} : x_i \in \mathcal{A} \text{ for all } i \in \mathbb{Z}\}.$$

Here $\mathcal{A}^{\mathbb{Z}}$ is the standard mathematical notation for the set of all functions from $\mathbb{Z}$ to $\mathcal{A}$, and such functions are just the bi-infinite sequences of elements from $\mathcal{A}$. Each sequence $x \in \mathcal{A}^{\mathbb{Z}}$ is called a *point* of the full shift. Points from the full 2-shift are also called *binary sequences*. If $\mathcal{A}$ has size $|\mathcal{A}| = r$, then there is a natural correspondence between the full $\mathcal{A}$-shift and the full r-shift, and sometimes the distinction between them is blurred. For example, it can be convenient to refer to the full shift on $\{+1, -1\}$ as the full 2-shift.

Blocks of consecutive symbols will play a central role. A *block* (or *word*) over $\mathcal{A}$ is a finite sequence of symbols from $\mathcal{A}$. We will write blocks without separating their symbols by commas or other punctuation, so that a typical block over $\mathcal{A} = \{a, b\}$ looks like *aababbabbb*. It is convenient to include the sequence of *no* symbols, called the *empty block* (or *empty word*) and denoted by ε. The *length* of a block u is the number of symbols it contains, and is denoted by $|u|$. Thus if $u = a_1a_2\ldots a_k$ is a nonempty block, then $|u| = k$, while $|\varepsilon| = 0$. A *k-block* is simply a block of length k. The set of all k-blocks over $\mathcal{A}$ is denoted $\mathcal{A}^k$. A *subblock* or *subword* of $u = a_1a_2\ldots a_k$ is a block

of the form $a_i a_{i+1} \dots a_j$, where $1 \leqslant i \leqslant j \leqslant k$. By convention, the empty block ε is a subblock of every block.

If x is a point in $\mathcal{A}^{\mathbb{Z}}$ and $i \leqslant j$, then we will denote the block of coordinates in x from position i to position j by

$$x_{[i,j]} = x_i x_{i+1} \dots x_j .$$

If $i > j$, define $x_{[i,j]}$ to be ε. It is also convenient to define

$$x_{[i,j)} = x_i x_{i+1} \dots x_{j-1} .$$

By extension, we will use the notation $x_{[i,\infty)}$ for the *right-infinite sequence* $x_i x_{i+1} x_{i+2} \dots$, although this is not really a block since it has infinite length. Similarly, $x_{(-\infty,i]} = \dots x_{i-2} x_{i-1} x_i$. The *central* $(2k+1)$*-block of* x is $x_{[-k,k]} = x_{-k} x_{-k+1} \dots x_k$. We sometimes will write $x_{[i]}$ for x_i, especially when we want to emphasize the index i.

Two blocks u and v can be put together, or *concatenated*, by writing u first and then v, forming a new block uv having length $|uv| = |u| + |v|$. Note that uv is in general not the same as vu, although they have the same length. By convention, $\varepsilon u = u\varepsilon = u$ for all blocks u. If $n \geqslant 1$, then u^n denotes the concatenation of n copies of u, and we put $u^0 = \varepsilon$. The law of exponents $u^m u^n = u^{m+n}$ then holds for all integers $m, n \geqslant 0$. The point $\dots uuu.uuu \dots$ is denoted by u^∞.

The index i in a point $x = (x_i)_{i \in \mathbb{Z}}$ can be thought of as indicating time, so that, for example, the time-0 coordinate of x is x_0. The passage of time corresponds to shifting the sequence one place to the left, and this gives a map or transformation from a full shift to itself.

Definition 1.1.2. The *shift map* σ on the full shift $\mathcal{A}^{\mathbb{Z}}$ maps a point x to the point $y = \sigma(x)$ whose ith coordinate is $y_i = x_{i+1}$.

The operation σ, pictured below, maps the full shift $\mathcal{A}^{\mathbb{Z}}$ onto itself. There

$$\begin{array}{lcl} x & = & \dots\ x_{-3}\ x_{-2}\ x_{-1}\ .\ x_0\ x_1\ x_2\ x_3\ \dots \\ \Big\downarrow \sigma & & \searrow\ \searrow\ \searrow\ \searrow\ \searrow\ \searrow \\ y = \sigma(x) & = & \dots\ x_{-2}\ x_{-1}\ x_0\ .\ x_1\ x_2\ x_3\ x_4\ \dots \end{array}$$

is also the inverse operation σ^{-1} of shifting one place to the right, so that σ is both one-to-one and onto. The composition of σ with itself $k > 0$ times $\sigma^k = \sigma \circ \dots \circ \sigma$ shifts sequences k places to the left, while $\sigma^{-k} = (\sigma^{-1})^k$ shifts the same amount to the right. This shifting operation is the reason $\mathcal{A}^{\mathbb{Z}}$ is called a full shift ("full" since all sequences of symbols are allowed).

The shift map is useful for expressing many of the concepts in symbolic dynamics. For example, one basic idea is that of codes, or rules, which

transform one sequence into another. For us, the most important codes are those that do not change with time. Consider the map $\phi: \{0,1\}^{\mathbb{Z}} \to \{0,1\}^{\mathbb{Z}}$ defined by the rule $\phi(x) = y$, where $y_i = x_i + x_{i+1} \pmod 2$. Then ϕ is a coding rule that replaces the symbol at index i with the sum modulo 2 of itself and its right neighbor. The coding operation ϕ acts the same at each coordinate, or is *stationary*, i.e., independent of time.

Another way to say this is that applying the rule ϕ and then shifting gives exactly the same result as shifting and then applying ϕ. Going through the following diagram to the right and then down gives the same result as going down and then to the right.

$$\begin{array}{ccc} x & \xrightarrow{\sigma} & \sigma(x) \\ \phi\downarrow & & \downarrow\phi \\ \phi(x) & \xrightarrow{\sigma} & \sigma(\phi(x)) = \phi(\sigma(x)) \end{array}$$

We can express this as $\sigma \circ \phi = \phi \circ \sigma$, or in terms of the coordinates by $\sigma(\phi(x))_{[i]} = \phi(\sigma(x))_{[i]}$, since both equal $x_{i+1} + x_{i+2} \pmod 2$. Recall that when two mappings f and g satisfy $f \circ g = g \circ f$, they are said to *commute*. Not all pairs of mappings commute (try: $f =$ "put on socks" and $g =$ "put on shoes"). Using this terminology, a code ϕ on the full 2-shift is stationary if it commutes with the shift map σ, which we can also express by saying that the following diagram commutes.

$$\begin{array}{ccc} \{0,1\}^{\mathbb{Z}} & \xrightarrow{\sigma} & \{0,1\}^{\mathbb{Z}} \\ \phi\downarrow & & \downarrow\phi \\ \{0,1\}^{\mathbb{Z}} & \xrightarrow{\sigma} & \{0,1\}^{\mathbb{Z}} \end{array}$$

We will discuss codes in more detail in §1.5.

Points in a full shift which return to themselves after a finite number of shifts are particularly simple to describe.

Definition 1.1.3. A point x is *periodic for* σ if $\sigma^n(x) = x$ for some $n \geq 1$, and we say that x has *period* n under σ. If x is periodic, the smallest positive integer n for which $\sigma^n(x) = x$ is the *least period* of x. If $\sigma(x) = x$, then x is called a *fixed point* for σ.

If x has least period k, then it has period $2k, 3k, \ldots$, and every period of x is a multiple of k (see Exercise 1.1.5). A fixed point for σ must have the form a^∞ for some symbol a, and a point of period n has the form u^∞ for some n-block u.

Iteration of the shift map provides the "dynamics" in symbolic dynamics (see Chapter 6). Naturally, the "symbolic" part refers to the symbols used to form sequences in the spaces we will study.