THE ALGEBRAIC STRUCTURE OF TRELLIS CODES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Mitchell D. Trott
August 1992
Abstract

Trellis codes, which are used in all modern high-speed computer modems, are usually described as a combination of a binary linear convolutional code and a mapping from coded output bits to signal points in a Euclidean signal space. This representation obscures the relationship between the algebraic structure of the binary code and the geometric structure of the signal space code.

In this dissertation, we present a representation of trellis codes as dynamical systems in which the output alphabet is a group of isometries (distance-preserving transformations) of Euclidean space. A system over a group of isometries is “converted” into a trellis code by applying each sequence of isometries in the code componentwise to an initial signal space sequence \( \mathbf{x} \); the set of all such sequences is a trellis code in an infinite-dimensional Euclidean space.

A code with such a representation is geometrically uniform, and therefore has a number of strong symmetry properties. Foremost among them is the uniform error property: when the code is used over an additive white Gaussian noise channel with maximum likelihood decoding, the probability of error is independent of the transmitted code sequence. This property greatly simplifies performance analysis.

Not all trellis codes admit such a representation; however, most good ones do. We show that the apparently nonlinear 8-state, two-dimensional Wei code used in V.32 modems has such a representation, as does the 16-state, four-dimensional Wei code used in higher-speed modems. These representations are novel and unexpected, and show that geometrical uniformity is a broader and subtler concept than was previously recognized.

We develop a structure theory for group systems that parallels the theory of ordinary linear systems over fields. Using only elementary group theory, we are able to develop such fundamental constructs of linear system theory as state spaces and minimal input-output realizations. Remarkably, these basic constructs depend only on additive (group) and not multiplicative (field) algebraic structure.
Acknowledgements

My greatest thanks go to my research advisors Tom Cover and Dave Forney. Their support, both academic and personal, has had a profound effect on my experience at Stanford. Interactions with Tom over the last few years have helped define my views on research and teaching. The standard he sets is one I aspire to meet.

The contents of this dissertation are in large measure the product of joint work with Dave Forney. His guidance, assistance and friendship leave me greatly indebted.

Ongoing correspondence with Andi Loeliger and Thomas Mittelholzer has contributed significantly to this work. I am also grateful for the help of my orals and reading committees members John Gill, Abbas El Gamal, and James Plummer.

My fellow Coverians Vittorio Castelli, Laura Ekroot, Will Equitz, Elza Erkip, Don Kimber, Drew Nobel, Erik Ordentlich, Jim Roche, Joy Thomas, Don Kimber, and Rick Wesel have made my time at Stanford enjoyable. I forgive you for not laughing at enough of my jokes. My friends Robert Lodenkamper, Tom Piatak, Erik Ruf, and Mike Yang endured even the bad jokes, a true test of friendship.

Sources of funding include NSF and IBM graduate fellowships as well as research assistantships supported by Seiko-Epson, the National Science Foundation and DARPA. These organizations made this work possible.

The support of my parents, brother and sister was invaluable, especially in times of doubt. I dedicate this thesis to them.
Contents

Abstract iii

Acknowledgements iv

1 Introduction 1

2 Dynamical Systems 5

2.1 Definitions and notation 6
2.2 Adjusting the flow of time 8
2.3 Controllability and observability 9

2.3.1 Controllability 9
2.3.2 Observability 12
2.3.3 Completeness 14
2.3.4 Examples 15

2.4 State and realizations 22

2.4.1 State-output realizations 22
2.4.2 Evolution laws, graphs and trellis sections 25
2.4.3 Canonical state-output realizations 26
2.4.4 Minimality 29
2.4.5 Controllable and observable state-output systems 31

2.5 Encoders 33

2.5.1 Invertibility and catastrophic encoders 34
2.5.2 Minimal encoders 36

3 Group Systems 37

3.1 Preliminaries 43

3.1.1 Group codes 43
3.1.2 Projections and subcodes 45
3.1.3 Controllability, observability and completeness 46

3.2 The state group 50

3.2.1 Definition and fundamental properties 50
3.2.2 The canonical minimal realization 54
3.2.3 Minimal trellis diagrams ....................................... 54
3.3 Inputs and state codes ................................................. 57
3.3.1 Inputs ............................................................... 57
3.3.2 The state code of $C$ .............................................. 59
3.4 Canonical encoders based on generators ......................... 61
3.4.1 Granules .......................................................... 61
3.4.2 Encoder construction .............................................. 66
3.4.3 Linear codes over fields ........................................... 70

4 Isometries and partitions .................................................. 73
4.1 Definitions .............................................................. 74
4.2 Group Actions .......................................................... 76
4.3 Geometrically uniform sets and partitions ......................... 77
4.3.1 Geometrically uniform sets ....................................... 78
4.3.2 Geometrically uniform partitions ............................... 80
4.4 Partitions generated by cosets of isometries ...................... 83
4.5 Normal geometrically uniform partitions .......................... 87
4.6 Constructing geometrically uniform partitions .................. 90

5 Isometry codes ............................................................ 93
5.1 The dynamics of generated Euclidean codes ....................... 94
5.2 Reduced codes and isometric labelings ............................. 96
5.3 Isometry and Euclidean trellis sections ........................... 97
5.4 Realizing existing codes as isometry codes ...................... 100
5.4.1 The V.32 code ..................................................... 101
5.4.2 Shift register graphs .............................................. 102
5.4.3 Testing for geometrically uniform partitions ................ 104
5.4.4 Finding a branch group ......................................... 105
5.4.5 The permutation labeling ....................................... 107
5.5 Two more examples .................................................... 110

6 Conclusions and open problems ......................................... 121

A Some elementary group theory ......................................... 123
List of Tables

2.1 Controllability, observability and completeness examples. . . . 16

4.1 Labeling of the partition $\mathbb{R}Z^2/4Z^2$. . . . . . . . . . 90

5.1 Group table for $D_4$, the symmetries of the square. . . . . . 106
5.2 The action of generator branches on the cosets of $B_a$. . . . 108
5.3 Permutation labeling for the 32 edges of the V.32 code. . . . 109
5.4 The correspondence between $(Z_2)^3$ and the partition cells of $Z^4/RD_4$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 112
5.5 The labeling of the cells of $Z^6/RD_6$. . . . . . . . . . . . . . 115
5.6 The assignment of $D_4 \times Z_2$ to trellis states. . . . . . . . 117
5.7 The consistent permutation labeling of generator branches. . 118

A.1 Group table for $D_3$, the symmetries of a triangle. . . . . . 127
List of Figures

1.1 A prototypical communication system. ................. 1

2.1 $a$ is $[k, k + \nu)$ controllable to $b$ ................. 10
2.2 $[k, k + \mu)$-observability ........................................ 13
2.3 The graph of an unobservable, complete, strongly controllable system .............................................. 19
2.4 The graph of a complete, strongly observable and strongly controllable system ........................................... 20
2.5 A sample trajectory in a controllable, strongly observable system ......................................................... 21
2.6 The axiom of state ............................................................. 23
2.7 The trellis diagram of a two-state realization. .............. 26
2.8 The past-induced realization of Example 2.27. ............... 27
2.9 The past-induced realization of Example 2.28. ............... 28
2.10 The future-induced realization of Example 2.28. ............ 29
2.11 The minimal realization of Example 2.28. .................... 30

3.1 Encoder and trellis diagram for 4-state rate-1/2 binary convolutional code. ................................. 38
3.2 Trellis diagram for an (8,4) binary block code. ............. 39
3.3 Encoder and 4-state trellis diagram for rate-1/2 convolutional code over $Z_4$. ................................. 40
3.4 Minimal encoder and 2-state trellis diagram for the same code. 41
3.5 The subcodes $C_{k-}$ and $C_{k+}$. ................................. 51
3.6 The zero state $C_{k-}C_{k+}$. ........................................ 53
3.7 Canonical realization of a length-2 code. ......................... 55
3.8 The trellis diagram of $C_{\text{even}}$. ................................. 56
3.9 A canonical encoder for a group code $C$ using trellis sections $T_{k,k+1}$. ................................. 59
3.10 The granule $\Gamma_{[k,k+1]}$. ........................................ 63
3.11 Minimal encoder in controller canonical form. ................... 67

4.1 An 8-PSK signal set. ............................................... 76
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Two geometrically uniform partitions of 8-PSK.</td>
<td>81</td>
</tr>
<tr>
<td>4.3</td>
<td>Two inequivalent 4-way partitions of the vertices of a cube.</td>
<td>81</td>
</tr>
<tr>
<td>4.4</td>
<td>The geometrically uniform partition $\mathbb{R}Z^2/\mathbb{A}Z^2$.</td>
<td>89</td>
</tr>
<tr>
<td>5.1</td>
<td>Trellis diagram for the V.32 code.</td>
<td>102</td>
</tr>
<tr>
<td>5.2</td>
<td>Three time-invariant 8-state granule decompositions.</td>
<td>103</td>
</tr>
<tr>
<td>5.3</td>
<td>Three 8-state shift-register trellis sections.</td>
<td>103</td>
</tr>
<tr>
<td>5.4</td>
<td>Isometry generator sequence for the V.32 code.</td>
<td>109</td>
</tr>
<tr>
<td>5.5</td>
<td>The orbits of states under 90 degree rotation.</td>
<td>116</td>
</tr>
<tr>
<td>A.1</td>
<td>The second isomorphism theorem with two normal subgroups.</td>
<td>128</td>
</tr>
<tr>
<td>A.2</td>
<td>The third isomorphism theorem.</td>
<td>129</td>
</tr>
<tr>
<td>A.3</td>
<td>The correspondence theorem.</td>
<td>130</td>
</tr>
<tr>
<td>A.4</td>
<td>The homomorphism/normal subgroup theorem.</td>
<td>131</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In this dissertation we seek an algebraic structure theory for trellis codes. An ideal theory would describe constructive families of good trellis codes; at present, the dominant means of finding new codes remains a computer search combined with heuristics. While we fall short of this goal, in that we present no new trellis codes, our proposed framework includes many trellis codes not previously thought to have algebraic descriptions. We therefore expect that the results on group systems and isometry codes developed here represent the first steps toward a comprehensive theory.

A trellis code is a collection of real-valued or vector-valued sequences, usually described as the set of output sequences of a finite state encoder. A simplified trellis-coded communication system is shown in Figure 1.1. The channel in the figure is an amalgam of a modulator, demodulator and communication medium; in modem applications, the channel is commonly approximated as discrete, memoryless and corrupted by additive white Gaussian noise.

Figure 1.1: A prototypical communication system.

The sequences produced by the encoder can be interpreted as points in a
high-dimensional space. The decoder "quantizes" the received point to the nearest (most likely) valid code sequence. The error-correcting properties of a trellis code arise from the large distances that separate the possible output sequences: even in the presence of noise, one sequence is not likely to be mistaken for another. The best known trellis codes approach Shannon's channel capacity to within 3 dB [59, 60].

Ungerboeck [58] originally described trellis codes as a combination of a binary linear convolutional code and a mapping from output bits to points in a Euclidean signal space. This representation has several advantages. First, many good trellis codes have such a description. In addition, the theory of linear systems over fields [13, 14] facilitates efficient computer searches for new codes of this type.

Unfortunately, Ungerboeck's representation of trellis codes obscures the relationship between the algebraic structure of the binary code and the geometric structure of the signal space code. Only by coupling these two types of structure can one hope to find a constructive theory for generating good trellis codes.

Several alternative descriptions of trellis codes more closely relate algebraic and geometric structure. Calderbank and Sloane [8] define a class of convolutional "coset codes" over rings in which coded outputs are mapped to cosets of a partitioned lattice. Forney [16, 17] considers in detail the class of coset codes that consists of binary linear encoders followed by a natural mapping from output bits to lattice cosets. Massey and Mittelholzer [42, 43, 46, 47], and independently Filho et al. [11, 12], develop a theory of convolutional codes over rings for phase modulation trellis codes.

These and other representations of trellis codes are characterized by Forney [19] as "generalized coset codes." A generalized coset code is the combination of a group code and an isometric labeling, where a group code is a collection of sequences of group elements that is closed under a componentwise group operation, and an isometric labeling is a type of mapping from group elements to cells of a partitioned Euclidean signal set that preserves algebraic structure. A similar concept was introduced independently by Loeliger [38].

A trellis code with such a representation is geometrically uniform [19], and therefore has a number of strong symmetry properties. Foremost among them is the uniform error property: when the code is used over an additive white Gaussian noise channel with maximum likelihood decoding, the probability of error is independent of the transmitted code sequence. This property greatly simplifies performance analysis.
We represent trellis codes as systems over groups of isometries (distance-preserving transformations) of Euclidean space. An isometry code is a collection of sequences of isometries such that the componentwise composition of two code sequences is another code sequence. An isometry code is “converted” into a trellis code by applying a sequence of isometries in the code componentwise to an initial Euclidean sequence \( \mathbf{x} \); the set of all such sequences is a trellis code in an infinite-dimensional Euclidean space. Systems over isometry groups are a natural synthesis of the ideas of Forney [19] and Loeliger [38].

Many good trellis codes have isometry code representations. We show that the “nonlinear” 8-state, two-dimensional Wei code used in V.32 modems has a representation as an isometry code, as does the 16-state, four-dimensional Wei code used in higher-speed modems. The V.32 code cannot be represented as a linear system over a ring or a field.

Our results have two major parts. First, we develop a theory for group systems that parallels the theory of ordinary linear systems over fields. Using only elementary group theory, we are able to develop such fundamental constructs of linear system theory as state spaces and minimal input-output realizations. Remarkably, these basic constructs depend only on additive (group) and not multiplicative (linear) algebraic structure.

Second, we show how to realize existing trellis codes as isometry codes. Isometry codes provide a bridge between the algebraic and geometric structure of trellis codes. Our results on group systems allow us to relate the dynamics of an isometry code to the dynamics of the Euclidean code it generates.

The contents of this dissertation are organized as follows.

In Chapter 2 we investigate the structure of abstract dynamical systems. Following Willems [67], we define a dynamical system as a collection of sequences together with a flow of time. The purpose of this study is to show that the linear-system-theoretic concepts of controllability, observability, state and input-output realizations can be reformulated in a less structured setting. Willems’ approach to system theory is particularly relevant to trellis coding, where collections of sequences, rather than particular realizations, are of primary importance. We extend Willems’ program by introducing a new definition of observability, and we provide a comprehensive set of examples.

In Chapter 3 we develop the foundational results in the theory of group systems. A group system is a collection of sequences that is closed under
a componentwise group operation. As with "ordinary" linear systems over fields, group systems have essentially unique minimal realizations. We show how to construct these realizations and show how to build minimal feedforward encoders in canonical form. Large portions of Chapter 3 are adapted from Forney and Trott [22]. Loeliger and Mittelholzer [39, 40], with whom we have worked closely, present similar results from a different viewpoint.

In Chapter 4 we examine the relationship between collections of left cosets $U/U'$ of isometry groups and geometrically uniform Euclidean signal sets and partitions. We identify $\Gamma(Y)/\Gamma(Y)_Y$ as the canonical generating quotient for a geometrically uniform partition $Y$. The results of this chapter are an extension of those of Forney [19] and Loeliger [38].

In Chapter 5 we develop a procedure for realizing trellis codes as isometry codes. Three detailed examples are presented: Wei’s V.32 code, a four-dimensional code of Wei, and an unusual six-dimensional code of Chouly and Sari. These are the first interesting examples of trellis codes with nonabelian state groups.

In Appendix A we review the basic tools needed from group theory, namely homomorphisms and the isomorphism theorems, which can be found in an early chapter of any textbook on group theory. Remarkably, our main results require only these elementary tools.

In Chapter 6 we conclude with a survey of open questions.
Chapter 2

Dynamical Systems

We begin with an abstract treatment of discrete time dynamical systems. In this general setting, a dynamical system is nothing more than a collection of sequences together with a flow of time. Our study is motivated by the need to identify the parts of classical linear system theory over fields that carry over to settings with weaker algebraic structure.

Our approach to system theory differs substantially from the traditional treatment. From the viewpoint of classical system theory, a system consists of inputs, states and outputs together with a set of equations that describe how these variables evolve over time. Following Willems [65, 66, 67], we instead define a system as a collection of allowed behaviors. Inputs, states, and outputs are not identified \textit{a priori} by the theory; instead, these components of a system are inferred from analysis. State, for example, is that part of a system which satisfies state-like properties.

Our main concern is the application of system theory to coding theory. A code is a collection of sequences (codewords) over some alphabet. An encoder is constructed from a code by associating state and input sequences with each code sequence. Thus, in coding theory, state and input are derivative rather than primary concepts. For this reason our viewpoint has much in common with Willems', and we take [67] as our basic systems theory text.

There are, however, some differences. In Willems view, an abstract dynamical system is constructed as a model of a real or imagined physical system. In coding theory, a physical system such as a modem is constructed from the abstract model. Unlike Willems, we do not merely discover the inputs to the system—we assign them. Collections of output sequences, rather than particular input-output realizations, are of prime importance. Causality plays a limited role, for the code designer has explicit control over which
parts of the system are inputs and which are outputs.

The chapter is organized as follows. Dynamical systems are defined in Section 2.1, together with notation for manipulating sequences and subsequences. The ordering on the time axis is an important part of the definition of a dynamical systems; we consider in Section 2.2 how changing this ordering affects the system. Abstract definitions of controllability, observability and completeness are presented in Section 2.3. Surprisingly, these concepts exist with minimal assumptions on the structure of a system.

Section 2.4 introduces state and state-output realizations of dynamical systems. We show how to construct two canonical realizations defined by Willems [67]; these realizations are globally minimal if and only if they are equivalent. We then show how notions of state controllability and observability relate to the abstract definitions given in the previous section.

One of our goals in the next chapter is to construct minimal encoders, or input-state-output realizations, for group systems. To this end we must define encoders and what it means for one to be minimal. We do this in Section 2.5.

### 2.1 Definitions and notation

A dynamical system is a collection of sequences. The sequences are elements of a Cartesian product sequence space

$$\mathcal{W} = \prod_{k \in I} G_k,$$

(2.1)

where $I$ is some countable ordered time axis, and \{\(G_k : k \in I\)\} is an indexed collection of finite or infinite alphabets. If all alphabets \(G_k\) are equal to a common alphabet \(G\), then \(\mathcal{W} = G^I\) is the sequence space over \(G\) defined on \(I\). Sequences in a sequence space all have the same length, whether finite, half-infinite, or bi-infinite.

**Definition 2.1:** A discrete time dynamical system \(\mathcal{B}\) is a subset of a sequence space \(\mathcal{W}\).

The sequences of \(\mathcal{B}\) represent the set of allowed behaviors or trajectories of the system. Thus, in a broad sense, the study of dynamical systems is the study of the structure of subsets of Cartesian products.
2.1. DEFINITIONS AND NOTATION

Since the time axis $I$ is assumed to be countable and ordered, without loss of generality we take $I$ to be a subset of the integers $\mathbb{Z}$. (The study of codes defined on partially ordered index sets, such as subsets of $\mathbb{Z}^2$, is a subject for further investigation.) Most commonly $I$ is an interval, expressed in standard notation as, for example, $[a, b) = \{ k \in \mathbb{Z} : a \leq k < b \}$. Most of our examples will use either the finite time axis $I = [1, N]$, for block codes, or the infinite time axis $I = \mathbb{Z}$, for bi-infinite or convolutional codes. Of course, any system defined on a finite time axis $I$ can be trivially extended to a system defined on $\mathbb{Z}$ by setting the alphabets $G_k = \{0\}$ for $k \notin I$.

Let $J$ be a subset of the index set $I$. The complement of $J$ in $I$ is the set $I \setminus J$ of all $k \in I$ such that $k \notin J$. The complement of $I$ itself is the empty set $\emptyset$.

The ordering on the time axis $I$ determines a flow of time. At each time $k$, the time axis is partitioned into two intervals: the past $k^- = (-\infty, k) \cap I$ and the future $k^+ = (k, \infty) \cap I$. Note that the present time $k$ is part of the future; this asymmetry will propagate into many of our results. The partition of time into past and future is central to the definition of state given in Section 2.4.

If $\mathbf{x}$ is a sequence in $\mathcal{W}$, then the $k^{th}$ component of $\mathbf{x}$ is denoted by $x_k$, and the sequence $\mathbf{x}$ itself is denoted by

$$ (\ldots, x_{-1}, x_0, x_1, \ldots) $$

The optional vertical bar indicates the location of the time index $k = 0$. So, for example, $\mathbf{x} = (\ldots, 3 | 2, 1, \ldots)$ has $x_{-1} = 3, x_0 = 2$, and $x_1 = 1$. The portion of $\mathbf{x}$ defined on the interval $[a, b] \subseteq I$ is denoted by $\mathbf{x}_{[a, b]} = (x_a, x_{a+1}, \ldots, x_b)$. In particular, the present of $\mathbf{x}$ is $\mathbf{x}_{[k, k]} = x_k$, the past of $\mathbf{x}$ is $\mathbf{x}_{k^-} = (\ldots, x_{k-2}, x_{k-1})$, and the future of $\mathbf{x}$ is $\mathbf{x}_{k^+} = (x_k, x_{k+1}, \ldots)$.

We will often need to construct new sequences from pieces of existing sequences. If $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ are sequences in $\mathcal{W}$, then $\mathbf{w} = \mathbf{x} |_{\mathbf{y}} |_{\mathbf{z}}$ is the sequence defined by $w_{a^-} = x_{a^-}, w_{[a, b]} = y_{[a, b]}$ and $w_{b^+} = z_{b^+}$.

Notation for sequences of sequences is occasionally useful. If $\{\mathbf{x}_n : n \in J\}$ is an indexed family of sequences, then the $k^{th}$ component of the $n^{th}$ sequence is denoted by $(\mathbf{x}_n)_k$.

**Example 2.1:** Let $I = [1, 2]$ be the time axis, and let $G_1 = G_2 = \{a, b, c\}$ be identical three-letter alphabets. Then the sequence space $\mathcal{W} = G_1 \times G_2$ is the set of 9 sequences

$$ \mathcal{W} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, a), (c, b), (c, c)\}. $$
The set of 7 sequences

\[ \mathcal{B} = \{(a, a), (a, b), (a, c), (b, a), (b, c), (c, a), (c, b)\} \]

is a dynamical system \( \mathcal{B} \subseteq \mathcal{W} \). The "dynamics" of \( \mathcal{B} \) will become more evident when we discuss state-output realizations in Section 2.4. □

The output alphabet \( \mathcal{A}_k \) of a system \( \mathcal{B} \) is defined as the set of elements \( x_k \) that occur at time \( k \) in the sequences of \( \mathcal{B} \):

\[ \mathcal{A}_k = \{x_k : \mathcal{x} \in \mathcal{B}\}. \] (2.2)

A system is trim if \( \mathcal{A}_k = \mathcal{G}_k \) for all \( k \in I \) [67]. In a trim system every letter of every alphabet appears in some sequence. A trim system can always be extracted from a system that is not trim by appropriately shrinking the alphabets \( \mathcal{G}_k \).

Bi-infinite sequence spaces of the form \( \mathcal{W} = \mathcal{G}^\mathbb{Z} \) support a delay operator \( D \) that acts on sequences \( \mathcal{x} \in \mathcal{W} \) via

\[ D(\ldots, x_{-1} | x_0, x_1, \ldots) = (\ldots, x_{-2} | x_{-1}, x_0, \ldots). \] (2.3)

Equivalently, \( (D\mathcal{x})_k = x_{k-1} \) for \( k \in \mathbb{Z} \). A system \( \mathcal{B} \subseteq \mathcal{W} \) is time invariant if \( DB = B \). A system is periodic if \( D^pB = B \), where \( D^p \) is the composition of \( D \) with itself \( p \) times. The least such \( p \geq 1 \) is the period of \( \mathcal{B} \). Time-invariant systems have period 1. If \( \mathcal{B} \) is time invariant then its output alphabets \( \{\mathcal{A}_k : k \in \mathbb{Z}\} \) are all equal.

Example 2.2: Let \( \mathcal{W} = \mathcal{G}^\mathbb{Z} \) be a sequence space with alphabet \( \mathcal{G} = \{a, b, c\} \), and let the system \( \mathcal{B} \subseteq \mathcal{W} \) be the set of sequences \( \mathcal{x} \in \mathcal{W} \) such that \( x_k \neq x_{k+1} \) for all \( k \in \mathbb{Z} \). Then the dynamical system \( \mathcal{B} \) is both trim and time invariant. □

The theory of symbolic dynamics [1] concerns itself with time-invariant systems over finite alphabets, as in the preceding example. Willems restricts his treatment of linear systems to time-invariant linear systems, as do Loeliger and Mittelholzer [40]. We will not make this assumption.

2.2 Adjusting the flow of time

The dynamics of a system \( \mathcal{B} \) depend critically on the ordering of the time axis \( I \) and on its granularity.
2.3. CONTROLLABILITY AND OBSERVABILITY

Reordering time is a natural operation for block codes. Which component of, say, a Hamming code is first and which is last? When trying to find a minimal state realization for a block code, as in Kasami, Takata, Fujiwara and Lin [33], some orderings of time permit substantially smaller state sets than others.

The time axis can be adjusted in granularity to either hide or reveal additional structure of a system. For example, a sequence space \( \mathcal{W}_1 = \prod_{k \in \mathbb{Z}} G_k \) can be reassembled into sequence space \( \mathcal{W}_2 = \prod_{k \in \mathbb{Z}} (G_{2k} \times G_{2k+1}) \) by taking successive alphabets in pairs. The dynamics of a system \( B_1 \subseteq \mathcal{W}_1 \) will in general be quite different from the dynamics of the corresponding system \( B_2 \subseteq \mathcal{W}_2 \).

Another useful operation is to clump an infinite time axis into a finite number of intervals. For example, \( I = \mathbb{Z} \) can be broken into the finite set

\[
J = \{m^-, m, m+1, \ldots, n-1, n^+\}.
\]

(2.4)

In this way any convolutional code can be treated as a block code, where the infinite past and future are grouped into a single time index each.

2.3 Controllability and observability

Controllability and observability are properties usually associated with state-space realizations of linear systems. In this section we formulate definitions of controllability and observability that apply to dynamical systems in general. Surprisingly, the definitions make no explicit reference to "state."

Completeness is a topological property that ensures that a system can be defined solely in terms of local constraints on trajectories. The minimal encoder constructions for group codes presented in Chapter 3 apply to complete, strongly controllable codes.

The definitions of controllability and completeness are due to Willems [67]; observability also appears in Willems under the name "finite memory." We explore these definitions in somewhat greater depth than Willems, and provide a collection of examples which illustrate how the concepts interact.

2.3.1 Controllability

A system is controllable in the traditional sense if any state can be reached from any other. Willems [67] defines a system as controllable if the past of
any trajectory can be attached to the future of any other trajectory after a suitable transition interval. We extend Willems' definition to time varying systems.

![Diagram](image)

Figure 2.1: \(a\) is \([k, k + \nu)\) controllable to \(b\)

Let \(\mathcal{B}\) be a dynamical system with time axis \(I\). Given two trajectories \(a, b \in \mathcal{B}\), a time \(k \in I\), and a duration \(\nu \geq 0\), we say that \(a\) is \([k, k + \nu)\)-controllable to \(b\) if there exists a trajectory \(c \in \mathcal{B}\) such that \(c_{k-} = a_{k-}\) and \(c_{(k+\nu)^+} = b_{(k+\nu)^+}\). In words, there must exist a trajectory \(c = a|_{k} \ x|_{k+\nu} b\) that agrees with \(a\) on the past \(k^-\) and with \(b\) on the future \((k+\nu)^+\), as shown in Figure 2.1. The transition region between times \(k\) and \(k + \nu\) is constrained only by the requirement that \(c\) be a trajectory in \(\mathcal{B}\).

**Definition 2.2:** A system \(\mathcal{B}\) is *weakly controllable* if for all \(a, b \in \mathcal{B}\) and all \(k \in I\) there exists a \(\nu \geq 0\) such that that \(a\) is both \([k, k + \nu)\)-controllable and \([k - \nu, k)\)-controllable to \(b\).

**Definition 2.3:** A system \(\mathcal{B}\) is *strongly controllable* if there exists a \(\nu \geq 0\) such that \(a\) is \([k, k + \nu)\) controllable to \(b\) for all \(a, b \in \mathcal{B}\) and all \(k \in I\). The minimum such \(\nu\) is the *controllability index* of \(\mathcal{B}\); a strongly controllable system with controllability index \(\nu\) is \(\nu'\)-controllable for \(\nu' \geq \nu\). A 0-controllable system is *memoryless*, while a 1-controllable system has *unit memory*.

In a weakly controllable system, the sufficiently distant past places no constraint on the future and the sufficiently distant future places no constraint on the past. In a strongly controllable system, the past and future interact only over a transition interval of duration \(\nu\).
2.3. CONTROLLABILITY AND OBSERVABILITY

Example 2.3: Any system with a finite time axis $I$ is strongly controllable. □

Example 2.4: The system $\mathcal{B}$ defined in Example 2.2 is 1-controllable. □

While this definition of controllability may seem unusual, it is entirely analogous to the standard notion of controllability understood for linear systems over fields. For example, in a trim $\nu$-controllable state-output system, every state at time $k + \nu + 1$ can be reached from every state at time $k$. We will consider this in more detail in Section 2.4.

Strong controllability plays an important role in Chapter 3, where it is used in the strong controllability theorem for group codes (see Section 3.1.3). On the other hand, the definition of weak controllability is speculative. A weaker definition appears in [23]; loosely, we require here that a weakly controllable system be controllable infinitely often, while in [23] a weakly controllable system need only be controllable at one time.

Willems' definition of controllability relies on a notion of "nearness" in time determined by the ordering on the time axis $I$. We generalize Willems' definition to systems with unordered time axes by measuring nearness in terms of neighborhoods. Let $\mathcal{N}$ be a collection of subsets of $I$ such that each $k \in I$ is contained in some neighborhood $N \in \mathcal{N}$. The set $\mathcal{N}$ is a family of neighborhoods for $I$. A neighborhood cover is an indexed set of neighborhoods $\{N_k : k \in I\}$ of $\mathcal{N}$ such that $k \in N_k$ for $k \in I$.

Given two trajectories $a, b \in \mathcal{B}$, a subset $J \subseteq I$, and a family of neighborhoods $\mathcal{N}$, we say that $a$ is $\mathcal{N}$-controllable to $b$ on $J$ if there exists a neighborhood cover $\{N_k : k \in I\}$ and a trajectory $c \in \mathcal{B}$ such that $c$ agrees with $a$ on $J$ and agrees with $b$ on $I \setminus \bigcup_{k \in J} N_k$. A system $\mathcal{B}$ is $\mathcal{N}$-controllable if, for all subsets $J \subseteq I$ and all trajectories $a, b \in \mathcal{B}$, $a$ is $\mathcal{N}$-controllable on $J$ to $b$.

Example 2.5: A system defined on $I = \mathbb{Z}$ is strongly $\nu$-controllable if and only if it is $\mathcal{N}$-controllable for $\mathcal{N} = \{[k, k + \nu) : k \in \mathbb{Z}\}$; it is weakly controllable if and only if it is $\mathcal{N}$-controllable for $\mathcal{N} = \{[k, k + \nu) : \nu \geq 0, k \in \mathbb{Z}\}$. □

Example 2.6: Strong controllability can be defined for systems with multi-dimensional time axes. If the time axis is $I = \mathbb{Z} \times \mathbb{Z}$, then

$$\mathcal{N} = \{[k, k + \nu) \times [k, k + \nu) : k \in \mathbb{Z}\}$$

(2.5)

gives a reasonable notion of strong controllability. □
A maximal controllable subsystem is a subset of a dynamical system that is contained in no larger controllable subsystem. It is tempting, though incorrect, to conjecture that an uncontrollable system can always be partitioned into a disjoint union of maximal controllable subsystems. The conjecture fails because controllability is not an equivalence relation: if a trajectory \( \mathbf{a} \) is controllable to \( \mathbf{b} \) and if \( \mathbf{b} \) is controllable to \( \mathbf{c} \), then \( \mathbf{a} \) may or may not be controllable to \( \mathbf{c} \).

**Example 2.7:** Let \( \mathcal{B} \) be the dynamical system that consists of all sequences in \( \{0, 1, 2\}^\mathbb{Z} \) such that 1's and 2's never appear in the same sequence. Then \( \mathcal{B} \) is uncontrollable because, for example, the constant 1 sequence cannot be controlled to the constant 2 sequence. The maximal controllable subsystems of \( \mathcal{B} \) are the set of sequences that contain only 0's and 1's, and the set of sequences that contain only 0's and 2's. The constant 0 sequence appears in both, so the two subsystems are not a partition of \( \mathcal{B} \).

### 2.3.2 Observability

In linear system theory, observability and controllability are dual. Kailath [31] defines a realization of a linear system as (state) observable if the future state sequence can be recovered from the future input and output sequences. Similarly, Willems [67] defines observability to be the ability to deduce the value of one part of a system trajectory from another.

Our approach is more ambitious. We formulate a definition of observability that, like the definition of controllability given in the previous section, does not make explicit reference to states or to realizations. In this sense, our definition of observability is a better dual controllability than Willems'.

Let \( \mathcal{B} \) be a dynamical system defined on \( I \). Given a trajectory \( \mathbf{a} \in \mathcal{B} \) and an interval \( [k, k + \mu) \subseteq I \), we say that \( \mathbf{a} \) is \( [k, k + \mu) \)-observable if for all trajectories \( \mathbf{b} \in \mathcal{B} \) that agree with \( \mathbf{a} \) on \( [k, k + \mu) \), the sequence \( \mathbf{c} = \mathbf{a} |_{k} \mathbf{b} \) is a trajectory in \( \mathcal{B} \). This is illustrated in Figure 2.2. In other words, a trajectory is observable on an interval if its future is conditionally independent of its past given the interval. In symbolic dynamics, the portion \( \mathbf{a}_{(k,k+\mu)} \) of a \( [k, k + \mu) \)-observable sequence \( \mathbf{a} \) is a "magic word."

**Definition 2.4:** A system \( \mathcal{B} \) is weakly observable if for all trajectories \( \mathbf{a} \in \mathcal{B} \) and all \( k \in I \), there exists a \( \mu \geq 0 \) such that \( \mathbf{a} \) is both \( [k, k + \mu) \)-observable and \( [k - \mu, k) \)-observable.
When two trajectories in a weakly observable system agree over a sufficiently long interval, the past of one can be concatenated to the future of the other to form a valid trajectory. A system is strongly observable if the required observation interval width is bounded by some integer $\mu$.

**Definition 2.5:** A system $\mathcal{B}$ is **strongly observable** if there exists a $\mu \geq 0$ such that $a$ is $[k, k + \mu)$-observable for all $a \in \mathcal{B}$ and all $k \in I$. The minimum such $\mu$ is the **observability index** of $\mathcal{B}$; a strongly observable system with observability index $\mu$ is $\mu$-observable. A 0-observable system is **memoryless**, while a 1-observable system is **Markovian**.

As with weak controllability, our definition of weak observability is speculative.

Willems refers to the observability index as the **memory span** of the system. A $\mu$-observable system is said to have $\mu$-finite memory. Willems does not, however, make a connection between finite memory and observability, as we do here.

A system is 0-observable (memoryless) if and only if it is 0-controllable. In a memoryless system the past of any trajectory can be concatenated with the future of any other.

Observability can be defined for systems with unordered time axes in a manner analogous to controllability. We do not carry out this generalization here.
2.3.3 Completeness

In simplest terms, a system is complete if its behavior is determined locally. Willems views completeness as an essential property of dynamical systems:

It can be said that the study of non-complete systems does not fall within the competence of system theorists and could better be left to cosmologists or theologians [66, p. 567].

While Willems' view is perhaps extreme, complete systems avoid certain technical problems in realization theory (see Section 2.4). Completeness is a standard assumption in symbolic dynamics, and in Chapter 3 we restrict our study of group systems to complete ones.

Let $B_{[k,k+\mu]}$ denote the restriction of a system $B \subseteq W$ to the interval $[k, k + \mu]$.

**Definition 2.6:** A system $B \subseteq W$ is complete if $B$ consists of precisely those sequences $a \in W$ for which $a_{[k,k+\mu]} \in B_{[k,k+\mu]}$ for all $k \in I$ and all $\mu \geq 0$.

**Definition 2.7:** A system $B$ is $\mu$-complete if $B$ consists of the sequences $a \in W$ for which $a_{[k,k+\mu]} \in B_{[k,k+\mu]}$ for all $k \in I$. A 0-complete system is instantly specified.

The behavior of a complete system is determined by local constraints on trajectories. A system is $\mu$-complete if its behavior is determined by constraints on intervals of length $\mu$.

Completeness has a topological interpretation. Specifically, a system is complete if and only if it is a closed set in the product topology of the discrete topology on the symbol alphabets. We will identify the closed sets in this topology using a metric $d: W \times W \to R$.

Given two sequences $w, w' \in W$, define $d(w, w') = 2^{-i}$, where $i$ is the greatest integer such that $w_{[i,i]} = w'_{[i,i]}$. A sequence of sequences $w_0, w_1, \ldots$ in $W$ converges to a limit $w \in W$ if for all $\varepsilon > 0$ there exists an integer $N$ such that $d(w, w_k) \leq \varepsilon$ for $k \geq N$. In other words, a sequence converges in the product topology if it converges pointwise. A set $B \subseteq W$ is closed if contains the limit of every convergent sequence $b_0, b_1, \ldots$ of elements of $B$.

**Theorem 2.1:** A system $B$ is complete if and only if it is closed in the product topology.
2.3. CONTROLLABILITY AND OBSERVABILITY

**Proof:** A system $\mathcal{B}$ is a closed set if and only if the limit of any convergent sequence of trajectories in $\mathcal{B}$ is also in $\mathcal{B}$.

Assume $\mathcal{B}$ is complete. Let $x_0, x_1, \ldots$ be a convergent sequence of trajectories in $\mathcal{B}$ with limit $x$. To show that $x \in \mathcal{B}$, we must establish that $x_{[a,b]} \in \mathcal{B}_{[a,b]}$ for all finite intervals $[a, b]$. Let $[a, b]$ be a finite interval. Since the sequence of trajectories converges pointwise to $x$, there exists an $N$ such that $(x_i)_{[a,b]} = x_{[a,b]}$ for all $i \geq N$, hence $x_{[a,b]} \in \mathcal{B}_{[a,b]}$.

Assume $\mathcal{B}$ is closed. Given any $x \in \mathcal{W}$, we must show that $x \in \mathcal{B}$ when $x_{[a,b]} \in \mathcal{B}_{[a,b]}$ for all intervals $[a, b]$. We prove this by constructing a sequence of trajectories in $\mathcal{B}$ that converges pointwise to $x$. For $i = 0, 1, 2, \ldots$ let $x_i$ be any trajectory in $\mathcal{B}$ that agrees with $x$ on the interval $[-i, i]$. Such a trajectory must exist because $x_{[-i,i]} \in \mathcal{B}_{[-i,i]}$. By construction, $x_0, x_1, \ldots$ converges pointwise to $x$, hence the limit $x$ is a trajectory in $\mathcal{B}$. ■

Simple consequences of the equivalence of completeness and closure follow from elementary topology. In particular, any system with a finite time axis $I$ is complete, as is any intersection of complete systems. Stronger results hold for systems over groups (see Theorem 3.1 in Section 3.1.3).

The *completion* (or *closure*) of an incomplete system $\mathcal{B}$ is the intersection of all complete systems in $\mathcal{W}$ that contain $\mathcal{B}$. The closure of an incomplete system $\mathcal{B}$ may be quite different from $\mathcal{B}$ itself. For example, the closure of a controllable system may be uncontrollable, or vice versa.

Completeness and weak observability are independent, as illustrated by the examples in Section 2.3.4. Example 2.13 is incomplete and weakly observable, while Example 2.15 is complete and unobservable.

Strong observability and $\mu$-completeness, on the other hand, are close cousins. The following theorem is proved by Willems [67, Prop. 1.1, p. 185].

**Theorem 2.2:** A dynamical system is $\mu$-complete if and only if it is complete and $\mu$-observable.

2.3.4 Examples

It is a useful exercise to exhibit the 18 possible combinations of completeness, controllability and observability. These examples help illustrate how the properties of controllability and observability interact, and allow alternative definitions to be tested.
It can be shown (as a consequence of compactness) that a finite-alphabet, time-invariant, complete, weakly observable system must be strongly observable. Also, a finite-alphabet, time-invariant, weakly controllable, strongly observable system must be strongly controllable. Except in the five cases covered by these conditions, our examples are drawn from the restricted class of finite-alphabet time-invariant systems. In addition, we have selected the examples to be invariant under time reversal. A list of examples and their properties is displayed in Table 2.1.

<table>
<thead>
<tr>
<th>Example</th>
<th>controllable</th>
<th>observable</th>
<th>complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.19</td>
<td>strongly</td>
<td>strongly</td>
<td>strongly</td>
</tr>
<tr>
<td>2.11</td>
<td>strongly</td>
<td>weakly</td>
<td>weakly</td>
</tr>
<tr>
<td>2.15</td>
<td>strongly</td>
<td>not</td>
<td>weakly</td>
</tr>
<tr>
<td>2.8</td>
<td>strongly</td>
<td>strongly</td>
<td>not</td>
</tr>
<tr>
<td>2.13</td>
<td>strongly</td>
<td>weakly</td>
<td>not</td>
</tr>
<tr>
<td>2.9</td>
<td>strongly</td>
<td>not</td>
<td>not</td>
</tr>
<tr>
<td>2.20</td>
<td>weakly</td>
<td>strongly</td>
<td>strongly</td>
</tr>
<tr>
<td>2.12</td>
<td>weakly</td>
<td>weakly</td>
<td>weakly</td>
</tr>
<tr>
<td>2.16</td>
<td>weakly</td>
<td>not</td>
<td>weakly</td>
</tr>
<tr>
<td>2.21</td>
<td>weakly</td>
<td>strongly</td>
<td>not</td>
</tr>
<tr>
<td>2.17</td>
<td>weakly</td>
<td>weakly</td>
<td>not</td>
</tr>
<tr>
<td>2.10</td>
<td>weakly</td>
<td>not</td>
<td>not</td>
</tr>
<tr>
<td>2.22</td>
<td>not</td>
<td>strongly</td>
<td>strongly</td>
</tr>
<tr>
<td>2.23</td>
<td>not</td>
<td>weakly</td>
<td>weakly</td>
</tr>
<tr>
<td>2.18</td>
<td>not</td>
<td>not</td>
<td>weakly</td>
</tr>
<tr>
<td>2.23</td>
<td>not</td>
<td>strongly</td>
<td>not</td>
</tr>
<tr>
<td>2.23</td>
<td>not</td>
<td>weakly</td>
<td>not</td>
</tr>
<tr>
<td>2.14</td>
<td>not</td>
<td>not</td>
<td>not</td>
</tr>
</tbody>
</table>

Table 2.1: Controllability, observability and completeness examples.

**Example 2.8:** (strongly controllable, strongly observable, not complete) Let $B$ be the set of sequences in $\{0, 1\}^\mathbb{Z}$ that contain a finite number of 1's. Then $B$ is memoryless (0-observable and 0-controllable). The system is not complete, because every sequence in $\{0, 1\}^\mathbb{Z}$ has a finite number of 1's in every finite interval, whether or not it is a sequence in $B$. In other words, the "finite ones" constraint cannot be verified over finite intervals. The completion of $B$ is $\{0, 1\}^\mathbb{Z}$. □
Example 2.9: (strongly controllable, not observable, not complete) Let $B$ be the set of sequences in $\{0,1\}^\mathbb{Z}$ that contain a finite, even number of 1's. The system has controllability index 1 because any past with a finite number of 1's can be attached to any future with a finite number of 1's given an appropriately chosen length-1 transition interval.

The system is not observable. For all $\mu \geq 0$, let $a_\mu$ be a trajectory that has an odd number of 1's in the past, is 0 on $[0,\mu]$, and has an odd number of 1's in the future. Then $a_\mu$ agrees with the constant 0 sequence on $[0,\mu]$, but the past of the constant 0 sequence cannot be concatenated to the future of $a_\mu$ to form a trajectory.

The system $B$ is not complete because it is not closed. To see this, let $a_j$ be the trajectory defined for $k \in \mathbb{Z}$ by

$$
(a_j)_k = \begin{cases} 
1 & \text{if } -j \leq k < j, \\
0 & \text{otherwise.}
\end{cases}
$$

(2.6)

Then the sequence of trajectories $a_0, a_1, a_2, \ldots$ converges to the constant 1 sequence, which is not a trajectory in $B$. The closure of $B$ is the whole space $\{0,1\}^\mathbb{Z}$. □

Example 2.10: (weakly controllable, not observable, not complete) Let $B$ be the set of sequences in $\{0,1\}^\mathbb{Z}$ that have a prime number of 1's. The system is incomplete and unobservable for the same reasons as Example 2.9. Its closure is all of $\{0,1\}^\mathbb{Z}$.

The system is controllable but not strongly controllable. To attach any past to any future, choose the transition interval width $\nu$ to extend beyond all future 1's. Then fill in the transition interval with enough 1's to reach a prime total.

The system is not strongly controllable because the gaps between successive prime numbers may be arbitrarily large. To control a past with a large finite number of 1's to a constant 0 future may require an arbitrarily long transition interval. The interval must be long enough to contain sufficient 1's to reach a prime total. □

Example 2.11: (strongly controllable, weakly observable, complete) Let $B$ be the set of bi-infinite sequences of strictly positive integers such that integer $i$ occurs only in runs of length $i$ or less. For example,

$$
(\ldots, 3, 3, 3, 1, 3, 3, 2, 2, 4, 1, \ldots)
$$
is a trajectory in $\mathcal{B}$, but

$$(\ldots, 1, 1, 2, 2, 2, 3, 3, 3, 3, \ldots)$$

is not.

It is easy to see that the system is strongly controllable with controllability index 1. Any past $n_{k-1}^-$ can be concatenated with any future $y_{(k+1)+}$ by selecting a transition symbol $n_k$ that is not equal to $n_{k-1}$ or $y_{k+1}$.

The system is weakly observable. Any sufficiently long observation interval must contain the start of a new run, after which the past is independent of the future.

The system is complete. Any excessively long run is detected by checking a sufficiently long finite interval. $\square$

**Example 2.12:** (weakly controllable, weakly observable, complete) Let $\mathcal{B}$ be the set of bi-infinite sequences of strictly positive integers such that integer $i$ occurs in runs of length exactly $i$.

The system is weakly controllable because the transition interval required to attach any past to any future must be long enough to span the current run.

The system is weakly observable and complete for the same reason as Example 2.11. $\square$

**Example 2.13:** (strongly controllable, weakly observable, not complete) Let $\mathcal{B}$ be the set of sequences in $\{0, 1\}^\mathbb{Z}$ such that 1's occur in finite runs of even length and 0's occur singly. For example, $(\ldots, 0, 1, 1, 0, 1, 1, 1, 1, 0, \ldots)$ is a trajectory in $\mathcal{B}$.

The controllability index is at least 3. The trajectory $(\ldots, 1|1, 0, 1, 1, \ldots)$ is $[0, 3)$-controllable but not $[0, 2)$-controllable to $(\ldots, 0|1, 1, 0, 1, \ldots)$.

The system is weakly observable because any sufficiently long observation interval must contain a 0, after which the past and future are independent.

The completion of $\mathcal{B}$ is the set of sequences for which 1's occur in either infinite runs or in finite runs of even length. $\square$

**Example 2.14:** (not controllable, not observable, not complete) Let $\mathcal{B}$ be the set of sequences in $\{0, 1\}^\mathbb{Z}$ that have either a finite number of 0's or a finite number of 1's. The system is uncontrollable because a past containing
2.3. CONTROLLABILITY AND OBSERVABILITY

ininitely many 0's cannot be controlled to a future containing infinitely many 1's. The closure of $B$ is $\{0,1\}^\mathbb{Z}$. The system is unobservable because two sequences may agree on an arbitrarily long finite interval, yet the past of one may have infinitely many 0's while the future of the other may have infinitely many 1's. □

![Graph](image)

Figure 2.3: The graph of an unobservable, complete, strongly controllable system.

**Example 2.15:** (strongly controllable, not observable, complete) Let $B$ be the set of sequences in $\{0,1,2\}^\mathbb{Z}$ for which 1's and 2's alternate, ignoring intervening 0's. The system is described as the set of symbol sequences produced by bi-infinite walks on the labeled, directed graph shown in Figure 2.3.

The system is not observable. Let $\mathbf{a} = (\ldots, 2, 1, 2, 1 | 0, 0, 0, \ldots)$ be a trajectory in $B$; then $\mathbf{a}$ is not $[0, \mu]$ observable for any $\mu$. To see this, let $\mathbf{b}$ be the trajectory $b_i = a_{\mu-i}$ for $i \in \mathbb{Z}$. Then $\mathbf{b}$ agrees with $\mathbf{a}$ on $[0, \mu]$, yet the sequence $\mathbf{c} = a | 0, b$ is not a trajectory in $B$ because it has two successive 1's separated by a length-$\mu + 1$ block of 0's.

In this example the abstract definition of observability agrees with the intuitive one: an unfortunate observer that sees only long strings of 0's can never recover the state of the system.

The system is complete, as is any system defined by the set of walks on a finite labeled directed graph. In particular, invalid sequences can always be detected by testing sufficiently wide finite intervals. The system is not $\mu$-complete because the interval widths are unbounded.

The existence of a two-state irreducible graph description of the system proves that $B$ is strongly controllable with controllability index no greater than 1. The controllability index is exactly 1 because a past ending in 2 cannot be attached to a future beginning with 2 without a transition symbol. □
Example 2.16: (weakly controllable, not observable, complete) Every sequence in \(\{0, 1\}^\mathbb{Z}\) can be parsed into alternating blocks of 0's and 1's. Let \(B\) be the set of sequences in \(\{0, 1\}^\mathbb{Z}\) such that each block of 1's in the sequence has either infinite length or length \(2^n\) for some integer \(n \geq 0\).

The system is not observable because the sequence \((\ldots, 0, 0, 0 | 1, 1, 1, \ldots)\) is not \([0, \mu]\)-observable for any \(\mu\).

The system is complete, though not \(\mu\)-complete, because a block of 1's whose length is not a power of 2 is always detected by a sufficiently large interval.

The system is controllable: place as many 1's in the transition interval as necessary to make a legal block, then add 0's until a 0 is encountered in the future. \(\square\)

Example 2.17: (weakly controllable, weakly observable, not complete) Modify Example 2.16 so that infinite blocks of 1's are excluded. The new system is incomplete; its completion is the system defined in Example 2.16. The system is observable, but not strongly observable, because a sufficiently large observation interval must contain a 0, after which the future is not influenced by the past. \(\square\)

Example 2.18: (not controllable, not observable, complete) Let \(B\) be the set of sequences in \(\{0, 1, 2\}^\mathbb{Z}\) such that 1's and 2's never appear in the same sequence. The system is complete, for if 1's in 2's appear in the same sequence then there is some finite interval that contains both a 1 and a 2. The system is uncontrollable because the constant 1 sequence cannot be controlled to the constant 2 sequence. The system is unobservable because the sequence \((\ldots, 1, 1, 1 | 0, 0, 0, \ldots)\) is not \([0, \mu]\)-observable for any \(\mu\). \(\square\)

![Figure 2.4: The graph of a complete, strongly observable and strongly controllable system.](image)
2.3. CONTROLLABILITY AND OBSERVABILITY

Example 2.19: (strongly controllable, strongly observable, complete) Let $B$ be the set of sequences in $\{0,1\}^\mathbb{Z}$ such that two 0's never appear adjacent. In magnetic coding circles this is called a (0,1)-runlength constraint. Its graph is shown in Figure 2.4. The system is Markovian (1-observable) and 1-controllable.

Time-invariant $\mu$-complete systems over finite alphabets are known as shifts of finite type in the symbolic dynamics literature. They can always be represented as the set of walks on a finite graph. If there exists an irreducible graph representation, then the system is strongly controllable; if not, it is uncontrollable. □

![Figure 2.5: A sample trajectory in a controllable, strongly observable system.](image)

Example 2.20: (weakly controllable, strongly observable, complete) Systems of this type either have infinite alphabets or are time varying. Let $\mathcal{W} = (\mathbb{Z} \times \mathbb{Z})^\mathbb{Z}$. The trajectories of the system are ordered pairs $(x, y)$ of integer sequences. A sample trajectory, for simplicity drawn with a continuous time axis, is shown in Figure 2.5. The magnitude of the $y$ sequence determines height the peaks in the $x$ sequence. A sequence $(x, y)$ is a trajectory in the system if and only if, for all $k \in \mathbb{Z}$, $0 \leq x_k \leq y_k$, $|x_{k+1} - x_k| \in \{0, 1\}$, and

$$
(x_{k+1}, y_{k+1}) = \begin{cases} 
(\{0,1\}, \mathbb{Z}^+) & \text{if } x_k = 0 \text{ (x has reached a valley),} \\
(x_k - 1, y_k) & \text{if } x_k = y_k \text{ (x has reached a peak),} \\
(x_k + 1, y_k) & \text{if } x_k > x_{k-1} \text{ (x is increasing),} \\
(x_k - 1, y_k) & \text{if } x_k < x_{k-1} \text{ (x is decreasing).}
\end{cases}
$$

(2.7)

The system evolves deterministically until $x_k$ reaches 0, at which point $y_{k+1}$ can assume any positive integer value. Once $x$ leaves 0, its climbs linearly to $y$ and then descends linearly to 0. The behavioral equations that describe the system span three components, hence the system is 3-complete and strongly observable with observability index 3.
The system is controllable because the system "resets" at valleys. Any past can be concatenated with any future provided that the transition interval is wide enough to allow the past to reach a valley and the future to begin at a valley. The interval between these valleys may be arbitrarily large, hence the system is not strongly controllable. □

Example 2.21: (weakly controllable, strongly observable, not complete) Let $B$ be the set of bi-infinite sequences of positive integers that consist of concatenations of finite sequences of the form $(0,1,\ldots,n-1,n,n-1,\ldots,2,1)$ for $n \geq 0$. For example, $(\ldots,2,1,0,1,0,1,2,3,2,1,0,\ldots)$ is a trajectory in $B$. The system $B$ is the projection of the system defined in Example 2.20 onto its $z$ component.

The system is not complete because its completion includes the infinite ascending sequence. The completion of $B$ is not controllable. □

Example 2.22: (not controllable, strongly observable, complete) Let $B$ be the two-element system in $\{0,1\}^\mathbb{Z}$ that consists of the constant 0 sequence and the constant 1 sequence.

As a second example, let $B$ be the periodic system that consists of the two possible sequences of alternating 0's and 1's. □

Example 2.23: Examples of uncontrollable systems that are (i) weakly observable and complete, (ii) strongly observable and incomplete, and (iii) weakly observable and incomplete are easily constructed from Examples 2.12, 2.21 and 2.17 by adding the constant 1 sequence to the systems. □

2.4 State and realizations

In this section we introduce a definition of state advanced by Willems [67], and show how the abstract definitions of observability and controllability apply to state-output realizations of dynamical systems.

2.4.1 State-output realizations

Following Willems [67], we do not view state as a component present a priori in a dynamical system. Instead, state is a derivative property determined by the constraint placed on the future by the past.
2.4. STATE AND REALIZATIONS

State in a dynamical system is a natural consequence of the flow of time. At each time \( k \), the order on the time axis \( I \) partitions a trajectory \( a \in B \) into a known "past" \( a_{k^-} \) and an unknown "future" \( a_{k^+} \). The constraint the known past places on the unknown future is the essence of state. We will use this notion explicitly when we construct minimal realizations in the next section.

A state-output system is a special type of system in which the trajectories \( x = (s, y) \) are pairs of state sequences \( s \) and output sequences \( y \). The alphabet \( G_k \) at each time \( k \) in a state-output system is a Cartesian product of a state set \( S_k \) and an output set \( Y_k \). Sequences in \( \mathcal{W} = \prod G_k \) are pairs \( (s, y) \), where \( s \in \prod S_k \) and \( y \in \prod Y_k \). (More properly, the trajectories in a state-output realization are sequences of pairs, not pairs of sequences. We will not distinguish sharply between these cases.) The system passes through state sequence \( s \) while producing output sequence \( y \).

Definition 2.8: A state-output system \( B \) is a subset of a sequence space \( \mathcal{W} = \prod_{k \in I} (S_k \times Y_k) \) such that \( B \) satisfies the axiom of state, where \( S_k \) is the state alphabet and \( Y_k \) is the output alphabet for \( k \in I \).

![Figure 2.6: The axiom of state.](image)

The axiom of state [67] is illustrated in Figure 2.6: if two trajectories \( (s, y) \) and \( (s', y') \) in \( B \) pass through the same state \( s_k = s'_k \) at time \( k \), then the concatenation \( (s |_{k^-} s', y |_{k} y') \) of the past \( (s, y)_{k^-} \) of one to the future \( (s', y')_{k^+} \) of the other must be a trajectory in \( B \).

The axiom of state convincingly encapsulates intuitive notions of state. In particular, state acts as a sufficient statistic for the past given the future and for the future given the past. A state-output system that satisfies the axiom of state is Markovian (1-observable). The axiom of state agrees with
the concepts of state that appear in automata theory, linear system theory, and symbolic dynamics.

State can as easily be discovered as assumed: given a 1-observable dynamical system \( B \subseteq \prod_{k \in I} G_k \), a state-output system results from any projection of the symbol alphabets \( G_k \) into state-output components \( S_k \times Y_k \) such that the projected system satisfies the axiom of state. As a special case, any 1-observable system can be viewed as a state-output system by setting the state alphabet \( S_k = G_k \) and the output alphabet \( Y_k = \{0\} \) for \( k \in I \).

A state-output system \( B \) induces a state system \( B_S \) and an output system \( B_Y \). The state system \( B_S \subseteq \prod S_k \) is the set of state trajectories that occur in \( B \), while the output system \( B_Y \subseteq \prod Y_k \) is the set of output trajectories that occur in \( B \).

**Definition 2.9:** A state-output realization \( R \) of a system \( C \) is a state-output system such that \( R_Y = C \).

A state-output realization of a system \( C \) is a state-output system whose set of output trajectories is precisely \( C \). A given system has infinitely many possible realizations; we shall be most interested in realizations which are in some sense minimal.

**Definition 2.10:** A state output realization \( R \) of \( C \) is externally induced [67] if each trajectory in \( C \) appears in a unique trajectory of \( R \).

In an externally-induced realization, the state trajectory can be uniquely recovered from the output trajectory. The minimal realizations we construct in Chapter 3 for complete group systems will have this property.

**Example 2.24:** The labeled, directed graph in Figure 2.3 represents a time-invariant state-output realization \( R \) of the system \( B \subseteq \{0,1,2\}^Z \) defined in Example 2.15. The output alphabet \( Y_k \) is \( \{0,1,2\} \) and the state alphabet \( S_k \) is \( \{a,b\} \) for \( k \in Z \). With one exception, the realization assigns each trajectory \( y \) in \( B \) a unique state sequence determined by the unique bi-infinite walk that produces \( y \). For example, the output trajectory \( y = (\ldots,1,2,1|2,1,2,\ldots) \in B \) is paired with the state trajectory \( s = (\ldots,b,a,b|a,b,a,\ldots) \) to form the realization trajectory \( (s,y) \in R \). It is easy to verify that \( R_Y = B \) and \( R_S = \{a,b\}^Z \).

The realization is not externally induced, because the constant 0 output trajectory appears twice in the realization \( R \): once paired with the constant \( a \) state sequence and once with the constant \( b \) state sequence. \( \Box \)
2.4. STATE AND REALIZATIONS

2.4.2 Evolution laws, graphs and trellis sections

A labeled directed graph is a special type of what Willems terms a discrete-time evolution law. An evolution law \( \mathcal{L} = \{ \mathcal{L}_k : k \in I \} \) is an indexed collection of sets of triples \( \mathcal{L}_k \subseteq S_k \times Y_k \times S_{k+1} \) that specify the possible next states and outputs given the current state. Every state-output system \( \mathcal{B} \) induces an evolution law

\[
\mathcal{L}_k = \{(s_k, y_k, s_{k+1}) : (s, y) \in \mathcal{B}\}
\]  

for \( k \in I \). Conversely, every evolution law \( \mathcal{L} = \{ \mathcal{L}_k : k \in I \} \) induces a state-output system

\[
\mathcal{B} = \{(s, y) \in \mathcal{W} : (s_k, y_k, s_{k+1}) \in \mathcal{L}_k \text{ for } k \in I\}
\]  

defined by the set of sequences that satisfy the evolution law at each time \( k \).

Willems [67] proves that if \( \mathcal{L} \) is the evolution law induced by a state-output system \( \mathcal{B} \), then the state-output system induced by \( \mathcal{L} \) is the completion of \( \mathcal{B} \). In other words, complete state-output systems are uniquely determined by their evolution laws. Note, however, that an incomplete system can have a complete realization, as in the following example due to Loeliger and Mittelholzer [private correspondence].

**Example 2.25:** Let \( S_k = Y_k = Z \) for \( k \in Z \), and define the evolution law \( \mathcal{L} \) by \( \mathcal{L}_k = \{(s_k, 2u_k + s_k, u_k) : s_k, u_k \in Z\} \) for \( k \in Z \). Let \( \mathcal{B} \) be the (complete) state-output system induced by \( \mathcal{L} \). Then the output system \( \mathcal{B}_Y \) is not complete: trajectories in \( \mathcal{B}_Y \) may contain arbitrarily long blocks of 1's, yet the constant 1 sequence is not in the output system. \( \square \)

We will see in the next chapter that complete group systems have complete minimal realizations, and hence are uniquely determined by their evolution laws.

A time-invariant evolution law can be depicted as a labeled, directed graph. The nodes in the graph represent the states in the state set \( S \). The labels in the graph are elements of the output alphabet \( Y \). Each triple \( (s_k, y_k, s_{k+1}) \in S \times Y \times S \) in the evolution law appears as a directed edge from node \( s_k \) to node \( s_{k+1} \) with label \( y_k \). Conversely, a labeled, directed graph defines an evolution law, provided that any two edges that connect the same two states have different labels.

An infinite *walk* on a directed graph is a connected sequence of graph edges \( \{e_k : k \in Z\} \) such that for all \( k \in Z \) the target node of edge \( e_k \) is the
source node of edge \( e_{k+1} \). A walk defines an output trajectory in \( \Pi_{k \in \mathbb{Z}} Y_k \) found by reading the labels off the edges of the walk. The set of all walks on the graph determines the realization induced by the evolution law. The walk taken to reach a graph node does not restrict the choice of walks leaving the node—this is precisely the condition on state-output realizations established by the axiom of state.

A time-varying evolution law can be depicted as a trellis diagram. A trellis diagram is a type of labeled, directed graph that consists of consecutive trellis sections, one for each time \( k \in I \). The \( k^{th} \) section has one node for each element of the state set \( S_k \). A triple \((s_k, y_k, s_{k+1}) \in \mathcal{L}_k \) is represented by an edge that originates at node \( s_k \) in the \( k^{th} \) trellis section, terminates at node \( s_{k+1} \) in section \( k+1 \), and is labeled with \( y_k \).

![Trellis Diagram](image)

Figure 2.7: The trellis diagram of a two-state realization.

**Example 2.26:** The graph in Figure 2.3 is equivalent to the trellis diagram in Figure 2.7. □

### 2.4.3 Canonical state-output realizations

We next consider two canonical state-output realizations of a dynamical system; these are referred to by Willems [67] as past induced and future induced. We will use these realizations in the next chapter to construct minimal realizations of group systems.

In the past-induced realization of a system \( \mathcal{B} \), two output trajectories pass through the same state at time \( k \in I \) if they have the same set of possible future continuations. Specifically, two trajectories \( \mathbf{a}, \mathbf{a}' \in \mathcal{B} \) are past equivalent if

\[
\{ b \in \mathcal{B} : \mathbf{a}_k b \in \mathcal{B} \} = \{ b \in \mathcal{B} : \mathbf{a}'_k b \in \mathcal{B} \}.
\]  

(2.10)
2.4. **STATE AND REALIZATIONS**

Past equivalence partitions the trajectories of $B$ into equivalence classes at each time $k$. Each equivalence class becomes an element in the state alphabet $S_k$, and each trajectory $b$ in $B$ is paired with the unique state trajectory in $\prod_{k \in I} S_k$ that indicates $b$'s past-equivalent class membership for $k \in I$. The resulting externally-induced state-output system satisfies the axiom of state [67].

![Diagram](image)

**Figure 2.8:** The past-induced realization of Example 2.27.

**Example 2.27:** Consider the system, introduced in Example 2.1, defined on $I = [1,2]$ that consists of the seven length-2 sequences

$$B = \{(a,a), (a,b), (a,c), (b,a), (b,c), (c,a), (c,b)\}.$$  

The past-induced realization for this system is given in Figure 2.8. Every path from left to right through the graph is a trajectory in the realization. At time $k = 1$ all trajectories are past equivalent, hence there is one state at time 1 in the past-induced realization. At time $k = 2$ there are three possible pasts, namely $a$, $b$ and $c$. Each of these pasts is represented by a different state because each has a different set of possible futures: $a$ can be continued by $a$ or $b$ or $c$, $b$ can be continued by $a$ or $c$, and $c$ can be continued by $a$ or $b$. There is no time $k = 3$, but the graph includes a final state for the sake of symmetry. □

The past induced realization is known under various names. It appears in automata theory as a minimal deterministic automaton. A graph is deterministic or unifilar if the edges leaving any graph node have distinct output labels. In system theory, Nerode equivalence [31, p. 316] is similar to past equivalence. The Shannon cover is the closest analog in symbolic dynamics.
to the past-induced realization, though the past-induced realization agrees with the Shannon cover only for finite type (strongly observable) systems.

**Example 2.28:** An approximation of the past-induced realization of the system defined in Examples 2.15 and 2.24 is shown in Figure 2.9. The graph has three states rather than two because a trajectory with a constant 0 past can be continued by either a 1 or a 2. No other past has this set of future continuations, hence trajectories with 0 pasts are lumped into a special class under past equivalence.

The past-induced realization for this system cannot be interpreted as the set of all walks on a graph because the realization is not complete. For example, the constant $a$ and $b$ state trajectories are not in the realization, though they are valid bi-infinite walks. Still, the graph in Figure 2.9 is a useful tool for visualizing the realization dynamics.

The Shannon cover (Figure 2.3) and the past-induced realization (Figure 2.9) for this example do not agree. □

The previous example illustrates some limitations of the past-induced realization. First, it is not necessarily complete. Second, while there is no reason to expect the past-induced realization to have smaller state sets than all other realizations, it is not minimal even over deterministic ones.

The future-induced realization is defined analogously to the past-induced realization. Two trajectories $a, a' \in B$ are *future equivalent* if

$$\{b \in B : b \, |_k \, a \in B\} = \{b \in B : b \, |_k \, a' \in B\}. \quad (2.11)$$
In words, two trajectories are future equivalent if their futures have the same set of possible past continuations. As with past equivalence, the state set at each time $k$ in the future-induced realization is the set of equivalence classes determined by future equivalence.

![Diagram](image)

Figure 2.10: The future-induced realization of Example 2.28.

**Example 2.29:** The future-induced realization for the system of Example 2.27 is shown in Figure 2.10. The past- and future-induced realizations are related by a reflection for this example because the system is invariant under time reversal. This is not true in general. □

### 2.4.4 Minimality

Loosely, a realization is minimal if its state alphabets $S_k$ are as small as possible at each time $k \in I$. Willems [65, 67] considers the problem of defining minimality in some detail. We present only a brief overview of his results.

Willems defines a realization $\mathcal{R}$ of a system $\mathcal{B}$ to be *locally minimal* if it is trim and if no states of $\mathcal{R}$ can be pairwise merged to form a smaller realization of $\mathcal{B}$. This definition of minimality yields only a partial order on realizations; squeezing the state set down at one time may force it to balloon out at another.

Willems shows that the past-induced and future-induced realizations are locally minimal. More importantly for our purposes, Willems proves the following [67, Thm. 2.2]:

Theorem 2.3: The past-induced and future-induced realizations of a system $\mathcal{B}$ are equivalent if and only if all minimal realizations of $\mathcal{B}$ are equivalent.

Two realizations are equivalent if they are related by a renaming of states. Thus, there exists an essentially unique minimal realization for a system $\mathcal{B}$ if and only if the past-induced and future-induced realizations are identical, which is true if and only if past equivalence and future equivalence partitions the sequences of $\mathcal{B}$ into identical equivalence classes.

![Diagram](https://example.com/diagram.png)

Figure 2.11: The minimal realization of Example 2.28.

Example 2.30: The past-induced and future-induced realizations given in Examples 2.27 and 2.29 are not equal, hence there may or may exist a realization with strictly smaller state sets. In this case one does exist, as shown in Figure 2.11. This example illustrates a well-known result from automata theory: nondeterministic automata may have fewer states than deterministic ones. □

The situation depicted in the previous example never arises for linear systems or group system. Willems shows that the past-induced and future-induced realizations of linear systems over fields are the same, hence linear systems have unique minimal realizations. We will prove the same result for group systems in Chapter 3.

A second result, also due to Willems [67, Thm. 2.4, p. 212], shows that complete group systems have complete minimal realizations.

Theorem 2.4: If $\mathcal{B}$ is a complete system, and if the past-induced and future-induced realizations of $\mathcal{B}$ are equal, then all minimal realizations of $\mathcal{B}$ are complete.
2.4. STATE AND REALIZATIONS

In light of the comments in Section 2.4.2, a complete system whose past-induced and future-induced realizations are equal is uniquely described by its minimal trellis diagram.

2.4.5 Controllable and observable state-output systems

We are now in a position to justify the abstract definitions of controllability and observability given in Section 2.3. We show that strongly controllable systems have strongly state-controllable realizations, and that strongly observable systems have strongly state-observable realizations.

**Definition 2.11:** A state-output system $\mathcal{B}$ is strongly state observable with memory $m$ and anticipation $a$ if, for all times $k \in I$ and all trajectories $(s, y)$ and $(s', y')$ in $\mathcal{B}$, $y_{[k-m,k+a)} = y'_{[k-m,k+a)}$ implies that $s_k = s'_k$.

Thus, a realization is strongly state observable with memory $m$ and anticipation $a$ if the state at time $k$ can be determined by observing outputs on the interval $[k - m, k + a)$. (The terms “memory” and “anticipation” are borrowed from symbolic dynamics, e.g., Adler, Coppersmith, and Hassner [1].) We have not found a satisfactory definition of weak state observability.

In standard linear system theory, a realization is said to be observable if the current state can be deduced from a finite segment of the future. In our terminology, this corresponds to a state-observable realization with memory 0 and anticipation $a \geq 0$.

The following theorem establishes a close relationship between observability and state observability.

**Theorem 2.5:** If $\mathcal{B}$ is $\mu$-observable then it has a state-observable realization with memory $m$ and anticipation $a$ such that $m + a = \mu$. Conversely, if $\mathcal{B}$ has a state-observable realization with memory $m$ and anticipation $a$ then $\mathcal{B}$ is strongly observable with observability index $\mu \leq m + a$.

**Proof:** Assume $\mathcal{R}$ is a state-observable realization of $\mathcal{B}$ with memory $m$ and anticipation $a$. We must show that every $y \in \mathcal{B}$ is $[k, k + m + a)$-observable for $k \in I$. Let $(s, y)$ and $(s', y')$ be trajectories $\mathcal{R}$ such that $y_{[k,k+m+a)} = y'_{[k,k+m+a)}$. Then $s_{k+m} = s'_{k+m}$. The axiom of state implies that $(s_{[k+m} s', y_{[k+m} y')$ is a trajectory in $\mathcal{R}$, hence $y_{[k+m} y' = y_{[k} y'$ is a trajectory in $\mathcal{B}$. 
CHAPTER 2. DYNAMICAL SYSTEMS

Assume \( B \) is \( \mu \)-observable. Let \( R \) be the past-induced realization of \( B \). We claim that \( R \) is state observable with memory \( \mu \) and anticipation 0. It suffices to show that if two trajectories \( y, y' \in B \) agree on \([k-\mu, k)\) then they are past equivalent. This follows immediately from the definition of observability and past equivalence.

**Definition 2.12:** A state-output system \( B \) is **strongly state controllable** if there exists a \( \nu \geq 0 \) such that for all states \( s'_k \in S_k \) and \( s'_{k+\nu} \in S_{k+\nu} \), there exists a trajectory \((s, y) \in B\) such that \( s_k = s'_k \) and \( s_{k+\nu} = s'_{k+\nu} \). The minimum such \( \nu \) is the **state-controllability index** of \( B \).

A system is strongly state controllable if it can be driven from any state to any other in at most \( \nu \) steps. Willems [67] uses the term **point controllable** to describe a somewhat weaker notion of state controllability.

The output system \( R_Y \) of a strongly state-controllable system \( R \) with state-controllability index \( \mu \) is strongly controllable with controllability index less than or equal to \( \mu \). Conversely, the state-controllability index of the past-induced realization of a system \( B \) can be tightly upper-bounded using the observability and controllability indices of \( B \).

**Theorem 2.6:** If a system \( B \) is \( \nu \)-controllable and \( \mu \)-observable then the controllability index of its past-induced realization is at most \( \nu + \mu \).

**Proof:** Let \( x = (s, y) \) and \( x' = (s', y') \) be two trajectories in the past-induced realization \( R \) of \( B \). For all times \( k \in I \), we must find a realization trajectory \( x'' = (s'', y'') \) that agrees with \( x \) on \( k^- \) and with \( x' \) on \((k+\nu+\mu)^+\).

Let \( y'' \) be any trajectory in \( B \) that agrees with \( y \) on \( k^- \) and with \( y' \) on \((k+\nu)^+\). The trajectory exists because \( B \) is \( \nu \)-controllable. Let \( x'' = (s'', y'') \) be the unique trajectory in the past-induced realization \( R \) whose output sequence is \( y'' \).

The trajectories \( y'' \) and \( y \) agree on \( k^- \), hence \( x''_{k-} = x_{k-} \). If two trajectories in a \( \mu \)-observable system agree on the interval \([k+\nu, k+\nu+\mu] \) then they are past equivalent at time \( k+\nu+\mu \). The trajectories \( y'' \) and \( y' \) are therefore past equivalent at time \( k+\nu+\mu \), hence \( s_{k+\nu+\mu} = s'_{k+\nu+\mu} \). We conclude that \( x'' \) and \( x' \) agree on \( k+\nu+\mu^+ \).
2.5 Encoders

An input-state-output system, or encoder, is a type of realization in which the system trajectories are triples of input, state, and output sequences. An encoder can be thought of as a machine that maps input sequences drawn from a simple memoryless input space to output sequences drawn from a potentially complex set of trajectories. In this section we define encoders, invertibility, and minimality.

Most of our definitions so far have been time symmetric: the axiom of state, for example, does not depend on whether time flows from the past to the future or from the future to the past. The causality properties required of encoders break this symmetry.

As with state-output systems, input-state-output systems can be defined in terms of general properties that ensure encoder-like behavior. Willems takes this approach in [67], where he identifies “processing” and “nonanticipating” as key properties required of encoders. While Willems’ approach is both careful and convincing, its complexity diminishes its intuitive appeal. We therefore take a more direct approach.

We define an encoder by equations that govern its evolution in time. A (time varying) encoder $E$ is defined by input sets $U_k$, state sets $S_k$, output sets $Y_k$, state update functions $g_k: U_k \times S_k \rightarrow S_{k+1}$, and output functions $f_k: U_k \times S_k \rightarrow Y_k$ for all $k \in I$. The trajectories $(u, s, y)$ of an encoder $E$ are the combinations of input sequences $u \in \prod_{k \in I} U_k$, state sequences $s \in \prod_{k \in I} S_k$, and output sequences $y \in \prod_{k \in I} Y_k$ that satisfy the encoder’s dynamical equations

$$s_{k+1} = g_k(u_k, s_k) \quad (2.12)$$

$$y_k = f_k(u_k, s_k) \quad (2.13)$$

for all times $k \in I$. The axiom of state is implicit here, as is a notion of causality. In Willems’ terminology, Equations (2.12) and (2.13) are an input-state-output evolution law.

A single input sequence in an encoder can be associated with multiple output sequences, hence an encoder is not a mapping from inputs to outputs. More usual definitions of encoders, expressed in terms of half-infinite sequences, resolve this question by assuming a prespecified “initial state” at time $k = 0$ that disambiguates the input-output mapping. Sliding-block encoders, defined below, are a second way to avoid this question.

**Definition 2.13:** An encoder $E$ is sliding block with memory $m$ and an-
A sliding-block encoder is thus a mapping from input sequences to output sequences. An encoder is feedforward with memory \(m\) if it is sliding block with anticipation 0. Note that encoder state, unlike the output, is not necessarily a function of the \(m\) previous inputs. We will construct feedforward encoders for group systems in Chapter 3.

An encoder \(\mathcal{E}\) can be viewed as a state-output system by taking the input-output pair \((u, y)\) as the "output" and \(s\) as the state for each trajectory \((u, s, y) \in \mathcal{E}\). The trellis diagram of this state-output system has several important properties. Each branch in the trellis has an input and an output label. The branches leaving the same state have distinct input labels, and every input label appears on some exiting branch. Thus, at a given time \(k\), the number of branches leaving each state is exactly \(|U_k|\).

An input-state-output realization of a system \(\mathcal{B}\) is an encoder \(\mathcal{E}\) whose set of output trajectories \(\mathcal{E}_Y\) is precisely \(\mathcal{B}\). Not all systems have input-state-output realizations. For example, the time-invariant system shown in Figure 2.4 has irrational graph entropy, hence it has no time-invariant encoder. Symbolic dynamics considers the problem of constructing input-state-output systems that realize arbitrarily large fractions of a time-invariant finite-alphabet system \(\mathcal{B}\) [1, 41]. This problem will not arise in the next chapter because the same number of branches leave each state in the minimal trellis diagram of a group system.

### 2.5.1 Invertibility and catastrophic encoders

When an encoder is used as part of a communication system, the input sequence represents information to be stored, encoded, or transmitted. An encoder is useful in this capacity only if the input sequence can be uniquely recovered from the output sequence. Such an encoder is said to be invertible.

**Definition 2.14:** An encoder \(\mathcal{E}\) is invertible if for all output trajectories \(y \in \mathcal{E}_Y\), there exists a unique input trajectory \(u \in \mathcal{E}_U\) such that \((u, s, y)\) is a trajectory in \(\mathcal{E}\) for some state trajectory \(s \in \mathcal{E}_S\).

An alternative definition of invertibility can be formulated in terms of half-infinite sequences: an encoder is invertible if the future input is recoverable.
from the future output (and perhaps also the initial state). Differences between the half-infinite and bi-infinite definitions are most apparent when applied to unobservable systems. We will not pursue this further here.

An encoder has a sliding block inverse if the input at time \( k \) can be expressed as a function of a finite window on the output sequence around time \( k \). More precisely:

**Definition 2.15:** An encoder \( \mathcal{E} \) has a *sliding block inverse* with memory \( m \) and anticipation \( a \) if, for all trajectories \( (u, s, y), (u', s', y') \) in \( \mathcal{E} \) and all \( k \in I \), if \( y_{[k-m,k+a]} = y'_{[k-m,k+a]} \) then \( u_k = u'_k \).

An encoder has a *feedforward inverse* if it has a sliding block inverse with anticipation 0. An encoder is *systematic* if it has a sliding block inverse with memory 0 and anticipation 0. In a systematic encoder, the current input can be recovered as a function of the current output.

An encoder with a sliding block inverse might just as well be called an input-observable encoder, given the similarity between the above definition and the definition of a state-observable system (see Section 2.4.5).

**Definition 2.16:** An invertible encoder is *catastrophic* if there exist two output sequences that differ in finitely many components whose corresponding input sequences differ in infinitely many components.

When a catastrophic encoder is used over a noisy channel, a finite number of errors in an estimate of the output sequence can cause an infinite number of decoding errors. We will see in Section 3.3 that minimal encoders for group codes are never catastrophic.

An encoder with a sliding block inverse cannot be catastrophic. The converse, however, is false: the lack of a sliding block inverse does not imply that the encoder is catastrophic.

Convolutional coding literature often fails to distinguish noncatastrophic encoders from encoders that have sliding-block inverses, perhaps because the two concepts are equivalent for linear encoders over finite fields [45]. Symbolic dynamics literature is more careful; see, for example, Karabed and Marcus [32].

A sliding block encoder \( \mathcal{E} \) with a sliding block inverse establishes a bijection from input sequences to output sequences. In the language of symbolic dynamics, the input and output systems \( \mathcal{E}_U \) and \( \mathcal{E}_Y \) are then said to be *topologically conjugate*. 
2.5.2 Minimal encoders

Intuitively, an encoder that realizes a system $\mathcal{B}$ is minimal if there exists no other encoder for $\mathcal{B}$ with smaller state sets. Unfortunately, the "pairwise merging" notion of minimality introduced in Section 2.4.4 for state-output systems is not effective when applied unmodified to encoders.

Every encoder $\mathcal{E}$ has an underlying state-output system $\mathcal{E}_{S,Y}$ that consists of the pairs of state and output sequences that occur in the trajectories of $\mathcal{E}$. This realization can be found by stripping the input labels from the trellis diagram of $\mathcal{E}$.

**Definition 2.17:** An encoder $\mathcal{E}$ is minimal if its underlying state-output system $\mathcal{E}_{S,Y}$ is minimal.

This definition does not guarantee that minimal encoders exist, nor does it show how to create a minimal encoder from a nonminimal one. However, if the past-induced and future-induced realizations of a system $\mathcal{B}$ are equal, and if there exists an encoder for $\mathcal{B}$ whose underlying realization is the past-induced realization, then by Theorem 2.3 the encoder is minimal and essentially unique up to input relabeling. We show in the next chapter that these conditions are satisfied for group systems.
Chapter 3

Group Systems

A group code or group system $C$ is a set of sequences $c = \{c_k, k \in I\}$ that forms a group with a componentwise group operation. If all code symbols $c_k$ are drawn from a common group $G$, then we say that $C$ is a group code over $G$ defined on $I$. Group codes occupy a middle ground between linear systems over fields and the dynamical systems considered in the previous chapter.

In this chapter we investigate the dynamical structure of group codes. We find that such linear-system-theoretic constructs as state spaces, state-transition (trellis) diagrams, and minimal encoders may be developed using only elementary tools from group theory, and that all of these objects are essentially group-theoretic. One might conclude that, up to a point, the study of the dynamics of discrete-time linear systems is best approached as an exercise in elementary group theory.

The motivations for this study are several:

1. **Hamming space codes.** Most conventional Hamming space codes are linear codes over a field $F$ or a vector space $V$, and so are group codes over $F$ or $V$ as an additive group.

   For example, a binary linear $(N, K)$ block code $C$ is a $K$-dimensional subspace of the vector space $(F_2)^N$, where $F_2$ denotes the binary field GF(2). It follows that $C$ is also a subgroup of $(Z_2)^N$, where $Z_2$ denotes the binary group of integers modulo two. Thus $C$ is a group code over $Z_2$. In this case $C$ is defined on a finite index set $I$ of size $|I| = N$; e.g., $I = [1, N]$.

---

1This chapter is based on “The Dynamics of Group Codes: State Spaces, Trellis Diagrams and Canonical Encoders” by G. David Forney, Jr. and Mitchell D. Trott, which will appear in an upcoming issue of IEEE Transactions on Information Theory, 1993.
A binary linear rate-$k/n$ convolutional code $C$ is a subspace of the infinite-dimensional vector space $[[F_2]^n]^Z$. Thus $C$ is a group code over the group $(Z_2)^n$ defined on the time axis $I = Z$. Alternatively, $C$ is a group code over $Z_2$ defined on the Cartesian product index set $I = [1,n] \times Z$. A convolutional code is usually specified by a finite-state encoder $E$, or by a trellis diagram.

Example 3.1: Figure 3.1 shows a four-state encoder and a corresponding trellis diagram for the binary linear time-invariant rate-$1/2$ convolutional code $C$ that is generated by the sums of time shifts of the generator sequence $g = (\ldots, 00,11,01,11,00,\ldots)$. □

![Diagram](image)

Figure 3.1: Encoder and trellis diagram for 4-state rate-$1/2$ binary convolutional code.

Previous work on the algebraic structure of linear time-invariant convolutional codes over fields [13, 14, 15, 18, 44, 45] has successfully addressed the specification of minimal encoders and related questions such as the avoidance of catastrophic error propagation. We will show that most of these results may be derived using only the group structure of the field, independent of its multiplicative structure. This is surprising, since the earlier work relied heavily on such algebraic ideas as polynomial factorization, the representation of sequences as rational functions, and the invariant factor theorem (Smith-MacMillan canonical form). Furthermore, none of our results depend on time invariance.

A few previous papers [21, 17, 33, 69] have also investigated the trellis structure of linear block codes over fields. In particular, [17, Appendix] gives a general derivation of the trellis structure of linear block codes and lattices. This chapter generalizes these results. Again, the dynamical structure of a linear block code over a field $F$ does not depend on the multiplicative structure of $F$. 
Figure 3.2: Trellis diagram for an (8, 4) binary block code.

**Example 3.2:** Figure 3.2 shows a four-state trellis diagram for a binary linear (8, 4) block code (a Reed-Muller code, or an extended Hamming code), regarded as a code over \((\mathbb{Z}_2)^2\). The code is generated by the four sequences

\[
\begin{align*}
    g_1 &= (11, 11, 00, 00); \\
    g_2 &= (00, 11, 11, 00); \\
    g_3 &= (00, 00, 11, 11); \\
    g_4 &= (01, 01, 01, 01).
\end{align*}
\]

\[\square\]

2. **Euclidean space codes.** Good Euclidean space signal sets can often be characterized as the orbit \(\{u(x_0) : u \in U\}\) of an initial point \(x_0\) in a Euclidean vector space \(V\) under transformation by a group \(U\) of isometries of \(V\).

For example, Slepian’s “group codes for the Gaussian channel” [53] are signal sets (not codes) that are generated by a discrete group of orthogonal transformations of real \(n\)-space \(R^n\). An \(n\)-dimensional lattice \(\Lambda \subseteq R^n\), or a lattice translate \(\Lambda + x_0\), is generated by a discrete group of translations of \(R^n\). More generally, a “geometrically uniform signal set” [19] is generated by a discrete group of isometries of \(R^n\). We will consider geometrically uniform signal sets in detail in Chapter 4.

A group code \(C\) over a group of isometries \(U\) defined on an index set \(I\) generates a geometrically uniform signal space code \(\{c(x_0) : c \in C\}\) in a higher-dimensional space \(V^I\), or even an infinite-dimensional space \(V^\mathbb{Z}\) in the case of trellis codes [19]. Thus the study of group codes, possibly
over nonabelian groups, is a natural outgrowth of the study of geometrically uniform Euclidean-space codes.

"Mod-2" lattices and trellis codes are defined as the set of all integer sequences that are congruent modulo 2 to sequences in a binary linear block or convolutional code, respectively [16]. For example, the mod-2 trellis code corresponding to the rate-1/2 binary convolutional code of Figure 3.1 is a version of the Ungerboeck 4-state two-dimensional trellis code, and the mod-2 lattice corresponding to the (8, 4) binary block code of Figure 3.2 is a version of the Gosset lattice $E_8$. A mod-2 lattice or trellis code is represented by the same trellis diagram as the corresponding binary code, except that a trellis branch labeled with a binary $n$-tuple $b$ now represents the coset $2\mathbb{Z}^n + b$ of $2\mathbb{Z}^n$ in $\mathbb{Z}^n$, an infinite subset of $\mathbb{Z}^n$. Such codes are examples of "coset codes" [8, 16, 17].

It may be helpful to introduce a specific example of a Euclidean-space code that particularly motivated this work. This example was suggested by Massey et al. [43] to illustrate problems in specifying minimal linear encoders for convolutional codes over rings, which in turn was motivated by the problem of constructing rotationally invariant codes using PSK-type signal sets. A similar example appears in Kitchens [34, Example 3].

**Example 3.3:** Figure 3.3 depicts a four-state encoder and trellis diagram for a rate-1/2 linear time-invariant convolutional code $C$ over the ring $\mathbb{Z}_4$, defined as the set of all linear combinations (over $\mathbb{Z}_4$) of time shifts of the generator sequence $g = (\ldots, 00, 11, 13, 00, \ldots)$.

![Figure 3.3: Encoder and 4-state trellis diagram for rate-1/2 convolutional code over $\mathbb{Z}_4$.](image)
This encoder has two problems. First, it is catastrophic, because the infinite-weight input sequence \((\ldots, 0, 2, 2, 2, \ldots)\) generates the finite-weight code sequence \((\ldots, (0, 0), (2, 2), (0, 0), \ldots)\). (The zero branches are darkened in the trellis diagram.) Second, it is not minimal, because \(C\) can also be generated by the two-state encoder shown in Figure 3.4. In this encoder, an input in \(\mathbb{Z}_4\) is decomposed into an element of the subgroup \(2\mathbb{Z}_4\) and a representative of the quotient group \(\mathbb{Z}_4/2\mathbb{Z}_4\). As we shall see, a catastrophic encoder for a group code cannot be minimal.

![Diagram](image)

Figure 3.4: Minimal encoder and 2-state trellis diagram for the same code.

The results of this paper include a specification of minimal trellis diagrams and encoders as in Figure 3.4 that resolves such problems, using only group structure—in this case, for example, using only the group structure of \((\mathbb{Z}_4)^2\).

Interestingly, we find in general that a minimal encoder for a group code over a group may be nonlinear (nonhomomorphic). Furthermore, Kitchens [34, Example 4] shows that there exist group codes that are not isomorphic to a direct product \(G^2\) for any finite group \(G\). A feedforward homomorphic encoder establishes such an isomorphism, hence there exist group codes, as in the above example, that have no linear feedforward encoder.

3. **Linear systems.** The study of the dynamics of group codes is firmly embedded in linear system theory. Conversely, any ordinary discrete-time linear system may be regarded as a group code, since if the inputs and outputs of such a system are in a field, vector space, ring, or module, then set of pairs of input-output sequences may be regarded as a group code according to our definition.

However, there are significant differences between the viewpoint of coding theory and the usual viewpoint of linear system theory. In coding theory, one is primarily interested in the code \(C\). Finding an input-output description of \(C\), such as an encoder, is a secondary question.
As discussed in Chapter 2, Willems [65, 66, 67, 68] has championed a similar point of view in system theory. A dynamical system should be specified by the set $C$ of its possible behaviors (trajectories), and such constructs as state spaces, minimal realizations, and input-output behaviors of such realizations should be deduced from $C$. This chapter may be regarded as a contribution to Willems' program.

Willems shows that a dynamical system has an essentially unique minimal realization when its past-induced and future-induced realizations are equal. He then proves that any linear systems over a vector space $V$, such as $R^n$, has this property. Willems' linear system theory is derived using the multiplicative, geometric and topological properties of $V$ [67, Section 4].

We venture to suggest that our more primitive group-theoretic approach is more in the spirit of Willems' program than his own vector-space approach. Furthermore, it considerably extends the class of Willems-type systems that can be shown to have uniquely defined states and essentially unique minimal realizations.

There has been some interest in systems over rings, but little in systems over groups. Such work as there is, e.g., Brockett and Willsky [4] and Chizeck and Trott [9], takes the traditional input-output viewpoint.

Kitchens and Schmidt [34, 35] consider time-invariant subshifts (codes) over finite or compact groups from a symbol dynamics viewpoint; although not motivated by coding or system theory applications and restricted to time-invariant systems, this may be the closest prior work to ours.

4. Group theory. From the viewpoint of group theory, this chapter is simply a study of the structure of subgroups $C$ of direct product groups such as $G^N$ or $G^Z$. This must be an established topic in mathematics, but we are ignorant of the relevant literature.

Group theory is typically concerned with classifying groups (e.g., group codes) up to isomorphism. In coding theory, however, it is the particular embedding of $C$ in the direct product group that determines the key properties of the code. Furthermore, in system theory, it is the ordering of the index set $I$ that induces the dynamical structure of $C$. The viewpoints of coding and system theory may therefore shed some new light on this old topic.

The chapter is organized as follows. In Section 3.1 we introduce basic concepts, including a general definition of a group code $C$ and projections and subcodes of $C$. The definitions of controllability and observability given in the previous section are specialized to group codes. We show that a complete, $\nu$-controllable code $C$ is a product of sequences of length $\nu + 1$ or less.
3.1. **PRELIMINARIES**

In Section 3.2 we define the state group $\Sigma_k$ of a group code $C$ at time $k$. This definition is justified by showing that the properties expected of state are satisfied. Most importantly, given an ordered index set $I$, the state set at any time $k$ of any minimal realization of a group code $C$ must correspond to the state group $\Sigma_k$. This leads to a canonical state-output realization and trellis diagram of $C$.

In Section 3.3 we show that group codes have a natural input group that may be used as part of a canonical encoder structure. We also examine the state code of $C$, which is a group code that consists of all possible state sequences in $C$.

In Section 3.4, we construct a minimal feedforward encoder for a complete, $\nu$-controllable group code $C$ that uses sequences of length $\nu + 1$ as generators. This encoder has a layered structure, finite memory, and other desirable properties. The sizes of the state groups of $C$ may be determined by counting the states of the encoder. It reduces to the minimal encoders of Forney [13] and Roos [48] for linear time-invariant convolutional codes over fields.

### 3.1 Preliminaries

We first define group codes. We then discuss projections of group codes onto subsets of the index set $I$, and the images and kernels of such projections. Finally, we specialize the definitions of controllability, observability and closure (completeness) given in the previous chapter to systems over groups.

#### 3.1.1 Group codes

The following definition of a group code generalizes that given by Forney in [19]:

**Definition 3.1:** A group sequence space is a direct product group $W = \prod_{k \in I} G_k$, where the time axis $I$ is any subset of $\mathbb{Z}$, and the symbol groups $\{G_k : k \in I\}$ are arbitrary groups. A group code $C$ is a subgroup of a group sequence space. If all symbol groups are equal to a common group $G$, then we say that $C$ is a group code over $G$ defined on $I$.

As discussed in Section 2.2, the granularity of the time axis $I$ may be adjusted in various ways. For example, if $C$ is a code over $G$, then taking symbols in blocks of $N$ results in a description of $C$ in which the symbol
CHAPTER 3. GROUP SYSTEMS

group is $G^N$. Conversely, if the symbol group $G$ is itself a direct product, then a finer granularity may be achieved by taking the components of $G$ as individual symbols. For example, by taking symbols in pairs, the rate-1/2 convolutional code $C$ over $(Z_2)^2$ of Figure 3.1 becomes a unit-memory rate-2/4 convolutional code over $(Z_2)^4$; if the components of the symbol group $(Z_2)^2$ are taken as individual symbols, then $C$ becomes a time-varying group code over $Z_2$.

No restrictions are placed on the groups $G_k$. In particular, they may be nonabelian or infinite. However, to maintain contact with such linear-system-theoretic concepts as the zero state, the zero sequence, and so forth, we denote the identity of every group by $0$, the identity sequence by $0$, and the trivial group by the set $\{0\}$. The group operation is referred to as a "product" or "combination" of elements.

We define group codes as collections of two-sided infinite sequences. An alternative approach, commonly taken in linear system theory, considers systems in the subgroup $W_L$ of $W$ that consists of the "one-sided" sequences (formal Laurent series) $g$ for which $g_k = 0$ for $k$ less than some integer $b(g)$. The one-sided restriction ensures that convolutions of sequences are well defined, and that rational functions (ratios of polynomials) may be unambiguously identified with sequences in $W_L$. In the absence of multiplicative structure, however, neither of these considerations is operative. Moreover, a one-sided approach complicates the definition of rotationally invariant codes and breaks the symmetry between the past and the future. We therefore take a two-sided (bi-infinite) approach.

Sequences over groups can be assigned weights and lengths. A sequence $g$ begins at time $m$ if $g_m \neq 0$ but $g_k = 0$ for $k < m$, and ends at time $n$ if $g_n \neq 0$ but $g_k = 0$ for $k > n$. The (Hamming) weight of a sequence is the number of its nonzero components; a sequence is finite if its weight is finite. A sequence that starts at $m$ and ends at $n$ has length $n - m + 1$. The zero sequence $0$ is finite but has no length.

It is always possible to find a group $G$ that contains isomorphic copies of all symbol groups $G_k$, e.g., $G = W$. Thus every group code may be regarded as a group code over some common symbol group $G$. However, it is usually more convenient to restrict the symbol group $G_k$ to the set of symbols that are actually used in $C$. Recall from Section 2.1 that the output alphabet $A_k$ at time $k \in I$ of a code $C \subseteq W$ is the set of elements $c_k$ that occur in the sequences $c \in C$:

$$A_k = \{c_k : c \in C\}. \quad (3.1)$$
3.1. PRELIMINARIES

The output alphabet $A_k$ of a group code $C$ is a subgroup of the symbol alphabet $G_k$.

A code $C$ is **trim** if $G_k = A_k$ for $k \in I$. The **support** of a linear code $C$ is the subset $J$ of $I$ for which $A_k$ is nontrivial. A code $C$ defined on $I$ with support $J$ is equivalent to a code defined on $J$.

### 3.1.2 Projections and subcodes

Many of the results of this chapter are derived using projections and subcodes as the primary group-theoretic objects. These simple concepts will take us surprisingly far.

**Definition 3.2:** Given a subset $J \subseteq I$, the **projection** $P_J : \mathcal{W} \to \mathcal{W}$ is the map that sends a sequence $g \in \mathcal{W}$ to the sequence $h \in \mathcal{W}$ defined by

$$h_k = \begin{cases} g_k & \text{if } k \in J, \\ 0 & \text{if } k \notin J. \end{cases} \quad (3.2)$$

The projection $P_J$ acts by “zeroing out” the components of a sequence outside of $J$.

Projections are homomorphisms, since $P_J(gh) = P_J(g)P_J(h)$ for all $g, h \in \mathcal{W}$. The image of a group code $C \subseteq \mathcal{W}$ under the projection $P_J$ is the **projection of $C$ onto $J$**. A projection $P_J(C) \subseteq P_J(\mathcal{W})$ is thus a group code defined on $I$ with support contained in $J$.

The subset $J$ is typically an interval. For example, $P_{[m,n)}$ denotes the projection onto the interval $J = [m,n)$, while $P_{k+}$ and $P_{k-}$ denote projections onto the *future* $k^+ = [k, \infty) \cap I$ and the *past* $k^- = (-\infty, k) \cap I$ with respect to $k$.

**Definition 3.3:** Given a group code $C$ and a subset $J \subseteq I$, the **subcode** $C_J$ is the set of code sequences whose components are zero outside of $J$:

$$C_J = \{ c \in C : P_{I\setminus J}(c) = 0 \}. \quad (3.3)$$

Alternatively, $C_J$ is the set of sequences of $C$ that are invariant under the projection $P_J$:

$$C_J = \{ c \in C : P_J(c) = c \} = C \cap P_J(C). \quad (3.4)$$
CHAPTER 3. GROUP SYSTEMS

For example, the subcode $C_{[m,n]}$ is the set of sequences $c \in C$ whose components are zero outside of the interval $[m,n)$. The subcodes $C_{k^+}$ and $C_{k^-}$ are the sets of sequences $c \in C$ whose components are zero in the past and future with respect to $k$, respectively.

The subcode $C_{I \setminus J}$ is the kernel of the projection $P_{I}: C \rightarrow W$. The fundamental homomorphism theorem implies that $C_{I \setminus J}$ is a normal subgroup of $C$, and that the quotient group $C/C_{I \setminus J}$ is isomorphic to $P_{I}(C)$. Reversing the roles of $J$ and $I \setminus J$, it follows that $C_{J}$ is a normal subgroup of $C$, and

$$C/C_{J} \cong P_{I \setminus J}(C). \quad (3.5)$$

Projections $P_{J}(C)$ and subcodes $C_{J}$ are thus group codes defined on $I$ whose support is contained in $J$. Projections and subcodes are fundamental to the algebraic structure of $C$, and play dual roles.

If $C_{J}$ and $C_{J'}$ are subcodes of $C$, then the product $C_{J}C_{J'}$ is a normal subgroup of $C$. Furthermore, if the subsets $J$ and $J'$ are disjoint, then $C_{J}C_{J'}$ is an internal direct product (see Appendix A).

We shall not distinguish sharply between the codes $P_{J}(C)$ and $C_{J}$, which are defined on $I$, and the equivalent codes defined on $J$. For example, the output group $A_{k}$ is naturally isomorphic to the projection $P_{[k,k]}(C)$ of $C$ onto the length-1 interval $J = [k,k]$; depending on context, we shall refer to either $A_{k}$ or $P_{[k,k]}(C)$ as the output group at time $k$.

3.1.3 Controllability, observability and completeness

The definitions of controllability, observability and completeness given in the previous chapter have special consequences when applied to group codes.

In general, we shall require that a group code $C$ be closed, so that it can be described locally in time by a trellis diagram, and strongly controllable, so that it can be generated by a finite-memory minimal encoder. (For pointing out the necessity of closure, we thank Loeliger and Mittelholzer [private communication].) Staiger [55, 56] appears to be one of the first authors to recognize the importance of closure in convolutional coding theory.

Recall that a system is complete if it is a closed set in the product topology of the discrete topology on the symbol alphabets. A group sequence space $W$ endowed with this topology is a topological group [30].

Example 3.4: The set of finite sequences $C_{f}$ of a time-invariant group code
3.1. PRELIMINARIES

$C$ is an incomplete group code when nontrivial. Similarly, the set of one-sided sequences (formal Laurent series) $W_L$ is incomplete. □

Using the theory of topological groups, it can be shown, for example, that the closure of a group code is a group code.

We will sometimes take infinite products of normal, closed subgroups $\{C_k : k \in \mathbb{Z}\}$ of a closed group code $C$. The product is defined only when the support of each subgroup intersects only finitely many others. Then $\prod_{k \in \mathbb{Z}} C_k$ is defined as the set of all products $\{\prod_{k \in \mathbb{Z}} c_k : c_k \in C_k\}$, where

$$\prod_{k \in \mathbb{Z}} c_k = \lim_{i \to \infty} c_{-i} c_{-i+1} \cdots c_{i-1} c_i. \quad (3.6)$$

The assumption that the support of each subgroup $C_k$ intersects only finitely many others guarantees that the limit exists and that $\prod_{k \in \mathbb{Z}} C_k$ is a subgroup of $C$. The subgroup is normal, but it is not necessarily closed.

The following elementary theorem provides several conditions that can be used to verify that a code $C$ is closed.

**Theorem 3.1:** A group code is $C$ closed if

1. $C$ has finite support;
2. $C$ is a product of closed group codes with disjoint support;
3. $C$ is an intersection of closed group codes; or,
4. $C$ is the kernel of a projection $P_J$ of a closed group code.

**Proof:** 1. Let $C$ be a group code with finite support, and let $c_0, c_1, \ldots$ be a sequence of trajectories in $C$ that converges pointwise to $c$. For any $i \geq 0$, there exists an $N$ such that $(c_k)_{[-i,i]} = c_{[-i,i]}$ for $k \geq N$. Pick $i$ so that the interval $[-i,i]$ contains the support of $C$. Then $c_k = c$ for $k \geq N$. The limit $c$ is therefore in $C$.

2. Let $C = C_1 C_2 \cdots$ be a product of group codes $C_1, C_2, \ldots$ with disjoint support $J_1, J_2, \ldots$, and let $c_0, c_1, \ldots$ be a sequence of trajectories in $C$ that converges pointwise to $c$. Then $P_{J_i}(c_0), P_{J_i}(c_1), \ldots$ converges pointwise in $C_i$ to $P_{J_i}(c)$, hence $c = P_{J_1}(c) P_{J_2}(c) \cdots$ is a sequence in $C$.

3. Intersections of closed sets are closed.

4. The kernel of $P_J : C \to W$ is closed because it is the intersection of the closed sets $C$ and $P \setminus J(W)$. □
CHAPTER 3. GROUP SYSTEMS

The theorem implies, for example, that if \( C \) is closed, then the products \( C_k^{-}C_k^{+} \) and \( \prod_{k \in I} C_{[k,k+1]} \) are closed for \( k \in I \). However, products of closed subgroups are not guaranteed to be closed in general: even if \( C \) is closed, \( C_{k+1}^{-}C_k^{+} \) or \( \prod_{k \in I} C_{[k,k+1]} \) may not be. Both of these counterexamples require infinite output groups.

Kitchens [34] shows that closed time-invariant group codes over finite groups must in fact be strongly observable ("finite type" in the language of symbolic dynamics), and hence must be finite-state.

The definition of observability given in Section 2.3.2 becomes especially sharp when applied to group codes:

**Theorem 3.2:** If any sequence \( c \) in a group code \( C \) is \([k, k+\mu)\)-observable then all code sequences are \([k, k + \mu)\)-observable.

**Proof:** A sequence \( c \) is \([k, k+\mu)\)-observable if for all \( b \in C \) that agree with \( c \) on \([k, k+\mu)\) the sequence \( P_{k^-}(c)P_{k^+}(b) \) is in \( C \). Pick any \( c' \in C \). If \( d' \) agrees with \( c' \) on \([k, k+\mu)\) then \( d'c'^{-1}c \) agrees with \( c \) on \([k, k+\nu)\), which in turn implies that \( P_{k^-}(c)P_{k^+}(d'c'^{-1}c) \) is in \( C \). Multiplying on the left by \( c^{-1}_-c' \) proves that \( P_{k^-}(c')P_{k^+}(d') \) is in \( C \).

Thus, to prove that a group code is \( \mu \)-observable, it suffices to show that the zero sequence \( 0 \) is \([k, k+\mu)\)-observable but not \([k, k+\mu-1)\)-observable for all \( k \in I \).

The definition of controllability is also strengthened when applied to group codes. The following theorem shows that strongly controllable codes can be described in terms of their finite-length sequences.

**Theorem 3.3 (Strong controllability theorem):** Let \( C \) be a group code. If \( C \) is \( \nu \)-controllable then \( C \) is contained in the set \( \prod C_{[k,k+\nu]} \) of products of sequences of length \( \nu + 1 \) or less, with equality if \( C \) is closed. Conversely, if \( C = \prod C_{[k,k+\nu]} \) then \( C \) is strongly controllable with controllability index at most \( \nu \).

**Proof:** If \( C \) is \( \nu \)-controllable, then for any time \( k \in I \) and any two sequences \( c, c' \in C \), there exists a sequence \( c'' \in C \) that agrees with \( c \) on the past \( k^- \) and with \( c' \) on the future \( (k+\nu)^+ \).

We first show that \( C_k^{+} = C_{[k,k+\nu]}C_{(k+1)^+} \). Clearly \( C_{[k,k+\nu]}C_{(k+1)^+} \subseteq C_k^{+} \). To prove the reverse inclusion, for any \( c \in C_k^{+} \) use \( \nu \)-controllability to find
3.1. PRELIMINARIES

\( b \in C \) that agrees with \( c \) on \((k + 1)^-\) and with \( 0 \) on \((k + \nu + 1)^+\), and let \( a = b^{-1}c \). Then \( b \in C_{[k,k+\nu]} \), \( a \in C_{(k+1)^+} \), and \( c = ba \).

By induction, \( C_{k^+} = (\prod_{i=k}^j C_{[i,i+\nu]}C_{(j+1)^+}) \) for \( j > k \), hence any \( c \in C_{k^+} \) can be expressed as a limit

\[
c = \lim_{i \to \infty} c_k c_{k+1} \cdots c_{i-1} c_i
\]

where \( c_i \in C_{[i,i+\nu]} \). We conclude that

\[
C_{k^+} \subseteq \prod_{i=k}^\infty C_{[i,i+\nu]},
\]

with equality if \( C_{k^+} \) is closed. A symmetric argument shows that

\[
C_{k^-} \subseteq \prod_{i=-\infty}^{k-\nu-1} C_{[i,i+\nu]}.
\]

We next show for all \( k \in I \) that \( C = C_{(k+\nu)^-}C_{k^+} \). For any \( c \in C \), let \( b \in C \) be a sequence that agrees with \( c \) on \( k^- \) and with \( 0 \) on \((k+\nu)^+\), and let \( a = cb^{-1} \). Then \( a \in C_{k^+} \), \( b \in C_{(k+\nu)^-} \), and \( c = ba \). Thus \( C \subseteq C_{(k+\nu)^-}C_{k^+} \). The reverse inclusion is obvious.

Combining this result with (3.8) and (3.9) yields

\[
C = C_{(k+\nu)^-}C_{k^+}
\]

\[
\subseteq (\prod_{i=-\infty}^{k-\nu-1} C_{[i,i+\nu]})(\prod_{i=k}^\infty C_{[i,i+\nu]})
\]

\[
\subseteq \prod_{i=-\infty}^\infty C_{[i,i+\nu]},
\]

with equality if \( C \) is closed. Every sequence in \( C \) is therefore a product of sequences of length \( \nu + 1 \) or less.

Conversely, assume that \( C \) consists of the products of sequences of length \( \nu + 1 \) or less. We must show for all \( i \in Z \) and \( c, c' \in C \) that \( c \) is \([i,i+\nu])-\) controllable to \( c' \). Let \( c = \prod_{k \in Z} c_k \) and \( c' = \prod_{k \in Z} c'_k \) be decompositions of \( c \) and \( c' \) into products of length-(\( \nu + 1 \)) sequences, where \( c_k, c'_k \in C_{[k,k+\nu]} \) for \( k \in Z \). Set \( d = (\prod_{k<i} c_k)(\prod_{k>i} c'_k) \). Then \( d \in C \) agrees with \( c \) on the past \( k^- \) and with \( c' \) on the future \((k+\nu)^+\).

The decomposition of code sequences into a product finite-length sequences is central to the minimal encoder construction given in Section 3.4.
Example 3.5: The code $C$ in Example 3.3 is defined as the set of sums of a length-2 generator sequence $g$. The controllability index of $C$ is therefore at most 1. The code is not memoryless, hence the the controllability index is exactly 1. □

3.2 The state group

The state group $\Sigma_k$ of a group code $C$ at time $k$ arises from a partition of the time axis $I$ into the future $k^+$ and the past $k^-$. More generally, a state group of $C$ is induced by any two-way partition of $I$ into complementary subsets.

We define the state group $\Sigma_k$ as a quotient group, and show that it has the properties expected of state. In particular, states are minimally sufficient statistics for the future given the past (or vice versa). Any minimal state-output realization for $C$ must have a state space at time $k$ that corresponds to $\Sigma_k$, and all such realizations have essentially the same trellis (state-transition) diagram.

3.2.1 Definition and fundamental properties

The state group of a group code $C$ is the set of equivalence classes that appear as the states of the past-induced and future-induced realizations of $C$. We show that these realizations are equal.

Recall from Section 2.4.3 that two trajectories $c, c' \in C$ are past equivalent at time $k$ if they have the same set of possible future continuations. Future equivalence is defined similarly, with the roles of the past and future reversed. The next two lemmas develop a simple group-theoretic characterization of past and future equivalence.

Lemma 3.4: The set of sequences in a group code $C$ that agree with any $c \in C$ on the past $k^-$ is $cC_{k^+}$. The set of sequences that agree with $c$ on the future $k^+$ is $Cc_{k^-}$.

Proof: The set of sequences that are zero on the past $k^-$ is $C_{k^-}$, while the set of sequences that are zero on the future $k^+$ is $C_{k^+}$, as shown in Figure 3.5. Clearly every sequence in $cC_{k^+}$ agrees with $c$ on the past. Conversely, if $c' \in C$ agrees with $c$ on the past, then $c^{-1}c' \in C$ is zero on the past, hence $c' = c(c^{-1}c')$ is the product of $c$ with a sequence in $C_{k^+}$. A symmetric argument proves the result for $k^+$. □
3.2. THE STATE GROUP

Figure 3.5: The subcodes $C_k^-$ and $C_k^+$.

Lemma 3.5: Two sequences $c, c'$ in a group code $C$ are past equivalent at time $k$ if and only if $P_{k^+}(c)C_{k^+} = P_{k^+}(c')C_{k^+}$. They are future equivalent at time $k$ if and only if $P_{k^-}(c)C_{k^-} = P_{k^-}(c')C_{k^-}$.

Proof: Two sequences are past equivalent if they have the same set of possible futures. From the previous lemma, the set of futures of $c$ is $P_{k^+}(cC_{k^+}) = P_{k^+}(c)C_{k^+}$ and the set of futures of $c'$ is $P_{k^+}(c'C_{k^+}) = P_{k^+}(c')C_{k^+}$, hence $c$ and $c'$ are past equivalent if and only if $P_{k^+}(c)C_{k^+} = P_{k^+}(c')C_{k^+}$. The proof for future equivalence is analogous. ■

The previous lemma shows that the states at time $k$ of the past-induced realization correspond to the cosets of $C_{k^+}$ in $P_{k^+}(C)$, and that the states at time $k$ of the future-induced realization correspond to the cosets of $C_{k^-}$ in $P_{k^-}(C)$. Two sequences are in the same past-induced state if and only if their futures are in the same coset of $C_{k^+}$ in $P_{k^+}(C)$, and they are in the same future-induced state if and only if their pasts are in the same coset of $C_{k^-}$ in $P_{k^-}(C)$.

The main result of this section is that past-induced states and the future-induced states are the same.

Theorem 3.6 (State space theorem): The set of sequences in a group code $C$ that are past equivalent or future equivalent at time $k$ to $c \in C$ is $cC_{k^-}C_{k^+}$. Two sequences are therefore past equivalent if and only if they are future equivalent.

Proof: As shown in Section 3.1.2, the chain $C_{k^+} \subseteq C_{k^-}C_{k^+} \subseteq C$ is a normal series. The kernel of the projection $P_{k^+}$ acting on each code in this series is the subcode $C_{k^-}$. By the correspondence theorem, the image $\{0\} \subseteq C_{k^+} \subseteq P_{k^+}(C)$ of this series under $P_{k^+}$ is a normal series, and corresponding quotient groups are isomorphic. In particular,

$$P_{k^+}(C)/C_{k^+} \cong C/(C_{k^-}C_{k^+}).$$

(3.13)
The isomorphism is defined by the projection $P_{k^+}$: given two sequences $c, c' \in C$, the cosets $cC_k - C_{k^+}$ and $c'C_k - C_{k^+}$ are equal if and only if their projections $P_{k^+}(cC_k - C_{k^+}) = P_{k^+}(c)C_{k^+}$ and $P_{k^+}(c'C_k - C_{k^+}) = P_{k^+}(c')C_{k^+}$ are equal. By the previous lemma, these projections are equal if and only if $c$ and $c'$ are past equivalent. Two sequences are therefore past equivalent if and only if they are in the same coset of $C_k - C_{k^+}$ in $C$, i.e., if $cC_k - C_{k^+} = c'C_k - C_{k^+}$. The proof for future equivalence is symmetric.

Willems [67, Prop. 2.4(b)] shows that past equivalence is equivalent to future equivalence in linear dynamical systems. Although his definition of linearity is in terms of real vector spaces, his proof is essentially group-theoretic, as here.

The state space theorem implies that the concatenation $c \cdot c'$ of the past of a code sequence $c$ with the future of a code sequence $c'$ is in $C$ if and only if $c$ and $c'$ are in the same coset of $C_k - C_{k^+}$ in $C$. This is true if and only if $P_k^-(c)$ and $P_k^-(c')$ are in the same coset of $C_k^-$ in $P_k^-(C)$, or equivalently if and only if $P_k^+(c)$ and $P_k^+(c')$ are in the same coset of $C_{k^+}$ in $P_k^+(C)$.

Thus, given a past projection $P_k^-(c)$, all information relevant to predicting the future is captured in the future-induced state $P_k^-(c)C_{k^+}$. This is the essence of the concept of "state." In statistical terms, the coset $P_k^-(c)C_{k^-}$ is a sufficient statistic for the past with respect to any estimate of the future. Equally, given any future projection $P_k^+(c)$, the past-induced state $P_k^+(c)C_{k^+}$ captures all information relevant to retrodicting the past.

The zero state of a group code $C$ at time $k$ is the set of sequences that are past-equivalent (and future-equivalent) to the zero sequence $0$. According to the state space theorem, the zero state of $C$ is the normal subcode $C_k - C_{k^+}$. The zero state consists of all concatenations of pasts in $C_k^-$ with futures in $C_{k^+}$; concatenation and product are the same operation in this case because the supports of $C_k^-$ and $C_{k^+}$ are disjoint. The zero state is shown in Figure 3.6.

The equivalence classes of sequences that define the states of the the past- and future-induced realization at time $k$ are the cosets of the zero state $C_k - C_{k^+}$ in $C$. We therefore make the following definition.

**Definition 3.4:** The state group $\Sigma_k$ of a group code $C$ at time $k$ is the quotient group

$$\Sigma_k = C/(C_k - C_{k^+}).$$  \hspace{1cm} (3.14)

Similar definitions of the state of a linear dynamical system have been introduced by Willems in [66, Thm. 9], in the context of linear systems over
3.2. THE STATE GROUP

Figure 3.6: The zero state $C_k^{-} - C_k^{+}$.

vector spaces; by Forney in [17, Appendix], in the context of determining the
trellis diagrams of block codes and lattices; and by Mittelholzer and Massey
in [47, 46], in the context of determining minimal encoders for convolutional
codes over rings. It would be surprising if the literature of algebraic linear
system theory did not contain further similar definitions.

Example 3.6: Let $C$ be the group code over $\mathbb{Z}_4$ defined on $I = [1, 2]$ that
consists of the eight length-2 sequences

$$C = \{(0, 0), (0, 2), (1, 1), (1, 3), (2, 0), (2, 2), (3, 1), (3, 3)\}. \quad (3.15)$$

We wish to find the state group of $C$ at time $k = 2$. The subcode $C_2^{+}$ consists
of the sequences that are 0 at time $k = 1$, namely $C_2^{+} = \{(0, 0), (0, 2)\}$. The
subcode $C_2^{-}$ consists of the sequences that are 0 at time $k = 2$, namely $C_2^{-} = \{(0, 0), (2, 0)\}$. The zero state $C_2^{-} - C_2^{+}$ is therefore $\{(0, 0), (0, 2), (2, 0), (2, 2)\}$. The
only other coset of $C_2^{-} - C_2^{+}$ in $C$ is

$$C_2^{-} - C_2^{+} + (1, 1) = \{(1, 1), (1, 3), (3, 1), (3, 3)\}. \quad (3.16)$$

The state group $\Sigma_2 = C/C_2^{-} - C_2^{+}$ contains two cosets, hence it is isomorphic
to $\mathbb{Z}_2$. $\square$

The state group at time $k$ is defined by a partition of the time axis $I$ into
the past $k^{-}$ and the future $k^{+}$. More generally, any partition of $I$ into a subset
$J$ and a complementary subset $I \setminus J$ yields a state group of $C$. The subcodes
$C_J$ and $C_{I \setminus J}$ are normal subgroups of $C$, and the product subcode $C_J C_{I \setminus J}$
is an internal direct product which is also normal in $C$. The quotient group
$C/(C_J C_{I \setminus J})$ is therefore well defined. The preceding results carry through
essentially unchanged, with the subset $J$ playing the role of the past and
$I \setminus J$ playing the role of the future. We therefore identify the quotient group
$C/(C_J C_{I \setminus J})$ as the state group of $C$ induced by the partition $\{J, I \setminus J\}$. 
3.2.2 The canonical minimal realization

Every code sequence \( c \in C \) belongs to, or passes through, a definite state \( cc_k-C_{k+} \in \Sigma_k \) at each time \( k \). The state map \( \sigma_k: C \to C/(C_k-C_{k+}) \) is the homomorphism (the natural map) that sends a sequence \( c \in C \) to its state \( cc_k-C_{k+} \) at time \( k \). Thus, every code sequence \( c \) has an associated state sequence \( \sigma(c) \), where \( \sigma = (\sigma_k, k \in I) \) is the state sequence map. This pairing of code sequences and state sequences defines the canonical realization of a group code \( C \).

As defined in Section 2.4, a state-output realization is a set of pairs \( (s, c) \) of state sequences \( s \in \Pi_{k \in I} S_k \) and code sequences \( c \in \Pi_{k \in I} G_k \). The state space theorem shows that the past-induced and future-induced realizations of a group code are the same, hence, by Theorem 2.3, the realization is minimal and essentially unique.

**Theorem 3.7 (Minimal realization theorem):** Given a group code \( C \), the state-output realization defined by the set of trajectories \( B = \{(\sigma(c), c) \in C\} \) is a minimal realization of \( C \). Moreover, all minimal realizations of \( C \) are equivalent to this canonical state-output realization, up to state set relabeling.

Loeliger and Mittelholzer [40] develop an equivalent result for time-invariant group codes.

The state map \( \sigma \) from code sequences to state sequences is a homomorphism from \( C \) to the direct product group \( \Pi_{k \in I} \Sigma_k \), hence the canonical realization of a group code \( C \) is itself a group code. Thus, if a code sequence \( c \) passes through state sequence \( s \), and if \( c' \) passes through \( s' \), then \( cc' \) passes through \( ss' \). The algebraic structure of the state groups thus meshes naturally with the algebraic structure of the code.

**Example 3.7:** The canonical realization of the system described in Example 3.6 is given in Figure 3.7. The realization has two states at time 2, one for each coset of \( C_2-C_{2+} \) in \( C \).

3.2.3 Minimal trellis diagrams

Trellis diagrams are defined in Section 2.4.2. Examples are shown in Figures 3.1–3.4. The trellis sections of the canonical realization of a group code \( C \) have a natural group structure induced by the state groups of \( C \).
Figure 3.7: Canonical realization of a length-2 code.

The trellis section \( T_{k,k+1} \) of \( C \) at time \( k \) is constructed from the state spaces \( \Sigma_k \) and \( \Sigma_{k+1} \) and the output group \( A_k \cong P_{[k,k]}(C) \). There is a trellis branch (state transition) from \( s_k \in \Sigma_k \) to \( s_{k+1} \in \Sigma_{k+1} \) if there is a code sequence \( c \) such that \( \sigma_k(c) = s_k \) and \( \sigma_{k+1}(c) = s_{k+1} \); the associated output symbol is then \( c_k \). Since \( 0 \in C \), there must always be a branch connecting the zero state of \( \Sigma_k \) to the zero state of \( \Sigma_{k+1} \).

Algebraically, \( T_{k,k+1} \) is the subgroup \( \{(\sigma_k(c), c_k, \sigma_{k+1}(c)) : c \in C \} \) of the direct product group \( \Sigma_k \times A_k \times \Sigma_{k+1} \). The set of sequences of \( C \) that pass through the zero states of \( \Sigma_k \) and \( \Sigma_{k+1} \) via the zero symbol \( c_k = 0 \) is \( C_k \cap C_{k+1} \), hence by the fundamental homomorphism theorem

\[
T_{k,k+1} \cong C/(C_k \cap C_{k+1}). \tag{3.17}
\]

The reduced branch group \( \Sigma_{k,k+1} \) of \( C \) is the subgroup \( \{(\sigma_k(c), \sigma_{k+1}(c)) : c \in C \} \) of \( \Sigma_k \times \Sigma_{k+1} \) that consists of state pairs that are connected by branches in \( T_{k,k+1} \). The set of sequences that pass through the zero states of both \( \Sigma_k \) and \( \Sigma_{k+1} \) is \( C_k \cap C_{k+1} = C_k \cap [k,k][C_{[k,k]}C_{k+1}] \), hence

\[
\Sigma_{k,k+1} \cong C/(C_k \cap [k,k][C_{k+1}]). \tag{3.18}
\]

Two branches \( (s_k, c_k, s_{k+1}) \) and \( (s'_{k}, c'_{k}, s'_{k+1}) \) in \( T_{k,k+1} \) are parallel if \( s_k = s'_k \) and \( s_{k+1} = s'_{k+1} \). Parallel branches connect that same pair of states. The parallel transition subgroup at time \( k \) is the set of output symbols that appear on branches from the zero state at time \( k \) to the zero state at time \( k + 1 \). Specifically, the parallel transition subgroup at time \( k \) is the projection

\[
P_{[k,k]}(C_k \cap [k,k][C_{k+1}]) = C_{[k,k]} \). \tag{3.19}
\]

Depending on context, we may regard \( C_{[k,k]} \) as a subgroup either of \( P_{[k,k]}(C) \) or of \( A_k \).
The set of outputs associated with any pair \( (s_k, s_{k+1}) \) of connected states is a coset of \( C_{[k,k]} \) in \( P_{[k,k]}(C) \). "Factoring out" the parallel transitions from a trellis section \( T_{k,k+1} \) yields the reduced trellis section

\[
R_{k,k+1} = \{ (\sigma_k(c), P_{[k,k]}(c)C_{[k,k]}, \sigma_{k+1}(c)) : c \in C \}.
\]

The reduced trellis section \( R_{k,k+1} \) is a subgroup of \( \Sigma_k \times P_{[k,k]}(C)/C_{[k,k]} \times \Sigma_{k+1} \) that is isomorphic to the reduced branch group \( \Sigma_{k,k+1} \). The branches of \( R_{k,k+1} \) are labeled with cosets of \( C_{[k,k]} \) in \( P_{[k,k]}(C) \). A reduced trellis section therefore displays the "skeleton" of the trellis section \( T_{k,k+1} \); it is "fleshed out" by replacing each reduced branch by a set of branches labeled with elements of the appropriate coset of \( C_{[k,k]} \) in \( P_{[k,k]}(C) \).

The canonical trellis diagram for \( C \) is formed by identifying states of \( \Sigma_{k+1} \) in the trellis sections \( T_{k,k+1} \) and \( T_{k+1,k+2} \) for all \( k \in I \). The set of all paths through the canonical trellis diagram of \( C \) determines a group code that includes all sequences in \( C \). As stated in Theorem 2.4, Willems [67] has shown that if \( C \) is complete then the set of sequences determined by walks on the canonical trellis is exactly \( C \).

**Example 3.8:** (Loeliger) Let \( C_{\text{even}} \) be the time-invariant group code that consists of all finite binary sequences in \( (Z_2)^Z \) with even Hamming weight. Then \( C_{\text{even}} \) is not complete.

The state group of \( C_{\text{even}} \) at any time \( k \) is isomorphic to \( Z_2 \), since \( C_{\text{even},k-} \), the set of all finite even-weight sequences defined on \( (-\infty, k) \), is a proper subgroup of \( P_{k-}(C_{\text{even}}) \), which also includes the odd-weight sequences. The trellis diagram of \( C_{\text{even}} \) is shown in Figure 3.8. The closure of \( C_{\text{even}} \) is the group sequence space \( (Z_2)^Z \), which has a trivial one-state trellis diagram.

![Figure 3.8: The trellis diagram of \( C_{\text{even}} \).](image)

**Example 3.9:** (Loeliger) Let \( C \) be the time-invariant code that contains all periodic sequences over a group \( G \). Products and inverses of periodic sequences are periodic, hence \( C \) is a group code. It is uncontrollable because
the past of a sequence uniquely determines its future and vice versa. Systems of this type are autonomous [67]. The closure of \( C \) is the set of all sequences over \( G \).

As with any autonomous system, the canonical realization of \( C \) has one state for each code sequence. The canonical realization is therefore closed. For this example, the system realized by the closure of the canonical realization of \( C \) is not the closure of \( C \). □

Loeliger and Mittelholzer [39, 40] define group codes as the set of walks on a group trellis diagram. They arrive at essentially the same minimal realizations and trellis diagrams as do we.

### 3.3 Inputs and state codes

We have shown that, given a choice of time axis \( I \), every group code \( C \) has an essentially unique minimal realization and trellis diagram. We now examine the dynamical structure of \( C \) in greater detail.

We give a natural definition of the input groups \( F_k \) of a linear code \( C \) at each time \( k \) which leads to a minimal encoder structure based on group extensions. The underlying state-output realization of the encoder is the canonical realization of \( C \).

The dynamics of \( C \) may be characterized by the state code \( \sigma(C) \) of \( C \). The state code specifies the reduced trellis diagram of \( C \), which is found by "factoring out" parallel transitions.

For notational simplicity, we shall assume henceforth that the time axis \( I \) is \( \mathbb{Z} \). The range of the time index \( k \) in any expression is therefore understood to be \( \mathbb{Z} \).

#### 3.3.1 Inputs

Willems' definition of a dynamical system is based only on the set \( C \) of its possible output sequences. We have seen that if \( C \) is a group code then there is a natural group-theoretic definition of the states of \( C \) at each time \( k \). In this section we show that for group systems there is also a natural group-theoretic definition of inputs that leads to a canonical encoder structure.
Definition 3.5: The input group $F_k$ of a group code $C$ at time $k$ is $F_k = P_{[k,k]}(C_k^+)$. Depending on context, we may regard an input group $F_k$ as a subgroup either of $P_{[k,k]}(C)$ or of the output group $A_k$. The input group $F_k$ is the set of outputs that occur at time $k$ in code sequences that pass through the zero state $C_k - C_{k^+}$. Given a sequence $c$ in $C$, the set of possible outputs at time $k$ of sequences that pass through state $cC_k - C_{k^+}$ is the coset $P_{[k,k]}(c)F_k$ of $F_k$ in $A_k$.

The last-output group at time $k$ is defined analogously to the input group $F_k$ as $L_k = P_{[k,k]}(C_{k+1}^+)$. The last-output group becomes the input group if the ordering of the time axis is reversed.

The kernel of the projection $P_{[k,k]}$ restricted to $C_{k^+}$ is $C_{k+1}^+$, hence, by the fundamental homomorphism theorem,

$$F_k \cong C_{k^+} / C_{k+1}^+$$

for $k \in \mathbb{Z}$. The input granule theorem developed in Section 3.4 gives a stronger result.

The input group is also isomorphic to the subgroup of branches

$$\tilde{T}_{k,k+1} = \{ (\sigma_k(c), c_k, \sigma_{k+1}(c)) : c \in C_k - C_{k^+} \}$$

in the trellis section $T_{k,k+1}$ that leave the zero state at time $k$. From Equation (3.17), the trellis section of $C_k - C_{k^+}$ at time $k$ is isomorphic to $C_k - C_{k^+} / (C_k - C_{k+1}^+) \cong C_{k^+} / C_{k+1}^+$, which is isomorphic to $F_k$ by the previous paragraph.

A coset of $\tilde{T}_{k,k+1}$ in $T_{k,k+1}$ is the set of branches that leaves some state $cC_k - C_{k^+}$ at time $k$. All cosets have the same number of elements, hence every state in the canonical trellis $T_{k,k+1}$ has the same number $|F_k|$ of exiting branches. The canonical trellis is thus essentially an encoder.

We can convert the canonical trellis diagram into an encoder as follows. By the third isomorphism theorem, the quotient $T_{k,k+1} / \tilde{T}_{k,k+1}$ is isomorphic to the state group $C / (C_{k} - C^+_{k}) = \Sigma_k$. The trellis section $T_{k,k+1}$ is therefore a group extension of $\tilde{T}_{k,k+1} \cong F_k$ by $\Sigma_k$. Through this extension, a trellis branch at time $k$ is uniquely identified by a state in $\Sigma_k$ and an input in $F_k$. We arrive at the encoder structure shown in Figure 3.9.

A more careful treatment of encoders with this general structure may be found in [40]. The key observation here is that the underlying state-output realization of a minimal encoder for a group code $C$ is the canonical realization of $C$. 

58

CHAPTER 3. GROUP SYSTEMS
3.3. INPUTS AND STATE CODES

Figure 3.9: A canonical encoder for a group code \( C \) using trellis sections \( T_{k,k+1} \).

Catastrophic encoders are defined in Section 2.5.1. The relationship between minimal encoders and minimal realizations allows us to prove the following:

**Theorem 3.8:** A minimal encoder for a group code \( C \) must be noncatastrophic.

**Proof:** If \( c \) and \( c' \) are two code sequences that differ only over a finite interval \( [m,n) \), then their state sequences \( \sigma(c) \) and \( \sigma(c') \) can differ only in \( (m,n) \), since the past-induced states up to time \( m \) are equal, and so are the future-induced states at time \( n \) and beyond. In a minimal encoder for \( C \), the encoder state sequence tracks the code state sequence, so the (unique) input sequences that generate \( c \) and \( c' \) can differ only in \( [m,n) \). \( \blacksquare \)

It can also be shown that minimal encoders for strongly observable groups codes have sliding block inverses, and thus cannot be catastrophic.

### 3.3.2 The state code of \( C \)

As we have seen, a code sequence \( c \) in a group code \( C \) passes through a well-defined state sequence \( \sigma(c) = (\ldots, \sigma_{-1}(c), \sigma_0(c), \sigma_1(c), \ldots) \), where the state \( \sigma_k(c) \) in \( \Sigma_k \) is determined by the natural map

\[
\sigma_k: C \to C/(C_k-C_{k+}) = \Sigma_k.
\] (3.23)

The state sequence \( \sigma(c) \) is thus an element of the state sequence space \( \prod \Sigma_k \), and the state sequence map \( \sigma: C \to \prod \Sigma_k \) is a homomorphism that sends a code sequence \( c \) to its state sequence \( \sigma(c) \).
Definition 3.6: Let $\sigma: C \to \prod \Sigma_k$ be the state sequence map of a group code $C$. Its image $\sigma(C)$ is the state code of $C$. Its kernel $C_0$ is the parallel transition code of $C$.

The state code $\sigma(C)$ is a subgroup of the state sequence space $\prod \Sigma_k$, and is therefore itself a group code. By the fundamental homomorphism theorem, $\sigma(C)$ is isomorphic to $C/C_0$. In an earlier paper [22], $\sigma(C)$ was called the “metacode” of $C$. Loeliger and Mittelholzer[39, 40] refer to the state sequence map $\sigma$ by the suggestive name of “derivative.” By iteratively taking derivatives, they develop a layered canonical minimal encoder that is complementary in structure to the one we construct in Section 3.4.

The parallel transition code $C_0$ is the set of all sequences in $C$ that pass through the zero state at all times $k \in \mathbb{Z}$; therefore, by the definition of the zero state, $C_0 = \cap_k C_k - C_k^+$.

Theorem 3.9 (Parallel transition code theorem): The parallel transition code of a closed (complete) group code $C$ is the direct product $C_0 = \prod C_{[k,k]}$ of the parallel transition subgroups $C_{[k,k]}$.

Proof: By Theorem 3.1, both $\cap_k C_k - C_k^+$ and $\prod C_{[k,k]}$ are closed. It thus suffices to show for any finite interval $[m, n)$ that $P_{[m,n)}(C_0) = P_{[m,n)}(\prod C_k - C_k^+)$. Obviously

$$P_{[m,n)}(\prod_{k \in \mathbb{Z}} C_{[k,k]}) = \prod_{k \in [m,n)} C_{[k,k]}.$$  \hspace{1cm} (3.24)

As noted in Section 3.2.3, the intersection of $C_k - C_k^+$ and $C_{k+1} - C_{k+1}^+$ is $C_k - C_{[k,k]} C_{k+1}^+$; more generally, by induction,

$$\bigcap_{k \in [m,n)} C_k - C_k^+ = C_m - (\prod_{k \in [m,n)} C_{[k,k]} C_{n}^+),$$  \hspace{1cm} (3.25)

so $P_{[m,n)}(C_0) = \prod_{k \in [m,n)} C_{[k,k]}$ also. \hfill \Box

The parallel transition code $C_0$ is the maximal subcode of $C$ that is memoryless. We may therefore refer to $C_0$ as the nondynamical component of the code $C$, and to $C_{[k,k]}$ as the nondynamical component of the output group $A_k \cong P_{[k,k]}(C)$.

Given a code sequence $c$ in $C$, the set of all code sequences that pass through the state sequence $\sigma(c)$ is precisely the coset $Cc_0$ of the parallel transition code $C_0$. This suggests that the parallel transition subgroups $C_{[k,k]}$ may be “factored out” without affecting the dynamical structure of $C$. 
For all $k \in \mathbb{Z}$, let $r_k : P_{[k,k]}(C) \to P_{[k,k]}(C')/C_{[k,k]}$ be the natural map, and let $r : C \to \prod P_{[k,k]}(C)/C_{[k,k]}$ be defined componentwise by $r(c) = (\ldots, r_{-1}(c_{-1}), r_0(c_0), r_1(c_1), \ldots)$. Then $r$ is a homomorphism that maps $C$ to the reduced code $C' = r(C)$.

The kernel of $r$ is the parallel transition code $C_0$, hence $C' \cong C/C_0$. In group-theoretic terms, $C$ is an extension of the parallel transition code $C_0$ by $C' = r(C)$. The trellis section of $C'$ at time $k$ is thus the reduced trellis section $R_{k,k+1}$. The parallel transition code of $C'$ is trivial.

Different codes may have the same reduced code. The dynamics of the two codes are identical, but the codes are not. Two codes whose reduced codes are identical, up to componentwise isomorphism, are dynamically equivalent. For example: all closed memoryless codes defined on the same time axis are dynamically equivalent, and the convolutional code over $\mathbb{Z}_4$ of Example 3.3 is dynamically equivalent to a linear two-state rate-1/2 convolutional code over $\mathbb{Z}_2$.

### 3.4 Canonical encoders based on generators

Our goal in this section is to construct a feedforward canonical minimal encoder with memory $\nu$ for a $\nu$-controllable, complete, group code $C$. We assume that the time axis of $C$ is $\mathbb{Z}$.

Our encoder is based on a chain coset decomposition of code sequences into a product of finite-length representative sequences, or generators. The encoder forms code sequences by combining generators that are activated by input sequences. We first describe the coset decomposition, then show how to use the components of the decomposition ("granules") in the construction of a feedforward encoder, and finally prove that the resulting encoder is minimal.

The encoder construction is similar in flavor to one developed by Roos [48] for convolutional codes over fields, but the algebra is quite different. Our results reduce to Roos' as a special case.

#### 3.4.1 Granules

Our encoder construction is based on the decomposition of the code into elementary constituents, called granules. Granules are derived from a chain
coset decomposition of $C$ into quotients of its $j$-controllable subcodes, and then a further decomposition of these quotient groups along the time axis.

While the granules are initially defined as constituents of the code $C$, we shall see that they may also be taken as the elementary constituents of the input groups $F_k$ or of the state code $\sigma(C)$.

**Definition 3.7:** The $j$-controllable subcode $C_j$ of a closed group code $C$ is largest subgroup of $C$ that is $j$-controllable.

Intuitively, the $j$-controllable subcode of $C$ corresponds to the set of walks on the canonical trellis of $C$ that wander no more than $j$ steps from the 0 state.

It must be proved that the $j$-controllable subcode exists and is unique. If $C_j$ is $j$-controllable then the strong controllability theorem (see Section 3.1.3) implies that every sequence in $C_j$ is a product of code sequences of length $j + 1$ or less, hence $C_j \subseteq \prod C_{[k,k+j]}$. Conversely, $\prod C_{[k,k+j]}$ is a subcode of $C$ that is $j$-controllable, hence if $C_j$ is maximal then $\prod C_{[k,k+j]} \subseteq C_j$. We conclude that

$$C_j = \prod C_{[k,k+j]}.$$  \hfill (3.26)

The $j$-controllable subcodes of a closed code need not be closed, and their closures need not be controllable. Such cases can arise only when the output groups of $C$ are infinite, i.e., when $C$ is not compact. While our results can be adapted to such cases, we will generally assume that the $j$-controllable subcodes are closed.

A group code $C$ is $\nu$-controllable if and only if $C = C_\nu$. The $0$-controllable subcode $C_0 = \prod C_{[k,k]}$ is the parallel transition code of $C$. Clearly $C_j$ is the $j$-controllable subcode of $C_{j'}$ for any $j' \geq j$. Also, $C_j$ is normal in $C$ because it is a product of normal subcodes $C_{[k,k+j]}$, hence

$$C_0 \subseteq C_1 \subseteq \cdots \subseteq C_\nu = C$$  \hfill (3.27)

is a normal series. A chain coset decomposition yields a one-to-one correspondence

$$C \leftrightarrow C_0 \times (C_1/C_0) \times \cdots \times (C_\nu/C_{\nu-1}).$$  \hfill (3.28)

The quotient groups $C_j/C_{j-1}$, $1 \leq j \leq \nu$, may be further decomposed as follows.
Definition 3.8: For $k \in \mathbb{Z}$ and $0 \leq j \leq \nu$, the granule $\Gamma_{[k,k+j]}$ of $C$ is the quotient group

$$\Gamma_{[k,k+j]} = C_{[k,k+j]}/C_{[k,k+j]}C_{(k,k+j)}.$$  \hfill (3.29)

In particular, the nondynamical granule $\Gamma_{[k,k]}$ is $C_{[k,k]}$.

An abstract depiction of a granule is shown in Figure 3.10. The "hump" on the left side of the figure, indicated by horizontal shading, is the set of sequences $C_{[k,k+j]}$ that are nonzero only on the interval $[k, k + j - 1]$. The hump on the right, indicated by vertical shading, is the set of sequences $C_{(k,k+j)}$ that are nonzero only on the interval $[k + 1, k + j]$. The unshaded hump that spans the interval $[k, k + j]$ is $C_{[k,k+j]}$.

The granule $\Gamma_{[k,k+j]}$ represents what is needed to generate the sequences in $C_{[k,k+j]}$ that are not in $C_{j-1}$. We state this more precisely as a theorem.

**Theorem 3.10 (Code granule theorem):** If $C_j$ and $C_{j-1}$ are the $j$-controllable and $j-1$-controllable subcodes of a closed group code $C$, and if $C_j$ and $C_{j-1}$ are closed, then $C_j/C_{j-1}$ is isomorphic to the direct product $\prod \Gamma_{[k,k+j]}$.

**Proof:** From the definition of $C_j$, $C_j = C_jC_{j-1} = \prod C_{[k,k+j]}C_{j-1}$. Thus

$$C_j/C_{j-1} = \prod (C_{[k,k+j]}C_{j-1})/C_{j-1}. \hfill (3.30)$$

This is an internal direct product, since $(C_{[k,k+j]}C_{j-1}) \cap \prod_{i \neq k} C_{[i,i+j]}C_{j-1} = C_{j-1}$. By the second isomorphism theorem,

$$C_j/C_{j-1} \cong \prod C_{[k,k+j]}/(C_{j-1} \cap C_{[k,k+j]}), \hfill (3.31)$$

$$= \prod C_{[k,k+j]}/C_{[k,k+j]}C_{(k,k+j)}, \hfill (3.32)$$

which is the desired direct product of granules. \[\square\]
**Definition 3.9:** If $C$ is a closed $\nu$-controllable group code, its $j^{th}$ input group at time $k$ for $0 \leq j \leq \nu$ is the image $F_{k,j} = P_{[k,k]}(C_{[k,k+j]})$ of $C_{[k,k+j]}$ under the projection $P_{[k,k]}$. Its time-$k$ input chain is the series $F_{k,0} \subseteq F_{k,1} \subseteq \cdots \subseteq F_{k,\nu}$.

Note that $F_{k,0} = C_{[k,k]}$. Also, by the strong controllability theorem,

$$F_{k,\nu} = P_{[k,k]}(C_{k^+}) = F_k. \quad (3.33)$$

The time-$k$ input chain is a normal series, since it is the projection of the normal series $C_{[k,k]} \subseteq C_{[k,k+1]} \subseteq \cdots \subseteq C_{[k,k+\nu]}$.

**Theorem 3.11 (Input granule theorem):** If $C_{[k,k]} = F_{k,0} \subseteq F_{k,1} \subseteq \cdots \subseteq F_{k,\nu} = F_k$ is the time-$k$ input chain of a closed, $\nu$-controllable group code $C$, then

$$F_{k,j}/F_{k,j-1} \cong \Gamma_{[k,k+j]} \quad \text{(3.34)}$$

for $1 \leq j \leq \nu$.

**Proof:** By definition, $\Gamma_{[k,k+j]} = C_{[k,k+j]}/(C_{[k,k+j]}C_{(k,k+j)})$. The projection $P_{[k,k]}$ restricted to $C_{[k,k+j]}$ has kernel $C_{(k,k+j)}$ and image $P_{[k,k]}(C_{[k,k+j]}) = F_{k,j}$; the projection $P_{[k,k]}$ restricted to $C_{[k,k+j]}C_{(k,k+j)}$ has kernel $C_{(k,k+j)}$ and image $P_{[k,k]}(C_{[k,k+j]}) = F_{k,j-1}$. Therefore, by the correspondence theorem, the granules are isomorphic to the quotients of the input chain. \hfill \Box

The input group $F_k$ therefore has a chain coset decomposition into components isomorphic to the granules $\{\Gamma_{[k,k+j]} : 0 \leq j \leq \nu\}$. These granules may therefore be regarded as the elementary constituents of the input group at time $k$.

Analogously, if we define the $j^{th}$ last-output group at time $k$ as $L_{k,j} = P_{[k,k]}(C_{[k-j,k]})$ and the time-$k$ last-output chain as $L_{k,0} \subseteq L_{k,1} \subseteq \cdots \subseteq L_{k,\nu}$, then $L_0 = C_{[k,k]}$, $L_{k,\nu} = L_k$, and

$$L_{k,j}/L_{k,j-1} \cong \Gamma_{[k-j,k]} \quad \text{(3.35)}$$

for $1 \leq j \leq \nu$.

Let $\sigma(C_j)$ be the state code of the $j$-controllable subcode $C_j$. The image $\sigma_k(C_j)$ of the state map at time $k$ is the $j$-controllable state group of $C$ at time $k$. The $j$-controllable state groups and state codes form normal series:

$$\{0\} = \sigma_k(C_0) \subseteq \sigma_k(C_1) \subseteq \cdots \subseteq \sigma_k(C_{\nu}) = \sigma_k(C) = \Sigma_k; \quad \text{(3.36)}$$

$$\{0\} = \sigma(C_0) \subseteq \sigma(C_1) \subseteq \cdots \subseteq \sigma(C_{\nu}) = \sigma(C). \quad \text{(3.37)}$$
3.4. **CANONICAL ENCODERS BASED ON GENERATORS**

As with the quotients of the \( j \)-controllable subcodes \( C_j \), the quotients of the \( j \)-controllable state groups \( \sigma_k(C_j) \) are isomorphic to direct products of granules:

**Theorem 3.12 (State granule theorem):** If \( \sigma_k(C_j) \) is the \( j \)-controllable state group of a group code \( C \) then

\[
\sigma_k(C_j)/\sigma_k(C_{j-1}) \cong \prod_{i \in [k-j-1]} \Gamma_{[i,i+j]}
\]  

(3.38)

for \( k \in \mathbb{Z} \) and \( j \geq 1 \).

**Proof:** The kernel of the state map \( \sigma_k: C \rightarrow \Sigma_k \) is \( C_kC_k^+ \), and \( C_j \) and \( C_{j-1} \) are normal subgroups of \( C \). The image of \( C_jC_kC_k^+ \) under \( \sigma_k \) is

\[
\sigma_k(C_jC_kC_k^+) = \sigma_k(C_j)\sigma_k(C_k-C_k^+) = \sigma_k(C_j),
\]

(3.39)

and similarly the image of \( C_{j-1}C_k-C_k^+ \) under \( \sigma_k \) is \( \sigma_k(C_{j-1}) \). Now if

\[
C_{j,[k-j,k+j]} = \prod_{i \in [k-j,k-1]} C_{[i,i+j]}
\]

(3.40)

is the subcode of \( C_j \) whose support is \([k-j,k+j]\), then

\[
C_{j,[k-j,k+j]}(C_{j-1}C_kC_k^+) = C_jC_kC_k^+,
\]

(3.41)

since the elements of \( C_j \) that are not in \( C_{j,[k-j,k+j]} \) are elements of the product \( \prod_{i \in [k-j,k-1]} C_{[i,i+j]} \subseteq C_k-C_k^+ \). The intersection of \( C_{j,[k-j,k+j]} \) and \( C_{j-1}C_k-C_k^+ \) is

\[
C_{j,[k-j,k+j]} \cap (C_{j-1}C_k-C_k^+) = C_{j,[k-j,k+j]} \cap C_{j-1} = C_{j-1,[k-j,k+j]}.
\]

(3.42)

Hence, by the homomorphism/normal subgroup theorem (Appendix A),

\[
\sigma_k(C_j)/\sigma_k(C_{j-1}) \cong (C_jC_k-C_k^+)/(C_{j-1}C_k-C_k^+)
\]

(3.43)

\[
\cong C_{j,[k-j,k+j]}/C_{j-1,[k-j,k+j]}.
\]

(3.44)

Finally, the code granule theorem applied to \( C_{j,[k-j,k+j]} \) shows that

\[
C_{j,[k-j,k+j]}/C_{j-1,[k-j,k+j]} \cong \prod_{i \in [k-j,k-1]} \Gamma_{[i,i+j]}.
\]

(3.45)
The state granule theorem thus defines a decomposition of the state group \( \Sigma_k \) of a \( \nu \)-controllable code into a Cartesian product of direct products of granules:

\[
\Sigma_k \leftrightarrow \bigotimes_{1 \leq j \leq \nu, i \in [k-j, k-1]} \Gamma_{[i,i+j]}.
\] (3.46)

The coset decomposition of the state group is not necessarily direct product, hence the state group cannot by recovered from knowledge of the granules up to isomorphism alone.

From this decomposition, we can determine the size of the state groups of \( C \).

**Theorem 3.13 (State group size theorem):** If \( C \) is a \( \nu \)-controllable group code with time-\( k \) input chain \( F_{k,0} \subseteq F_{k,1} \subseteq \cdots F_{k,\nu} = F_k \), and the granules \( \Gamma_{[k,k+j]} \) of \( C \) are finite for \( k \in \mathbb{Z} \) and \( 1 \leq j \leq \nu \), then the size of the state group \( \Sigma_k \) of \( C \) at time \( k \) is

\[
|\Sigma_k| = \prod_{1 \leq j \leq \nu} \prod_{i \in [k-j,k-1]} |\Gamma_{[i,i+j]}| \quad (3.47)
\]

\[
= \prod_{1 \leq j \leq \nu} \prod_{i \in [k-j,k-1]} |F_{i,j}/F_{i,j-1}|. \quad (3.48)
\]

If \( C \) is time invariant, with \( F_0 \subseteq F_1 \subseteq \cdots F_\nu \) as the common input chain, then

\[
|\Sigma_k| = \prod_{1 \leq j \leq \nu} |F_j/F_{j-1}|^j. \quad (3.49)
\]

### 3.4.2 Encoder construction

We construct a minimal encoder for a closed, \( \nu \)-controllable code \( C \) from the unique decomposition of sequences \( c \in C \) into products of minimal-length generators.

A set of **minimal-length generators** for \( C \) is a set of coset representatives for the granules \( \Gamma_{[k,k+j]} = C_{[k,k+j]}/(C_{[k,k+j]}C_{(k,k+j)}) \) for \( 0 \leq j \leq \nu \) and \( k \in \mathbb{Z} \). We denote the minimal-length generators by

\[
[\Gamma_{[k,k+j]}] = \{ g(\gamma_{k,j}) : \gamma_{k,j} \in \Gamma_{[k,k+j]} \}, \quad (3.50)
\]

where \( g(0) \) is always taken as \( 0 \). (Minimal-length generators were introduced, without definition, as **trellis-oriented generators** by Forney [17, Appendix].)

A minimal-length generator for \( C \) cannot be expressed as a product of shorter code sequences. Also, since a nonzero generator \( g(\gamma_{k,j}) \) is an element of \( C_{[k,k+j]} \) but not of \( C_{[k,k+j]} \) or of \( C_{(k,k+j)} \), its length must be precisely \( j+1 \).
As shown in the code granule theorem, the set $\prod [\Gamma_{[k,k+j]}]$ serves as a set of coset representatives for the cosets of $C_{j-1}$ in $C_j$. Any code sequence in $C_j$, for $1 \leq j \leq \nu$, may be uniquely written as a product of a sequence in $C_{j-1}$ with a product of generators in $\{[\Gamma_{[k,k+j]}] : k \in \mathbb{Z} \}$. Thus, given a selection of minimal-length generators, any sequence in $C = C_\nu$ can be uniquely decomposed into a product of generators:

**Theorem 3.14 (Minimal-length generator theorem):** Let $C$ be a closed, $\nu$-controllable group code with closed $j$-controllable subcodes. Then there is one-to-one correspondence between the sequences of $C$ and the set of products of minimal-length generators. The correspondence is given by $c = \prod_{0 \leq j \leq \nu} \prod_{k \in \mathbb{Z}} g(\gamma_{k,j})$, where $\{\gamma_{k,j} \in \Gamma_{[k,j]} : 0 \leq j \leq \nu, k \in \mathbb{Z} \}$ is the unique coset decomposition of $c \in C$ into granules.

A minimal encoder is constructed from a set of minimal-length generators as follows. The input at time $k$ is an element of the input group $F_k$. The input is decomposed according to the input chain into an element $\gamma_{k,0} \in \Gamma_{[k,k]}$ of the parallel transition subgroup $C_{[k,k]} = \Gamma_{[k,k]}$ and input granules $\gamma_{k,j} \in \Gamma_{[k,k+j]}$ for $1 \leq j \leq \nu$. An input granule $\gamma_{k,j}$ is stored for $j$ time units in a shift register of length $j$, where it serves as a state granule. The output at time $k$ is the product of the components $g_k(\gamma_{i,j})$ for $0 \leq j \leq \nu$ and $k - j \leq i \leq k$. The encoder structure is illustrated in Figure 3.11.

![Figure 3.11: Minimal encoder in controller canonical form.](image)

The encoder is feedforward with memory $\nu$, since each granule $\gamma_{k,j}$ is
saved in a shift register for \( j \leq \nu \) time units and then discarded. Such an encoder is said to be in controller canonical form.

The encoder is minimal by construction, because the contents of the shift registers at time \( k \) are the coset decomposition of the state group \( \Sigma_k \) given by the state granule theorem. A granule \( \Gamma_{[k,k+j]} \) is "active" during the interval \([k, k + j]\). It is stored in a shift register and contributes to the state of the encoder during the interval \([k + 1, k + j]\). The nondynamical granule \( \Gamma_{[k,k]} \) is never part of the state.

The trellis diagram of any minimal encoder for a group code \( C \), ignoring input labeling, is the canonical minimal trellis of \( C \). Thus the granules \( \Gamma_{[k,k+j]} \) for \( 1 \leq j \leq \nu \) determine the trellis diagram of \( C \). Ignoring both input and output labeling, the minimal encoder construction proves that the canonical trellis of a group code \( C \) is a shift register graph. We shall say more about this in Section 5.4.2.

The encoder is layered, in the sense that an encoder constructed from the bottom \( j \) rows of shift registers in Figure 3.11 is a minimal encoder for \( C_j \). More concretely, if the input sequence is constrained so that the granules \( \gamma_{k,j'} \in \Gamma_{[k,k+j']} \) are zero for \( j + 1 \leq j' \leq \nu \) and \( k \in \mathbb{Z} \), then the resulting encoder is a minimal encoder for the \( j \)-controllable subcode \( C_j \) of \( C \).

An encoder for a code that is controllable but not strongly controllable has a similar minimal encoder, but with an infinite set of shift registers of unbounded length. An encoder for an uncontrollable code will have in addition an eternal state space \( \Sigma_{unc} = C/C_{cont} \), where \( C_{cont} \) is the controllable subcode of \( C \). The code \( C \) will then consist of the union of \( |\Sigma_{unc}| \) cosets of \( C_{cont} \), none reachable from any other. Kitchens [34, Thm. 1(iv)] proves a stronger result for time-invariant codes over finite groups.

Since \( \Gamma_{[k,k+j]} \cong F_{k,j}/F_{k,j-1} \) under the projection \( P_{[k,k]} \), as shown in the proof of the input granule theorem, it is always possible to take the first outputs \( g_k(\gamma_{k,j}) \) of the generators \( \{g(\gamma_{k,j}) : \gamma_{k,j} \in \Gamma_{[k,k+j]}\} \) as a set of coset representatives \( [F_{k,j}/F_{k,j-1}] \) for the cosets of \( F_{k,j-1} \) in \( F_{k,j} \). Then if the encoder is in the zero state, its time-\( k \) output in response to an input \( f_k \in F_k \) is \( c_k = f_k \). Such an encoder is monic; the first nonzero output of the encoder is the first nonzero input.

**Example 3.10:** The nontrivial granules of the linear time-invariant binary convolutional code \( C \) of Figure 3.1 are simply the two-element subcodes \( C_{[k,k+2]} \) generated by time shifts \( D^k g \) of its length-3 generator \( g \):

\[
\Gamma_{[k,k+2]} = C_{[k,k+2]} = \{0, D^k g\}
\]
for \( k \in \mathbb{Z} \). This yields the linear, monic, time-invariant, minimal encoder of Figure 3.1. The code \( C \) is 2-controllable; its 0- and 1-controllable subcodes are trivial, \( C_0 = C_1 = \{0\} \). □

**Example 3.11:** The four nontrivial granules of the linear binary block code \( C \) of Figure 3.2 are each two-element groups, with representatives as follows:

\[
\begin{align*}
\Gamma_{[1,2]} &= C_{[1,2]} = \{0, (11, 11, 00, 00)\}; \\
\Gamma_{[2,3]} &= C_{[2,3]} = \{0, (00, 11, 11, 00)\}; \\
\Gamma_{[3,4]} &= C_{[3,4]} = \{0, (00, 00, 11, 11)\}; \\
\Gamma_{[1,4]} &= C_{[1,4]} / (C_{[1,2]} C_{[2,3]} C_{[3,4]}); \quad [\Gamma_{[1,4]}] = \{0, (10, 10, 10, 10)\}.
\end{align*}
\]

This yields the trellis diagram of Figure 3.2. The 0-controllable subcode of \( C \) is trivial, \( C_0 = \{0\} \); its 1-controllable (and 2-controllable) subcode is \( C_1 = C_{[1,2]} C_{[2,3]} C_{[3,4]} \), which is represented by the top two-state subtrellis in Figure 3.2. The code is trivially 3-controllable because \( C_3 = C_{[1,4]} = C \). □

**Example 3.12:** Consider again the time-invariant group code over \((\mathbb{Z}_4)^2\) generated by the linear time-invariant encoder of Figure 3.3. It is obvious from the form of the encoder that the controllability index of the code is \( \nu = 1 \). For any \( k \in \mathbb{Z} \), its input group \( F_k = P_{[k,k]}(C_k+) \) is the subgroup \( \{00, 11, 22, 33\} \) of \((\mathbb{Z}_4)^2\), which is isomorphic to \( \mathbb{Z}_4 \). The shortest non-zero code sequences are the time shifts of the parallel transition subgroup \( F_{k,0} = C_{[k,k]} = \Gamma_{[k,k]} \), which is \( \{00, 22\} \). \( F_{k,0} \) is isomorphic to \( 2\mathbb{Z}_4 \cong \mathbb{Z}_2 \). The subcode \( C_{[k,k+1]} \) has eight elements, including the four sequences in \( C_{[k,k+1]} C_{(k,k+1)} = C_{[k,k]} C_{[k+1,k+1]} \). The granule

\[
\Gamma_{[k,k+1]} = C_{[k,k+1]} / (C_{[k,k+1]} C_{(k,k+1)}) \cong F_{k,1} / F_{k,0}
\]

is therefore isomorphic to \( \mathbb{Z}_2 \), and the input chain \( F_{k,0} \subseteq F_{k,1} = F_k \) is isomorphic to \( 2\mathbb{Z}_4 \subseteq \mathbb{Z}_4 \); so \( |\Sigma_k| = 2 \) for all \( k \in \mathbb{Z} \). We may take \( \{00, 11\} \) as the coset representatives of the cosets of \( F_{k,0} = \{00, 22\} \) in \( F_k = \{00, 11, 22, 33\} \), and then take

\[
\begin{align*}
g(00) &= (\ldots, 00, 00, 00, 00, \ldots); \quad (3.53) \\
g(11) &= (\ldots, 00, 11, 13, 00, \ldots), \quad (3.54)
\end{align*}
\]

as the corresponding minimal-length generators. This yields the nonlinear, monic, time-invariant minimal encoder of Figure 3.4. □
3.4.3 Linear codes over fields

Loeliger and Mittelholzer [39, 40] give a parallel development of minimal encoders for linear time-invariant codes over groups, rings and fields. This development shows naturally how their encoder reduces to the known canonical minimal encoder [13] for linear convolutional codes over fields. So does the encoder constructed in the previous sections, as can be shown by a similar development.

Let $C$ be a linear rate-$k/n$ time-invariant convolutional code over a finite field $F_q = \text{GF}(q)$. The output space is then the vector space $V = (F_q)^n$. The order $q$ of $F_q$ must be a prime power, $q = p^s$, hence the additive group of $F_q$ is isomorphic to $(\mathbb{Z}_p)^s$. The code $C$ may be regarded as a linear time-invariant code over $((\mathbb{Z}_p)^s)^n$.

The groups $((\mathbb{Z}_p)^s)^n$ and $(((\mathbb{Z}_p)^s)^n)^{\mathbb{Z}}$ are elementary abelian, as are all of their subgroups. Any subgroup $H$ of an elementary abelian group $G$ has a complement $J$ such that $G$ is a direct sum (internal direct product) $G = H \oplus J$. Thus, all of the coset decompositions in the previous section become direct sums when applied to fields.

In particular, the subcode $C_{[k,k+j]}$ is the direct sum of the granule $\Gamma_{[k,k+j]}$ and the subcode $C_{[k,k+j]} \oplus C_{[k,k+j]}$, the code $C$ is equal to the direct sum of granules

$$C = \bigoplus_{0 \leq j \leq \nu} \bigoplus_{k \in \mathbb{Z}} \Gamma_{[k,k+j]}, \quad (3.55)$$

and the state group $\Sigma_k$ is the direct sum of granules

$$\Sigma_k = \bigoplus_{1 \leq j \leq \nu} \bigoplus_{i \in [k-j,k+j]} \Gamma_{[i,i+j]}.$$  \hspace{1cm} (3.56)

Equation 3.55 is equivalent to Theorem 1 of Roos [48, p. 679].

For a linear convolutional code $C$ over a field $F_q$, the input chain $F_0 \subseteq \cdots \subseteq F_\nu = F$ is a sequence of vector spaces of increasing dimension $d_j$ over $F_q$. The granules of $C$ are isomorphic to $(F_q)^{d_j-d_{j-1}}$ for $1 \leq j \leq \nu$. If $C$ is a rate-$k/n$ code, then the dimension of $F$ must be $k$, so $F \cong ((\mathbb{Z}_p)^s)^k$, and $C \cong (((\mathbb{Z}_p)^s)^k)^{\mathbb{Z}}$. The minimal encoder therefore establishes an isomorphism, not merely a one-to-one correspondence, between the input sequence space $F^{\mathbb{Z}}$ and the linear convolutional code $C$.

The dimensions $\{d_j : 1 \leq j \leq \nu\}$ may be alternatively characterized by $k$ parameters $\nu_i \leq \nu, 1 \leq i \leq k$, called the constraint lengths or Kronecker indices of $C$. The constraint lengths indicate the locations in the input chain where the dimension increases; they are defined such that $d_j = |\{i : \nu_i \leq j\}|$. 


The overall constraint length \( \nu_0 = \nu_1 + \ldots + \nu_k \) is equal to \( \nu_0 = \prod_{1 \leq j \leq \nu} d_j \), and by the state group size theorem,

\[
|\Sigma_k| = \prod_{1 \leq j \leq \nu} q^{d_j} = \prod_{1 \leq i \leq k} q^{\nu_i} = q^{\nu_0}.
\] (3.57)

A minimal encoder may then be constructed in controller canonical form from a set of \( k \) generators \( g_i, 1 \leq i \leq k \), where each \( g_i \) is an element of \( C_{[0, \nu]} \) that is not a sum of strictly shorter sequences. The full set of minimal-length generators is the set of time shifts of \( \{ \alpha_i g_i : 1 \leq i \leq k, \alpha_i \in F_q \} \). The minimal encoder shown in Figure 3.11 then has \( k \) shift registers of lengths \( \nu_i \leq \nu, 1 \leq i \leq k \), rather than \( \nu \) shift registers of length \( j \) and dimension \( d_j - d_{j-1}, 1 \leq j \leq \nu \).

These results reproduce the constructions in Forney [13] and Roos [48] of minimal encoders for linear time-invariant convolutional codes over finite fields. We conjecture that all of the results in [13, 14, 15, 18, 48] can be similarly replicated, using only the group structure of \( F_q \).

A similar development may be carried out for a discrete-time linear time-invariant system over a rational vector space \( V = Q^n \). However, extending these results to a real vector space \( R^n \) poses problems lying outside group theory; for example, as an additive group, \( R \cong R^n \) [49, p. 247]. It seems likely that such problems can be surmounted by restricting attention to only those subgroups of \( R^n \) that are also subspaces.

In the next two chapters we apply the basic structure theory developed in this chapter to codes over groups of isometries. Isometry codes connect the algebraic structure of group systems to the geometric structure of Euclidean codes.
CHAPTER 3. GROUP SYSTEMS
Chapter 4

Isometries and partitions

Most useful trellis codes are based on geometrically uniform signal sets and partitions. Such partitions arise naturally in codes over isometry groups. In this chapter we study the interplay between quotients of isometry groups and the partitions they generate. The results we develop are essential to the next chapter, where we realize trellis codes as group codes over isometry alphabets.

We begin in Section 4.1 by defining isometries of Euclidean and Hamming spaces. Isometry groups induce a type of group action on a set; we review the terminology and notation of group actions in Section 4.2.

We define geometrically uniform partitions in Section 4.3, as introduced by Forney [19], and describe briefly how they relate to similar concepts in coding theory and the theory of color symmetry.

Collections of left cosets of isometry groups are closely related to geometrically uniform partitions. We explore this relationship in detail in Section 4.4. The special case of normal quotients of isometry groups, which is of particular importance to isometry codes, is considered in Section 4.5.

Using the results of the previous two sections, in Section 4.6 we show how to enumerate geometrically uniform partitions and how to find quotient groups $U/U'$ that generate them.
4.1 Definitions

The Euclidean vector space $\mathbb{R}^n$ is constructed by equipping the $n$-fold direct product of the reals $\mathbb{R}$ with a Euclidean norm. If $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and $y = (y_1, \ldots, y_n) \in \mathbb{R}^n$, then the squared Euclidean distance $d^2(x, y)$ between $x$ and $y$ is

$$d^2(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2. \quad (4.1)$$

Infinite dimensional Euclidean space $\mathbb{R}^\mathbb{Z}$ is defined similarly, with the extended real-valued squared distance between sequences $x$ and $y$ defined by

$$d^2(x, y) = \sum_{i=-\infty}^{\infty} (x_i - y_i)^2. \quad (4.2)$$

We permit infinitely many components of a point in $\mathbb{R}^\mathbb{Z}$ to be nonzero; hence there exist points in $\mathbb{R}^\mathbb{Z}$ separated by an infinite distance. For this reason our distance measure on $\mathbb{R}^\mathbb{Z}$ is neither a metric nor a norm.

Our interest in Euclidean distance arises from its use in coding. When a collection of points in Euclidean space is used as a signaling alphabet in the presence of additive white Gaussian noise, maximum-likelihood decoding and nearest-neighbor decoding are equivalent.

The binary Hamming space $\{0,1\}^n$ is the set of all length-$n$ binary sequences equipped with a Hamming metric $d(x, y) = w(x \oplus y)$, where $w(x)$ is the weight of the binary vector $x$ and $\oplus$ is componentwise exclusive-or addition. More generally, the Hamming distance between two vectors is the number of components in which they differ. Infinite-dimensional binary Hamming space $\{0,1\}^\mathbb{Z}$ is defined analogously.

Binary Hamming space can be embedded in Euclidean space by lifting the binary 0's and 1's to their real counterparts. Under this lifting each binary sequence maps to a vertex of the unit $n$-cube; Hamming distance becomes squared Euclidean distance.

For the sake of concreteness, in our subsequent discussion we consider only finite-dimensional Euclidean spaces $\mathbb{R}^n$. Our results apply equally well to Hamming spaces, infinite-dimensional spaces and to any other space with a notion of distance. For example, a useful infinite-dimensional distance for partial-response channels measures the Euclidean distance between sequences after they are passed through a linear filter. For our purposes, the distance measure on a space $X$ serves only to define the group of distance-preserving transformations of $X$. 

4.1. DEFINITIONS

An isometry $\phi$ of $R^n$ is a bijection $\phi: R^n \rightarrow R^n$ that preserves Euclidean distance, so that $d(x, y) = d(\phi(x), \phi(y))$ for all $x, y \in R^n$. The set of all isometries of $R^n$, denoted by $\Gamma(R^n)$, forms a group under function composition. Functions act on the left: if $\phi, \psi \in \Gamma(R^n)$ and $x \in R^n$, then $\psi \phi x = \psi(\phi(x))$.

The isometries of finite dimensional Euclidean space are the orthonormal affine transformations. Every isometry $\phi: R^n \rightarrow R^n$ is uniquely represented by an $n \times n$ orthonormal matrix $A$ and an $n$-vector $t$ such that $\phi x = Ax + t$ for all $x \in R^n$. Conversely, any orthonormal matrix $A$ and translation vector $t$ defines a Euclidean isometry. The matrix characterization of Euclidean isometries is useful for explicitly constructing symmetry groups of subsets of Euclidean and Hamming spaces.

Infinite dimensional isometries are harder to describe in terms of matrices. For example, the delay operator is an isometry. We shall be most interested in infinite-dimensional isometries that act on finite blocks of coordinates such that the action of the isometry on one block is independent of its action on any other block. The sequences of isometries used in later sections to describe trellis codes have this character.

A Euclidean signal set $S$ is a collection of points in finite dimensional Euclidean space $R^n$. Signal sets are usually discrete: for every $x \in S$ there exists an $\varepsilon > 0$ such that $d(x, y) \geq \varepsilon$ for all $y \neq x$ in $S$. A finite signal set is a constellation. A Euclidean code is a subset of a finite or infinite Cartesian product $\prod_{k \in I} S_k$ of Euclidean signal sets $\{S_k : k \in I\}$. In other words, a Euclidean code is a dynamical system whose alphabets are Euclidean signal sets.

The symmetry group or space group $\Gamma(S)$ of a set $S \subset R^n$ is the group of isometries that map $S$ to itself:

$$\Gamma(S) = \{ \phi \in \Gamma(R^n) : \phi S = S \}.$$  \hspace{1cm} (4.3)

Most signal sets used in coding for transmission applications have nontrivial symmetry groups, primarily because the small performance gains afforded by asymmetric signal sets do not justify their increased implementation costs.

Example 4.1: The integer lattice $Z^2$ is a signal set in $R^2$. It is also a 2-dimensional Euclidean code over the 1-dimensional signal set $Z$. As a lattice, the symmetry group of $Z^2$ is the semidirect product of its group of translational symmetries, its lattice group [29], and the group of symmetries that fix the origin, its point group. The lattice group of $Z^2$ is the group of integer translations: $Z^2 + (x, y) = Z^2$ if and only if $x, y \in Z$. The point group of $Z^2$ is isomorphic to the 8-element dihedral group $D_4$, the group of
symmetries of the square, and can be represented by the eight orthonormal matrices

\[
\left\{ \begin{bmatrix} \pm 1 & 0 \\ 0 & \pm 1 \end{bmatrix}, \begin{bmatrix} 0 & \pm 1 \\ \pm 1 & 0 \end{bmatrix} \right\}. \tag{4.4}
\]

Thus, the symmetry group \( \Gamma(\mathbb{Z}^2) \) of \( \mathbb{Z}^2 \) is the group of affine transformations \( x \mapsto Ax + t \) where \( A \) is one of the 8 matrices in (4.4) and \( t \) is a vector in \( \mathbb{Z}^2 \).

\( \square \)

![Figure 4.1: An 8-PSK signal set.](image)

**Example 4.2:** The 8-PSK signal set shown in Figure 4.1 consists of 8 points spaced equally around the unit circle. The signal set may also be described as the 8-element Euclidean code \( \{(\pm a, \pm b), (\pm b, \pm a)\} \) over the one-dimensional signal set \( \{\pm a, \pm b\} \), where \( a = \sin \pi/8 \) and \( b = \cos \pi/8 \) [37].

The symmetry group of 8-PSK is the 16-element dihedral group \( D_8 \), the group of symmetries of the regular octagon. Any rotation by a multiple of \( \pi/4 \) radians is a symmetry of 8-PSK, as is any reflection through the perpendicular bisector of a line connecting two points in the set. Explicitly, \( D_8 \) consists of 16 orthogonal transformations of the form \( x \mapsto Ax \), where \( A \) is one of

\[
\left\{ \begin{bmatrix} \cos k\pi/8 & \pm \sin k\pi/8 \\ \sin k\pi/8 & \mp \cos k\pi/8 \end{bmatrix} : k = 0, 1, \ldots, 7 \right\}. \tag{4.5}
\]

\( \square \)

### 4.2 Group Actions

The action of a symmetry group \( \Gamma(S) \) on \( S \) is an example of group action on a set (see Rotman [49, pp. 178–194]). The theory we develop in the remainder
of the chapter applies to any group action, not merely those induced by isometries.

A group action of a group $G$ on a set $S$ is defined by a homomorphism $\psi$ from $G$ to the group of all permutations of $S$. The elements of $G$ act on $S$ via $gs = \psi(g)s$ for all $g \in G$ and $s \in S$. In particular, $1s = s$ and $g(hs) = (gh)s$ for all points $s \in S$ and elements $g, h \in G$, where $1$ denotes the identity element of $G$. A group of symmetries of $S$ is trivially a group action on $S$: the homomorphism $\psi$ that defines the group action is an injection.

Given an action of a group $G$ on a set $S$, the stabilizer $G_{S'}$ of a subset $S' \subseteq S$ is the collection of elements $g \in G$ such that $gS' = S'$, which is obviously a subgroup of $G$. In words, the stabilizer of $S'$ is the subgroup of $G$ that permutes the points of $S'$. For any $g \in G$, the stabilizer of the set $gS'$ is $G_{gS'} = gG_{S'}g^{-1}$. The stabilizer $G_x$ of a point $x \in S$ is a normal subgroup of $G$ if and only if $G_x$ acts trivially on $S$, i.e., if $gy = y$ for all $g \in G_x$ and $y \in S$.

A group action of $G$ on $S$ is transitive if for any two points $x, y \in S$ there exists a $g \in G$ with $gx = y$. A transitive group action is sharply transitive or regular if for all $x, y \in S$ the element $g \in G$ that sends $x$ to $y$ is unique. A transitive group action is sharply transitive if and only if every element of $S$ has a trivial stabilizer, which is true if and only if any one element of $S$ has a trivial stabilizer.

For any element $x \in S$, the set $Gx$ is the orbit of $x$ under $G$. A group action partitions a set into disjoint orbits; the group $G$ is transitive on $S$ if and only if there is only one orbit $Gx = S$ for any $x \in S$.

Transitive actions of isometry groups on signal sets are the subject of the next section.

### 4.3 Geometrically uniform sets and partitions

Our interest in codes over groups of isometries is motivated by the study of geometrically uniform signal sets, partitions and Euclidean codes. In this section we define these terms and review related concepts that appear in coding theory and other fields.
4.3.1 Geometrically uniform sets

The following definition applies to any finite or infinite dimensional space that has well-defined isometries.

Definition 4.1: A set is geometrically uniform if its symmetry group is transitive.

Many types of useful structures are geometrically uniform, including binary linear block or convolutional codes in Hamming space, Euclidean lattices, and (ignoring edge effects) most useful trellis codes.

Geometrically uniform codes have several desirable properties. Foremost among them is the uniform error property. When a geometrically uniform Euclidean code is used for data transmission over an additive white Gaussian noise channel with maximum likelihood decoding, the probability of error is independent of the transmitted codeword. The uniform error property greatly simplifies the performance analysis of geometrically uniform codes.

Intuitively speaking, every point in a geometrically uniform signal set looks the same. Two travellers, lost in this Euclidean universe, cannot arrange to meet at a point of the set, for without a shared coordinate system the signal points cannot be distinguished. Travelling from point to point will at best resolve the self-similarity of the universe up to reflection, assuming the travellers can tell their left hands from their right. (If, however, the symmetry group of the signal set contains no translations, then the center of the universe is distinguished as the sole point fixed by every symmetry. Every line of reflection passes through the center, every rotation revolves around the center, and every point in the signal set is the same distance from the center. The travelers can meet there and have a picnic.)

The formal concept of geometric uniformity likely originated in the theory of crystallography or in group theory, though man-made regular mosaics and patterns have existed for centuries. Forney introduced the terminology "geometrically uniform" in [19]; Loeliger defined a similar concept in [38]. See [19] for references to other related concepts in coding theory.

Loeliger [38] defined the concept of a signal set matched to a group and showed that a finite Euclidean signal set can be matched to a group if and only if it is geometrically uniform. In Loeliger's terminology, a matched mapping is a function $l: G \rightarrow S$ from group elements $G$ to points $S$ such that $d(l(g), l(g')) = d(l(g^{-1}g'), l(1))$ for all $g, g' \in G$, where 1 denotes the identity element of $G$. Theorem 5 of [38] establishes that any transitive
4.3. GEOMETRICALLY UNIFORM SETS AND PARTITIONS

A group of isometries $G$ of $S$ induces a matched mapping $l: g \mapsto g x$ for any initial point $x \in S$, and conversely for any matched mapping $l: G \to S$ there exists a homomorphism from $G$ to a transitive subgroup of $\Gamma(S)$. Wan [61] has extended Loeliger's proof to include infinite signal sets drawn from finite-dimensional Euclidean spaces. It is likely that the equivalence of these two concepts extends to infinite-dimensional spaces and non-Euclidean spaces as well.

Two dimensional geometrically uniform signal sets appear in Grünbaum and Shephard [28, pp. 238–243] as transitive dot patterns where they are classified into 39 categories. In this classification distinct categories of dot patterns may have isomorphic symmetry groups; if not, there could be only one type of periodic dot pattern for each of the 17 plane crystallographic groups.

To prove that a set is geometrically uniform it suffices to find a transitive subgroup of the symmetry group. A transitive group $U$ of symmetries of a set $S$ is a generating group for $S$; a sharply transitive group of symmetries is a faithful generating group. If $U$ is a generating group for $S$ then $U x = S$ for any $x \in S$. While the full symmetry group of a set may be difficult to describe, particularly for infinite dimensional codes, sharply transitive subgroups are often much simpler. Note, however, that transitive symmetry groups do not always have sharply transitive subgroups.

**Example 4.3:** The 8-PSK signal set shown in Figure 4.1 is geometrically uniform: rotations about the origin suffice to map any point to any other while preserving the set. The set of such rotations is a sharply transitive subgroup of $\Gamma(S)$. □

**Example 4.4:** Slepian [54] uses a matrix representation of the symmetric group $S_5$ on 5 letters to construct a 10 point geometrically uniform signal set in $R^5$ whose symmetry group contains no sharply transitive subgroup. □

Geometrically uniform sets have transitive symmetry groups. Conversely, given any group of isometries $C$, we can construct a geometrically uniform set $S = C x$ by applying $C$ to an initial point $x$. The set $S$ is the orbit of $x$ under $C$, hence the action of $C$ on $S$ is transitive by construction and $C$ is a generating group for $S$. 

4.3.2 Geometrically uniform partitions

A central idea in trellis coding theory is that of a partition of a set into a finite number of disjoint subsets. Partitions of signal sets are also used in Euclidean block codes; see, for example, Ginzburg [26] and Kschischang [36]. We will find in Chapter 5 that such partitions arise naturally in group codes over isometry alphabets.

Let \( Y = \{S_1, \ldots, S_m\} \) be a partition of \( S \) into disjoint subsets, so that

\[
S = \bigcup_{i=1}^{m} S_i. \tag{4.6}
\]

The subsets \( S_i \) are the cells of the partition. The partition symmetry group \( \Gamma(Y) \) of \( Y \) is the set of isometries \( \phi \in \Gamma(S) \) that permute the cells of the partition, so that for each \( i = 1, \ldots, m \) there exists a \( j \) such that \( \phi S_i = S_j \). Partition symmetries are the isometries of \( S \) that preserve the partition: if a point of \( S_i \) is mapped to a point of \( S_j \) then all points of \( S_i \) are mapped to \( S_j \). Clearly \( \Gamma(Y) \subseteq \Gamma(S) \). It often happens in coding applications that the partition symmetry group \( \Gamma(Y) \) equals the full symmetry group \( \Gamma(S) \), but this is not required.

In the language of group actions, the partition symmetry group \( \Gamma(Y) \) is a group action on the partition \( Y \) of \( S \). Each cell \( S_i \) is a block of \( S \); for all \( \phi \in \Gamma(Y) \), either \( \phi S_i = S_i \) or \( \phi S_i \cap S_i = \emptyset \).

**Definition 4.2:** A partition \( Y = \{S_1, \ldots, S_m\} \) of a set \( S \) is a geometrically uniform partition if the partition symmetry group \( \Gamma(Y) \) acts transitively on \( S \).

If \( \Gamma(Y) \) is transitive on the points of \( S \) then it is transitive on the partition cells. Hence, in a geometrically uniform partition any cell \( S_i \) can be mapped to any other cell \( S_j \) while preserving the partition. The cells \( \{S_1, \ldots, S_m\} \) are therefore mutually congruent. Moreover, each cell is geometrically uniform, because for any \( S_i \) there is a subgroup of \( \Gamma(Y) \) that acts transitively on \( S_i \).

**Example 4.5:** Two geometrically uniform 2-way partitions of 8-PSK are shown in Figure 4.2. The shape of each point indicates its membership in a partition cell. The partition symmetry group for the partition on the left is \( D_8 \), the full symmetry group of the signal set. The partition symmetry group for the second example is \( D_4 \) because rotation by \( \pi/8 \) radians does not permute the cells of the partition. □
4.3. GEOMETRICALLY UNIFORM SETS AND PARTITIONS

Figure 4.2: Two geometrically uniform partitions of 8-PSK.

Forney [19] uses the notation \( S/S' \) to denote a geometrically uniform partition of \( S \) into subsets congruent to \( S' \). This notation is ambiguous when the subset \( S' \) does not uniquely identify the partition. The following lemma, satisfied by virtually all of Forney's examples, gives a sufficient condition for \( S/S' \) to denote a unique partition.

**Lemma 4.1:** Given a geometrically uniform partition \( Y = \{S_1, \ldots, S_m\} \) of \( S \), if \( \Gamma(S) = \Gamma(Y) \) then \( Y \) is the unique partition of \( S \) that contains the cell \( S_1 \).

**Proof:** Assume the conditions of the theorem hold, and let \( Y' \) be a geometrically uniform partition of \( S \) that contains \( S_1 \). For any cell \( S_i' \in Y' \) there exists a \( \phi \in \Gamma(Y') \) such that \( \phi S_1 = S_i' \). But \( \Gamma(Y') \subseteq \Gamma(S) = \Gamma(Y) \), hence \( \phi S_1 = S_j \) for some \( j = 1, \ldots, m \). The cells of \( Y' \) are therefore equal to the cells of \( Y \).

Figure 4.3: Two inequivalent 4-way partitions of the vertices of a cube.

**Example 4.6:** Two geometrically uniform partitions of the vertices of a cube into four subsets are shown in Figure 4.3. The partitions are inequivalent in
that there exists no isometry that maps one partition to the other. Both partitions share the two-point cell marked by triangles, so by Lemma 4.1 their partition symmetry groups must be smaller than the symmetry group of the whole set.

If the origin of the $x$-$y$-$z$ coordinate system is the center of the cube, then the symmetry group of the vertices of the cube is the semidirect product of the $3!$ permutations of the $x$, $y$ and $z$ axes (a group isomorphic to $D_3$) with the $2^3$ combinations of reflections through the $x$-$y$, $y$-$z$ and $x$-$z$ planes (a group isomorphic to $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_2$), for a total of 48 isometries.

The symmetry group of the partition on the right is isomorphic to $D_4 \times \mathbb{Z}_2$. It is the semidirect product of the two-element group that contains the permutation of the $y$ and $z$ axes with the group that contains the $2^3$ reflections through the $x$-$y$, $y$-$z$ and $x$-$z$ planes.

The symmetry group of the partition on the left is isomorphic to $D_4$. It is the semidirect product of the two-element group that contains the composition of reflection through the $x$-$y$ plane and exchange of the $x$ and $y$ axes with the group that contains the $2^2$ combinations of reflections through the $y$-$z$ and $x$-$z$ planes. ♡

An $m$-way partition $Y = \{S_1, \ldots, S_m\}$ of a set $S$ can be interpreted as a coloring $k: S \to \{1, \ldots, m\}$ of the points of $S$ with $m$ distinct colors $1, \ldots, m$ such that $k(S_i) = i$ for $i = 1, \ldots, m$. In the theory of color symmetry, a branch of mathematical crystallography, a geometrically uniform partition is a chromatic coloring. A partition for which the partition symmetry group and the full symmetry group $\Gamma(S)$ coincide is a perfect coloring.

The color symmetry literature is concerned largely with questions of classification: what does it mean for two colorings to be equivalent, and how many equivalence classes of colorings result? Planar classifications have been carried out by Senechal [51, 52], Grünbaum and Shephard [27], Schwarzenberger [50] and Wieting [64]. Based on these studies, the problem of classifying all possible geometrically uniform partitions in higher dimensional spaces appears quite challenging.

A complication arises in the color symmetry literature from considering regular arrangements of patterns and tiles rather than points. Patterns and tiles admit more types of symmetries and colorings than points, hence classification schemes tend to be overly refined.
4.4  Partitions generated by cosets of isometries

In this section we show that geometrically uniform partitions are closely related to collections of left cosets $U//U'$ of isometries. Our primary interest lies in the special case where $U'$ is a normal subgroup of $U$, for such quotients arise naturally in systems over isometry groups. While the theory of geometrically uniform partitions is much simpler in this special case, it is also important to understand the types of geometrically uniform partitions excluded by the assumption of normality. Furthermore, when constructing new geometrically uniform partitions, it is difficult to identify a priori the partitions that can be generated by normal quotients $U/U'$. In the following discussion we therefore consider the general case where $U'$ is not necessarily normal in $U$.

Let $U$ be a generating group for $S$. Then $Ux = S$ for any $x \in S$. A subgroup $U'$ of $U$ of index $[U:U'] = m$ induces a partition of the group $U$ into $m$ left cosets $U//U' = \{u_1U', u_2U', \ldots, u_mU'\}$, where $\{u_i \in U : i = 1, \ldots, m\}$ is a set of coset representatives. The double slash $//$ indicates that $U'$ may or may not be a normal subgroup of $U$.

Applying the isometries of a coset $uU'$ to a point $x \in S$ yields a subset $uU'x$ of $S$. The transitivity of $U$ guarantees that every point of $S$ appears in a subset $uU'x$ for some $u \in U$. It may happen that, ignoring duplicates, the subsets $\{uU'x : u \in U\}$ comprise a partition of $S$; this occurs if for all $u, v \in U$ the subsets $uU'x$ and $vU'x$ are either disjoint or equal. The number of distinct sets in the partition must be a divisor of $m$.

The sets $uU'x$ and $vU'x$ are disjoint or equal if and only if $v^{-1}uU'x$ and $U'x$ are disjoint or equal, so to test if $U//U'$ induces a partition of $S$ it suffices to check that $U'x$ and $uU'x$ are either disjoint or equal for all $u \in U$.

**Definition 4.3:** Let $U$ be a transitive group of symmetries of $S$, let $x$ be a point in $S$, and let $U' \subseteq U$ be a subgroup of finite index such that for all $u \in U$ the subsets $U'x$ and $uU'x$ are either disjoint or equal. Then the collection of distinct sets in $\{uU'x : u \in U\}$ is the partition of $S$ induced by $U//U'$ with initial point $x$. We denote the partition by $U//U'x$.

The next lemma justifies our interest in partitions induced by cosets of isometries.

**Lemma 4.2:** Every induced partition $U//U'x$ is geometrically uniform.
**Proof:** We must show that the partition symmetry group of an induced partition $U//U'x$ is transitive. It suffices to show that the partition symmetry group contains $U$, which is transitive by assumption. An isometry $u \in U$ is a partition symmetry if for any partition cell $S'$, the set $uS'$ is also a cell of the partition. We know that $S' = vU'x$ for some $v \in U$, hence $uS' = uvU'x$ is a cell of the partition. 

Conversely, every geometrically uniform $m$-way partition is induced by a collection of $m$ left cosets of isometries.

**Lemma 4.3:** Let $U = \Gamma(S)$ be the partition symmetry group of an $m$-way geometrically uniform partition $Y$ and let $U' = U_{S'}$ be the stabilizer of a partition cell $S'$. Then $Y = U//U'x$ for any $x \in S'$.

**Proof:** We must show for all $u \in U$ that $uU'x$ is a cell of the partition $Y$, and that every partition cell equals $uU'x$ for some $u \in U$. The group $U$ is transitive on $S$, hence $U' = U_{S'}$ is transitive on $S'$ and $S' = U'x$. Partition symmetries permute the cells of the partition, hence $uU'x = uS'$ must be a cell of the partition for any $u \in U$. The transitivity of the partition symmetry group ensures that every partition cell equals $uU'x$ for some $u \in U$. 

So we see that geometrically uniform partitions are induced by cosets of isometry groups, and that certain cosets of isometry groups induce geometrically uniform partitions. The next lemma, which appears as Exercise 9.15 in Rotman [49], shows that normal subgroups always induce geometrically uniform partitions.

**Lemma 4.4:** Let $U$ be a transitive group of symmetries of a set $S$ and let $U'$ be a normal subgroup of finite index in $U$. Then $U/U'x$ is a geometrically uniform partition of $S$ for any initial point $x \in S$. Further, all choices of $x \in S$ yield the same partition.

**Proof:** Given some $x \in S$, we must show that for all $u \in U$ that $U'x$ and $uU'x$ are either disjoint or equal. Assume that $U'x$ and $uU'x$ intersect at $y$. We know that $U'y = Y'x$ for all $y \in U'x$, hence $U'x \subseteq U'(uU'x)$. The normality of $U'$ implies that $U'(uU'x) = uU'x$, so $U'x \subseteq uU'x$. For the reverse inclusion, observe that $U'x$ and $uU'x$ intersect if and only if $U'x$ and $u^{-1}U'x$ intersect, so the same reasoning implies that $U'x \subseteq u^{-1}U'x$, or equivalently that $uU'x \subseteq U'x$. We conclude that $U'x = uU'x$. 

To prove that all choices of $x \in S$ yield the same partition, we must show for all $x, y \in S$ that there exists a $u \in U$ such that $U'x = uU'y$. Let $u \in U$ be any isometry that maps $y$ to $x$. Then $U'x = U'uy = uU'y$ as desired. ■

Geometrically uniform partitions induced by normal subgroups are studied by Forney [19]. These types of partitions arise naturally in isometry codes, and shall be our main focus of attention in later sections. Unfortunately, not every geometrically uniform partition is induced by a normal quotient $U/U'$. The partition in the left half of Figure 4.3 is such a case.

A second sufficient condition for a collection of cosets $U//U'$ to induce a geometrically uniform partition is that $U'$ include all stabilizers $U_x = \{u \in U : ux = x\}$ of a point $x \in S$. An equivalent result, expressed in the language of color symmetry, appears in Schwarzenberger [50, p. 229].

**Lemma 4.5:** Let $U$ be a transitive group of symmetries of a set $S$, let $U'$ be a subgroup of finite index $[U:U'] = m$, and let $x \in S$ be an initial point. If $U'_x = U_x$, or equivalently if $U_x \subseteq U'$, then $U//U'$ induces an $m$-way geometrically uniform partition of $S$ with initial point $x$.

**Proof:** For all $u, v \in U$, we must show that $uU'x$ and $vU'x$ are disjoint when $uU' \neq vU'$. Assume $uU'x$ and $vU'x$ intersect at a point $y$. We must show that $uU' = vU'$. From the transitivity of $U$, there exists a $\phi \in U$ that maps $y$ to $x$. The intersection of $\phi uU'x$ and $\phi vU'x$ includes $\phi y = x$, so the cosets $\phi uU'$ and $\phi vU'$ of $U'$ both include stabilizers of $x$. But by assumption all stabilizers of $x$ are in $U'$, hence $U' = \phi uU' = \phi vU'$. ■

Lemma 4.3 supplies an immediate converse: every geometrically uniform partition is induced by some $U$, $U'$ and $x$ such that $U_x = U'_x$.

**Corollary 4.6:** Let $U$ be a sharply transitive group of symmetries of a set $S$. Then any subgroup $U'$ of finite index $[U:U'] = m$ induces a geometrically uniform partition of $S$ into $m$ subsets for any initial point $x \in S$.

**Proof:** A group of symmetries $U$ is sharply transitive if and only if $U_x$ is trivial for all $x \in S$. Any subgroup $U'$ of $U$ therefore satisfies the conditions of Lemma 4.5 for any $x \in S$. Indeed, all choices of $x \in S$ yield the same partition. ■
The geometrically uniform partitions considered by Forney [19] satisfy the conditions of both Corollary 4.6 and Lemma 4.4: the group $U$ is sharply transitive and the subgroup $U'$ is normal.

A single geometrically uniform partition can be induced by many different collections of left cosets $U//U'$ and initial points $x$. When enumerating partitions it is helpful to have tools that avoid duplicates. The next few lemmas provide some results in this direction.

**Lemma 4.7:** Let $U//U'x$ be a geometrically uniform partition of $S$ with initial point $x \in S$. For any $u \in U$ let $V' = uU'u^{-1}$ and $y = ux$. Then $U//V'y = U//U'x$.

**Proof:** For all $v_1 \in U$ we must show that $v_1V'y$ is equal to $v_2U'x$ for some $v_2 \in U$. Set $v_2 = v_1u$. Then $v_2U'x = v_1uU'x = v_1uU'u^{-1}ux = v_1V'y$. ■

Two partitions $Y, Y'$ of a set $S$ are congruent if there exists an isometry $\phi \in \Gamma(S)$ such that $\phi Y = Y'$. Congruent partitions are essentially equivalent for coding purposes.

**Lemma 4.8:** Let $U//U'x$ be a geometrically uniform partition of $S$ and let $Y'$ be congruent to $U//U'x$. Then there exists an isometry $u \in \Gamma(S)_x$ such that $Y = (uU'u^{-1})/(uU'u^{-1})x$.

**Proof:** Let $S'$ be the cell of the partition $Y$ that contains $x$, and let $u \in \Gamma(S)_x$ be an isometry that sends $U'x$ to $S'$ while fixing $x$. Such an isometry exists because $Y$ and $U//U'x$ are congruent and have transitive symmetry groups.

We must show that for any $w \in uU'u^{-1}$ the set $w(uU'u^{-1})x$ is congruent via $u$ to some cell of the partition $U//U'x$. Select $v \in U$ so that $w = uvu^{-1}$. Then

$$w(uU'u^{-1})x = uvu^{-1}uU'u^{-1}x$$

$$= uvU'u^{-1}x$$

$$= uvU'x$$

because $u$ fixes $x$. Thus any cell $w(uU'u^{-1})x$ in $Y$ is congruent via $u$ to a cell $vU'x$ in $U//U'x$. ■

Loosely speaking, the lemma shows that congruent partitions can be generated by conjugate generating groups. The proof of the lemma implies a
4.5. NORMAL GEOMETRICALLY UNIFOR M PARTITIONS

A transitive group $U$ of symmetries of a set $S$ is **minimally transitive** if it contains no proper transitive subgroup. It can be shown (using Zorn's lemma) that any transitive symmetry group has a minimally transitive subgroup. The final lemma of this section, a variant of Lemma 4.3, shows that minimally transitive groups of symmetries suffice to generate all possible geometrically uniform partitions.

**Lemma 4.9:** Let $U$ be any minimally transitive subgroup of the partition symmetry group $\Gamma(Y)$ of a geometrically uniform partition $Y$ and let $U' = U_{S'}$ be the stabilizer of a cell $S' \in Y$. Then $Y = U//U'x$ for any $x \in S'$.

**Proof:** As in the proof of Lemma 4.3. ■

We next consider in detail geometrically uniform partitions induced by quotients $U/U'$ of isometries where $U'$ is normal in $U$.

### 4.5 Normal geometrically uniform partitions

A geometrically uniform partition $Y$ of $S$ is **normal** if there exists a group $U$ and a normal subgroup $U'$ such that $U/U'x = Y$ for $x \in S$. We shall see that a normal partition has a canonical generating quotient $\Gamma(Y)/\Gamma(Y)_Y$ through which all other generating quotients factor. The canonical quotient can often be computed explicitly, and it plays a central role in later sections.

The existence of a normal quotient $U/U'$ that induces a partition $Y$ depends on the transitivity of the partition stabilizer $\Gamma(Y)_Y$. Given a partition $Y = \{S_1, \ldots, S_m\}$ of a set $S$, the **partition stabilizer** $\Gamma(Y)_Y$ is the group of isometries $\phi \in \Gamma(Y)$ such that $\phi S_i = S_i$ for $i = 1, \ldots, m$. The elements of the partition stabilizer permute the points of the partition but leave the partition cells $\{S_1, \ldots, S_m\}$ fixed.

**Lemma 4.10:** Let $Y = \{S_1, \ldots, S_m\}$ be a geometrically uniform partition of a set $S$. Then the partition stabilizer $\Gamma(Y)_Y$ is a normal subgroup of $\Gamma(Y)$. Conversely, if $U/U'x = Y$, where $U'$ is normal in $U$, then $U'$ is a subgroup of $\Gamma(Y)_Y$. 
Proof: To prove that $\Gamma(Y)_Y$ is normal in $\Gamma(Y)$, we must show for all $\phi \in \Gamma(Y)$ and $\gamma \in \Gamma(Y)_Y$ that $\phi^{-1} \gamma \phi S_i = S_i$ for $i = 1, \ldots, m$. We know that $\gamma(\phi S_i) = \phi S_i$, hence $\phi^{-1} \gamma \phi S_i = \phi^{-1} \phi S_i = S_i$ as required.

For the converse, assume that $U/U'x = Y$ for some $x \in S_1$. We must show that if $u' \in U'$ then $u'S_i = S_i$ for $i = 1, \ldots, m$. Select $u \in U$ such that $uS_1 = S_i$. Then $u'S_i = u'uS_1 = u(u^{-1}u'u)S_1 = uS_1 = S_i$ because $u^{-1}u'u \in U'$.

Corollary 4.11: Let $Y = \{S_1, \ldots, S_m\}$ be a geometrically uniform partition of a set $S$. There exists a normal quotient $U/U'$ that induces the partition $Y$ if and only if $\Gamma(Y)_Y$ is transitive on $S_1$.

Proof: If a partition stabilizer $\Gamma(Y)_Y$ is transitive on $S_1$ (in which case it is transitive on $S_i$ for $i = 1, \ldots, m$), then Lemmas 4.4 and 4.10 imply that $Y = \Gamma(Y)/\Gamma(Y)_Yx$ for any $x \in S$. Conversely, if $\Gamma(Y)_Y$ is not transitive on $S_1$ then no subgroup $U'$ of $\Gamma(Y)_Y$ can be transitive on $S_1$.

The following theorem establishes $\Gamma(Y)/\Gamma(Y)_Y$ as the canonical generating quotient for any isometry code based on the partition $Y$.

Theorem 4.12: If $\Gamma(Y)/\Gamma(Y)_Yx = U/U'x = Y$ then there is a homomorphism $h$ from $U/U'$ to a subgroup of $\Gamma(Y)/\Gamma(Y)_Y$ such that $uU'x = h(uU')x$ for all $u \in U$.

Proof: We know that $U \subseteq \Gamma(Y)$ and $U' \subseteq \Gamma(Y)_Y$. Define $h: U/U' \to \Gamma(Y)/\Gamma(Y)_Y$ by $h: uU' \mapsto u\Gamma(Y)_Y$. Then $h(uU')x = u(\Gamma(Y)_Yx) = uU'x$ as desired. The image of $h$ is $(U\Gamma(Y)_Y)/\Gamma(Y)_Y$. The map $h$ is a homomorphism because for any two cosets $uU'$ and $vU'$ in $U/U'$, $h(uU'vU') = h(uvU') = uv\Gamma(Y)_Y = u\Gamma(Y)_Yv\Gamma(Y) = h(uU')h(vU')$.

This theorem will play a key role in Section 5.4. Its central import is that any isometry code over a group of isometries $U$ with parallel transition group $U'$ generates the same Euclidean code as an isometry code over $\Gamma(Y)$ with parallel transition group $\Gamma(Y)_Y$.

The quotient group $\Gamma(Y)/\Gamma(Y)_Y$ has a faithful representation as a group of permutations of the cells of $Y = \{S_1, \ldots, S_m\}$. Specifically, for all $\phi \in \Gamma(Y)/\Gamma(Y)_Y$ there exists a permutation $p: Y \to Y$ such that $\phi S_i = p(S_i)$ for $i = 1, \ldots, m$. Conversely, two cosets $\phi, \phi' \in \Gamma(Y)/\Gamma(Y)_Y$ are equal if and only if their induced permutation actions on $Y$ are equal.
**Definition 4.4:** The group of *isometric permutations* of a geometrically uniform partition $Y$ is the set of permutations of $Y$ induced by $\Gamma(Y)/\Gamma(Y)_Y$.

The permutation representation of $\Gamma(Y)/\Gamma(Y)_Y$ will prove useful in Section 5.4 when constructing isometry code realizations of Euclidean codes.

![Diagram](image)

**Figure 4.4:** The geometrically uniform partition $\mathbf{RZ}^2/4\mathbf{Z}^2$.

**Example 4.7:** Let $Y$ be the geometrically uniform partition of the lattice translate $\mathbf{RZ}^2 + (1,0)$ into 8 translates of $4\mathbf{Z}^2$, where

$$
\mathbf{R} = \begin{bmatrix}
1 & -1 \\
1 & 1
\end{bmatrix}
$$

represents rotation by 45 degrees and scaling by $\sqrt{2}$. The 8 cells of the partition are labeled with the letters \{a,b,\ldots,h\} as indicated in Figure 4.4 and in Table 4.1.

The symmetry properties of the lattice $\mathbf{RZ}^2$ and its translate $\mathbf{RZ}^2 + (1,0)$ are essentially the same, so we shall not distinguish sharply between them in this example. The lattice translate is more useful for coding because it has no zero-energy point at the origin.

The symmetry group $\Gamma(\mathbf{RZ}^2)$ is the semidirect product of the group of translations by elements of $\mathbf{RZ}^2$ (the lattice group) by the group of symmetries of the square (the point group). The point group, $D_4$, consists of
Table 4.1: Labeling of the partition $\mathbb{RZ}^2/4\mathbb{Z}^2$.

<table>
<thead>
<tr>
<th>label</th>
<th>subset</th>
<th>label</th>
<th>subset</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>$4\mathbb{Z}^2 + (0,1)$</td>
<td>e</td>
<td>$4\mathbb{Z}^2 + (1,0)$</td>
</tr>
<tr>
<td>b</td>
<td>$4\mathbb{Z}^2 + (2,3)$</td>
<td>f</td>
<td>$4\mathbb{Z}^2 + (3,2)$</td>
</tr>
<tr>
<td>c</td>
<td>$4\mathbb{Z}^2 + (0,3)$</td>
<td>g</td>
<td>$4\mathbb{Z}^2 + (3,0)$</td>
</tr>
<tr>
<td>d</td>
<td>$4\mathbb{Z}^2 + (2,1)$</td>
<td>h</td>
<td>$4\mathbb{Z}^2 + (1,2)$</td>
</tr>
</tbody>
</table>

reflections across the $x$ axis, the $y$ axis, the line $x = y$ and the line $-x = y$ as well as rotations by 0, 90, 180 and 270 degrees about the origin. The space group and point group of $\mathbb{RZ}^2$ both permute the subsets of the partition $Y$, hence the partition symmetry group $\Gamma(\mathbb{Y})$ is equal to $\Gamma(\mathbb{RZ}^2)$. Lemma 4.1 therefore allows us to unambiguously refer to the partition $Y$ as $\mathbb{RZ}^2/4\mathbb{Z}^2$.

It is easy to see that the partition stabilizer $\Gamma(\mathbb{Y})_Y$ includes the group of translations by elements of $4\mathbb{Z}^2$ and no other translations. Of the 8 isometries in the point group of $\Gamma(\mathbb{Y})$, only the identity stabilizes the cells of $Y$. The partition stabilizer $\Gamma(\mathbb{Y})_Y$ therefore contains no non-translational isometries.

The quotient $\Gamma(\mathbb{Y})/\Gamma(\mathbb{Y})_Y$ is therefore isomorphic to the semidirect product of $\mathbb{Z}_4 \times \mathbb{Z}_2$ by $D_4$, since $\mathbb{RZ}^2/4\mathbb{Z}^2 \cong \mathbb{Z}_4 \times \mathbb{Z}_2$ when $\mathbb{RZ}^2$ and $4\mathbb{Z}^2$ are taken as additive groups. Each coset of $\Gamma(\mathbb{Y})/\Gamma(\mathbb{Y})_Y$ can be uniquely written as a permutation of the 8 cells of $Y$. The coset of $\Gamma(\mathbb{Y})_Y$ that contains rotation by 90 degrees clockwise about the origin acts as the permutation $(a \ e \ c \ g)(b \ f \ d \ h)$, written in standard cycle notation. Similarly, the coset of $\Gamma(\mathbb{Y})_Y$ that contains translation by $(2,0)$ has the same action on partition cells as the permutation $(a \ d)(b \ c)(e \ g)(f \ h)$. □

In the next section we use the tools developed above to describe a procedure for enumerating geometrically uniform partitions.

### 4.6 Constructing geometrically uniform partitions

In this section we consider two problems: how construct geometrically uniform partitions, and, once a partition $Y$ is discovered, how to find isometry quotients $U//U'$ that generate $Y$.

Given a geometrically uniform set $S$, we seek a systematic procedure that
finds geometrically uniform $m$-way partitions of $S$. The number of possible partitions grows rapidly with the dimensionality of the underlying space, so one should not expect an enumeration of geometrically uniform partitions to be computationally practical in all cases. A more reasonable goal is a procedure that enumerates specific types of partitions that are suspected to be useful for coding. The procedure we develop below can be adapted to such purposes.

Once we have found a geometrically uniform partition $Y = \{S_1, \ldots, S_m\}$, we would like a list of the pairs of groups $U$ and subgroups $U'$ that induce $Y$. In theory, at least, this problem has a simple solution: any transitive subgroup of the partition symmetry group $\Gamma(Y)$ will serve as a generating group $U$, and then $U'$ can be any subgroup of the stabilizer $U_{S_i}$ that is transitive on $S_1$. Of particular interest are the subgroups $U'$ that are normal in $U$.

We first consider the problem of finding geometrically uniform partitions of a given geometrically uniform set $S$. From the results of the previous sections, it suffices to consider only a single initial point $x \in S$. The choice of initial point does not affect the possible partitions of $S$ or (up to isomorphism) the possible groups $U$ and $U'$. Also, given a candidate generator group $U \subseteq \Gamma(S)$, only those subgroups $U'$ of $U$ that contain all stabilizers $U_x$ of $x$ are needed to generate the possible partitions of $S$ induced by $U$. Lastly, the conjugates $\{\phi U \phi^{-1} : \phi \in \Gamma(S)_x\}$ of a generator group $U$ yield congruent partitions and are essentially equivalent.

We proceed as follows. Compute the symmetry group $\Gamma(S)$ of $S$ and the stabilizer $\Gamma(S)_x$ of $x$ in $\Gamma(S)$. Find all transitive subgroups of $\Gamma(S)$ and arrange them in a subgroup lattice, ignoring conjugates. The leaves of the lattice are the minimally transitive generating groups. While minimally transitive groups suffice to generate all possible partitions, for efficiency it is best to start at the top of the transitive subgroup lattice and work down, breadth first, to the leaves. Any partition generated by a group $U$ can also be generated by any transitive subgroup of $U$, hence a breadth-first search avoids obvious duplicates.

Starting at the top of the subgroup lattice, consider a subgroup $U$ that acts transitively on $S$. To find all possible geometrically uniform $m$-way partitions of $S$ induced by $U$ we need only consider subgroups $U'$ of index $m$ in $U$ that contain the stabilizer $U_x$ of the initial point $x$. The stabilizer $U_x$ of $x$ is easily computed as $U_x = \Gamma(S)_x \cap U$. Find all subgroups $U'$ of $U$ of index $[U:U'] = m$ such that $U_x \subseteq U'$, or equivalently such that $U'_x = U_x$. Each induces an $m$-way geometrically uniform partition of $S$. 
CHAPTER 4. ISOMETRIES AND PARTITIONS

The conjugates $\{uU' u^{-1} : u \in U_x\}$ of $U'$ yield congruent partitions and should be avoided. More generally, if $N(U)$ is the normalizer of $U$ in $\Gamma(S)$, i.e., the largest subgroup of $\Gamma(S)$ which contains $U$ as a normal subgroup, then the conjugates $\{\phi U' \phi^{-1} : \phi \in N(U)_x\}$ also yield congruent partitions. The group $N(U)_x$ is the set of elements of $\Gamma(S)_x$ that normalize $U$, hence there is no need to compute the full normalizer $N(U)$.

Once we have found a geometrically uniform partition $Y = \{S_1, \ldots, S_m\}$ we wish to find the collections of left cosets $U/U'$ that generate it. Any transitive subgroup of the partition symmetry group $\Gamma(Y)$ will serve as a generating group $U$, and then $U'$ can be any subgroup of $U_{S_1}$ that is transitive on the partition cell $S_1$.

Theorem 4.12 shows that $\Gamma(Y)/\Gamma(Y)_Y$ and its transitive subgroups are the only quotients of interest when constructing isometry codes. If for some reason one wishes to find all subgroups $U''$ that are normal in some generating group $U$, the first step is to find the transitive subgroups of $\Gamma(Y)_Y$. For each transitive subgroup $U' \subseteq \Gamma(Y)_Y$, compute the normalizer of $U'$ in the partition symmetry group $\Gamma(Y)$. Any subgroup of the normalizer that is transitive on $Y$ can be used as a generating group $U$.

Algorithms for constructing a subgroup lattice, finding stabilizers and testing for transitivity are standard problems in computational group theory; see for example Butler [6]. For finite symmetry groups of moderate size the problem can be solved more or less mechanically. Some work in this direction appears in Benedetto et al. [2]. Analysis of infinite groups, however, may require the theory of crystallographic groups and group cohomology [29, 5]. Special cases, such as partitions of the $n$-dimensional integer lattice, are usually not so complex.

In the chapter that follows we tie our results on geometrically uniform partitions to the theory of systems over groups.
Chapter 5

Isometry codes

An isometry code is a group system over isometry alphabets. More formally,

**Definition 5.1:** An *isometry code* \( C \) defined on time axis \( I \) is a subgroup of a direct product group \( \prod_{k \in I} U_k \), where \( \{U_k : k \in I\} \) is an indexed collection of transitive isometry groups of signal sets \( \{S_k : k \in I\} \).

This chapter develops tools for realizing trellis codes isometry codes. An isometry code realization of a trellis code provides two main benefits: it establishes that the trellis code is geometrically uniform, and it gives an algebraic interpretation to the geometric structure of the trellis code. Through this structure one can hope to construct new codes and to better understand the behavior of existing ones.

A sequence of isometries is itself an isometry. Specifically, any \( c \in \prod_{k \in I} U_k \) is an isometry of the Cartesian product space \( \prod_{k \in I} S_k \). An isometry code \( C \) is therefore a group of isometries of the Euclidean space \( \prod_{k \in I} S_k \). We assume that \( C \) is trim, so that for all \( k \in I \) every isometry \( c_k \in U_k \) appears in some sequence \( c \in C \).

An isometry code is "converted" into a Euclidean code \( C \mathbf{x} \) by applying the isometries of the code to an initial sequence \( \mathbf{x} \in \prod_{k \in I} S_k \). Each isometry sequence sends the initial sequence \( \mathbf{x} \) to a trajectory in Euclidean space; the collection of all such sequences is a geometrically uniform Euclidean code. We say that \( C \) is an *isometry code realization* of the Euclidean code \( C \mathbf{x} \).

Forney [19] has shown that many useful trellis codes are generated by isometry codes. However, Forney's restriction to sharply transitive generating groups \( U \) and normal subgroups \( U' \) eliminates large classes of interesting
codes from his framework. This chapter can be viewed as a generalization of Forney's results.

Our results are organized as follows. In Section 5.1 we consider how the dynamics of an isometry code relate to the dynamics of the Euclidean code it generates. We find, in general, that the Euclidean code dynamics may be simpler than the isometry code dynamics.

In Section 5.2 we show how geometrically uniform partitions arise naturally in Euclidean codes generated by isometry codes. Using reduced codes, we generalize the concept of an “isometric labeling” introduced by Forney [19].

We introduce permutation labelings in Section 5.3 and develop a key theorem for converting Euclidean trellis sections into isometry trellis sections.

In Section 5.4 we develop a procedure for realizing trellis codes as isometry codes. The V.32 trellis code provides a detailed example. Two further examples are presented in the last section of the chapter.

5.1 The dynamics of generated Euclidean codes

The dynamics of the Euclidean code generated by an isometry code may be simpler than those of the isometry code itself. We will characterize the ways this can occur and show that the generated Euclidean code does not necessarily have a unique minimal realization in the sense of Willems [67].

Let \( C \) be an isometry code, and let \( \mathbf{x} \in \prod_{k \in \mathbb{Z}} S_k \) be an initial sequence. As a group code, \( C \) has a unique minimal state-output realization and trellis diagram that can be found using the techniques developed in Chapter 3. A trellis diagram for the Euclidean code \( C\mathbf{x} \) is readily constructed from the minimal trellis diagram of \( C \) by replacing each isometry label \( c_k \in U_k \) in the trellis section at time \( k \) by the Euclidean label \( c_k x_k \in S_k \). The resulting Euclidean trellis describes the state-output realization of \( C\mathbf{x} \) induced by \( C \).

The realization of \( C\mathbf{x} \) induced by the minimal realization of \( C \) is not necessarily minimal. As an extreme example, consider an isometry code over a group of rotations about the origin. Applying the code to the constant 0 initial sequence yields a Euclidean code that contains a single sequence. The dynamics of the Euclidean code are trivial no matter how complex the original isometry code.
5.1. THE DYNAMICS OF GENERATED EUCLIDEAN CODES

Though the induced realization may not be minimal, the following theorem shows that it has much in common with a past-induced or future-induced realization.

**Theorem 5.1:** Let $C$ be an isometry code and let $\mathbf{x}$ be an initial sequence. Then in the realization of $C\mathbf{x}$ induced by $C$, if two states have one future in common then they have all futures in common; if they have one past in common then they have all pasts in common.

**Proof:** The states of the canonical realization of $C$ at time $k$ are the cosets of $C/(C_k-C_{k+})$. For any state $cC_k-C_{k+}$, the set of sequences that pass through the corresponding state in the Euclidean realization induced by $C$ is $cC_k-C_{k+}\mathbf{x}$.

Let $c$ and $c'$ be isometry sequences in $C$ such that the Euclidean sequences $c\mathbf{x}$ and $c'\mathbf{x}$ agree on the future $k^+$. Then $C_k-C_{k+}c\mathbf{x}$ and $C_k-C_{k+}c'\mathbf{x}$ also agree on $k^+$. But $C_k-C_{k+}c = cC_k-C_{k+}$ for any $c \in C$, hence the states $cC_k-C_{k+}$ and $c'C_k-C_{k+}$ generate identical sets of futures when applied to the initial sequence $\mathbf{x}$. The same reasoning applies to sequences that agree on the past.

According to the theorem, the induced Euclidean realization fails to be minimal only if distinct states are past- or future-equivalent. A special case of this failure occurs when a state has two exiting (or entering) branches that have identical Euclidean labels.

Let $C_{x,k^-}$ be the group of sequences of $C$ such that $C_{x,k^-}\mathbf{x}$ agrees with $\mathbf{x}$ on $k^+$, and let $C_{x,k^+}$ be the group of sequences of $C$ such that $C_{x,k^+}\mathbf{x}$ agrees with $\mathbf{x}$ on $k^-$. Clearly $C_{k^-} \subseteq C_{x,k^-}$ and $C_{k^+} \subseteq C_{x,k^+}$. Moreover, the set of sequences of $C\mathbf{x}$ that agree on the future with any sequence $c\mathbf{x} \in C\mathbf{x}$ is $cC_{x,k^-}\mathbf{x}$, and the set of sequences of $C\mathbf{x}$ that agree on the past with $c\mathbf{x}$ is $cC_{x,k^+}\mathbf{x}$.

The subcodes $C_{x,k^-}$ and $C_{x,k^+}$ are not necessarily normal in $C$, hence the product $C_{x,k^-}C_{x,k^+}$ is not necessarily a subgroup of $C$. If it happens that $C_{x,k^-}C_{x,k^+}$ is a subgroup of $C$, then $C\mathbf{x}$ has a unique minimal realization whose states are generated by applying the left cosets of $C_{x,k^-}C_{x,k^+}$ in $C$ to the initial sequence $\mathbf{x}$. In general, however, the past- and future-induced realizations of the Euclidean code $C\mathbf{x}$ need not be the same.

In the special case where $C_k-C_{k^+} = C_{x,k^-}C_{x,k^+}$, the realization of the Euclidean code $C\mathbf{x}$ induced by the minimal realization of $C$ is itself minimal. When this condition holds we say that $C$ is a *faithful* isometry code realization.
of $C \mathbf{x}$. The examples we present at the end of the chapter deal exclusively with faithful realizations. Be warned, however, that not every geometrically uniform Euclidean code has a faithful isometry code realization, and even if such a realization exists it may be simpler to construct one that is not faithful.

In light of Theorem 5.1, it appears possible to develop a system theory for Euclidean codes generated by isometry codes that has much in common with the theory of group systems.

### 5.2 Reduced codes and isometric labelings

Geometrically uniform partitions arise naturally in isometry codes. At each time $k$, the parallel transition subgroup of an isometry code $C$ induces a geometrically uniform partition of the time-$k$ signal set $S$. Let $U$ be the output alphabet of $C$ at time $k$, let $U' = C_{[k,k]}$ be the parallel transition subgroup at time $k$, and let $x_k \in S_k$ be the time-$k$ component of the initial sequence $x$. Then $U'$ is a normal subgroup of $U$ and $U/U'x_k$ is a geometrically uniform partition of $S_k$.

Parallel transitions represent the nondynamical or uncoded part of the system. They can be “factored out” of an isometry code $C$ to yield a reduced system $C'$ whose output alphabet each at time $k$ is a quotient group $U/U'$. For all $k \in I$ let $r_k: U_k \rightarrow U_k/U'_k$ be the natural map, and let $r: \prod_{k \in I} U_k/U'_k \rightarrow \prod_{k \in I} U_k/U'_k$ be defined componentwise by $(r(c))_k = r_k(c_k)$ for $k \in I$ and $c \in \prod_{k \in I} U_k$. Then the image $C' = r(C)$ of $C$ under the homomorphism $r$ is the reduced code of $C$.

The canonical trellis diagram of $C'$ is essentially the same as the trellis diagram of $C$, except that each set of parallel branches is replaced by a single branch labeled with a coset of a quotient $U_k/U'_k$. The induced realizations of the Euclidean codes $C \mathbf{x}$ and $C' \mathbf{x}$ are similarly related. Each set of parallel branches in the Euclidean trellis induced by $C$ is replaced by a single branch labeled with a cell of the partition $U_k/U'_k x_k$ to form the trellis induced by $C'$.

Forney [19] uses reduced codes as a primary ingredient in his constructions of geometrically uniform Euclidean codes. In his terminology, a reduced code is a label code. Given a geometrically uniform partition $Y = U/U'x$ of a set $S$, Forney defines an isometric labeling $m: U/U' \rightarrow Y$ as the bijection $m(uU') = uU'x$ from the quotient group $U/U'$ to the cells of the geometrically uniform partition $Y$. A label code over $\prod_{k \in I} U_k/U'_k$ is converted into a geometrically uniform Euclidean code over $\prod_{k \in I} S_k$ by replacing each coset $u_k U'_k$ of $U'_k$ in
5.3. ISOMETRY AND EUCLIDEAN TRELLIS SECTIONS

$U_k$ by the cell $m_k(u_k U'_k) \subseteq S_k$.

In Forney's approach, the isometric labeling $m$ uniquely identifies the cells of the partition $U/U'x$ with the cosets of $U/U'$. The isometric labeling is bijective because the generating group $U$ is assumed to be sharply transitive on the set $S$.

The concept of an isometric labeling readily generalizes to the case where $U$ is not sharply transitive on $S$. When $U$ is not sharply transitive, the mapping $m$ defined by $m(uU') = uU'x$ for all $u \in U$ may send different cosets of $U/U'$ to the same cell of the partition $U/U'x$. The principle, however, remains the same: a label code $C'$ over $\prod_{k \in I} U_k/U'_k$ is converted into a geometrically uniform Euclidean code over $\prod_{k \in I} S_k$ by applying the isometric labelings $\{m_k : k \in I\}$ componentwise to the sequences of $C'$.

Removing the restriction that the generating group $U$ must be sharply transitive might at first appear to complicate the class of quotients $U/U'$ that need to be considered when constructing isometry codes. Theorem 4.12 of Section 4.5 shows, however, that for any geometrically uniform partition $Y$ generated by a normal quotient $U/U'$ there is a homomorphism $h: U/U' \rightarrow \Gamma(Y)/\Gamma(Y)_Y$ such that $h(uU')x = uU'x$. Applying the homomorphisms $\{h_k : k \in I\}$ componentwise to the sequences of a reduced code $C'$ over $\prod_{k \in I} U_k/U'_k$ yields a new reduced code $C''$ over $\prod_{k \in I} \Gamma(Y_k)/\Gamma(Y_k)_{Y_k}$ such that the induced Euclidean codes $C \bar{x}$ and $C' \bar{x}$ are equal. The quotient $\Gamma(Y)/\Gamma(Y)_Y$ therefore acts as a canonical output alphabet for a reduced isometry code over a partition $Y$.

5.3 Isometry and Euclidean trellis sections

We have seen how an isometric labeling can be used to convert a reduced trellis section with isometry labels to a trellis section labeled with cells of a geometrically uniform partition. But how can we go backwards, from Euclidean subsets to isometry labels?

In Forney's framework an isometric labeling $m$ is a bijection, hence there is a unique isometry label for each partition cell. In general, however, the labeling $m$ is many-to-one. To invert the labeling we will exploit the relationship between the branch group of a reduced trellis section and its isometry and Euclidean labelings.

The branch group $B$ of a reduced group trellis at time $k$ is the set of all pairs $(\sigma_k, \sigma_{k+1}) \in \Sigma_k \times \Sigma_{k+1}$ such that a branch connects state $\sigma_k$ to $\sigma_{k+1}$,
where $\Sigma_k$ and $\Sigma_{k+1}$ are the state groups at times $k$ and $k+1$, respectively.

**Definition 5.2:** A *Euclidean labeling* of a branch group $B$ is a mapping $n: B \rightarrow Y$ from branches to the cells of a partition $Y$. A *permutation labeling* of $B$ is a homomorphism $l$ from $B$ to the group of permutations of the cells of $Y$. A Euclidean labeling $n$ and a permutation labeling $l$ are consistent if $l(b)x = n(b)$ for all $b \in B$, where $x = n(0)$ is the partition cell assigned to the identity branch 0.

Consistent labelings of branch groups and isometric labelings of partitions describe equivalent concepts. If $l: B \rightarrow U/U'$ is a permutation labeling of the branches of $B$ with cosets of isometries, and if $m: U/U' \rightarrow Y$ is an isometric labeling of $Y$, then the Euclidean labeling $n: B \rightarrow Y$ defined by $n(b) = m(l(b))$ is consistent with $l$. In other words, the induced Euclidean labeling of an isometry trellis is consistent.

The following theorem relates the structure of the branch group to the structure of consistent Euclidean and permutation labelings. It is our main tool for realizing Euclidean codes as isometry codes.

**Theorem 5.2:** Let $B$ be a reduced branch group with a Euclidean labeling $n: B \rightarrow Y$, let $x$ be the Euclidean label of the identity branch, and for $y \in Y$ let $B_y$ denote the set of branches labeled $y$. There exists a consistent permutation labeling $l$ of $B$ if and only if the following two conditions hold:

i. the set $B_x$ of branches labeled $x$ is a subgroup of $B$, and

ii. two branches have the same Euclidean label if any only if they are in the same left coset of $B_x$ in $B$.

When it exists, the consistent permutation labeling $l$ is uniquely defined for all $b \in B$ and $y, z \in Y$ by $l(b)y = z$ if $bB_y = B_z$.

**Proof:** Assume there exists a consistent permutation labeling $l$. Then the Euclidean label of a branch $b \in B$ is $l(b)x$, and a branch $b$ has Euclidean label $x$ if and only if $l(b)$ sends $x$ to $x$. The set of branches $B_x$ labeled $x$ is therefore the subgroup of $B$ that stabilizes $x$ under the group action defined by $l$.

To prove (ii), observe that if $b$ and $b'$ are two branches in the same left coset of $B_x$, then $b^{-1}b'$ is in $B_x$. Consequently $l(b^{-1}b') = l(b)^{-1}l(b')$ fixes $x$, so if $l(b')$ sends $x$ to some cell $y$ then $l(b)^{-1}$ must send $y$ to $x$. Branches $b$ and $b'$ therefore both have the same Euclidean label $l(b)x = l(b')x = y$. Reversing
5.3. ISOMETRY AND EUCLIDEAN TRELLIS SECTIONS

this reasoning proves that two branches with the same Euclidean label are in the same left coset of \( B_x \) in \( B \).

For the last part of the theorem, given a consistent permutation labeling \( l \) we know for all \( y, z \in Y \) that \( l(B_y)x = y \) and \( l(B_z)x = z \), hence if \( bB_y = B_z \) then \( l(b)y = l(b)l(B_y)x = l(bB_y)x = l(B_z)x = z \).

Conversely, assume the conditions (i) and (ii) hold. Define a mapping \( l \) from \( B \) to the group of permutations of \( Y \) by \( l(b)y = z \) when \( bB_y = B_z \). We must show that \( l \) is a homomorphism consistent with the Euclidean labeling. For all \( b, b' \in B \) and \( y \in Y \), select \( z \) and \( w \) such that \( bB_y = B_z \) and \( b'B_z = B_w \). Then \( b'bB_y = B_w \), hence \( w = l(b'b)y = l(b')l(b)y \). The map \( l \) is therefore a homomorphism. It is also consistent: for all \( b \in B \) if \( bB_x = B_z \) then \( l(b)x = z \), and \( bB_x = B_z \) implies that \( b \in B_z \) has Euclidean label \( z \). ■

The theorem shows how to find isometry code realizations of Euclidean codes. Given a reduced Euclidean trellis section, assign a branch group \( B \) to the trellis that satisfies conditions (i) and (ii) of Theorem 5.2. Construct the unique consistent permutation labeling \( l \) of \( B \) according the provisions of the theorem. If \( l(B) \) contains only isometric permutations, then in light of discussion following Theorem 4.12 \( l \) induces a unique labeling of the branches of \( B \) with cosets of the canonical generating quotient \( \Gamma(Y)/\Gamma(Y)_Y \).

In short, to realize a Euclidean trellis as an isometry code we construct a branch group \( B \) and a consistent permutation labeling of \( B \), then show that the permutations are isometric. We take this approach in the next section to realize the V.32 trellis code as an isometry code.

Theorem 5.2 is most interesting when applied to nonabelian branch groups. When \( B \) is abelian, which is true if and only if the state groups of the trellis section are abelian, then a consistent isometric permutation labeling constructed from Theorem 5.2 is equivalent to an "ordinary" bijective isometric labeling in the sense of Forney [19]. The subgroup \( B_x \) of an abelian branch group \( B \) is normal, hence the image \( l(B_x) \) of \( B_x \) under the consistent permutation labeling \( l \) must be a normal subgroup of \( l(B) \) that fixes \( x \). A subgroup of \( l(B) \) that stabilizes a point \( x \) is normal if and only if it is trivial, which implies that every branch in \( B_x \) is assigned the identity permutation label. Thus, two branches in an abelian branch group \( B \) with a consistent permutation labeling have the same Euclidean label if and only if they have the same permutation label.

A special case of Theorem 5.2 applies to a group code \( C \) over a group \( G \) combined with a bijection \( m \) from \( G \) cells of a partition. This special case applies, for example, to any trellis code described as a binary linear
convolutional code followed by a mapping from output bits to partition cells. The branches of the canonical trellis of $C$ are labeled with elements of the output group $G$ via a homomorphism $l$. Because $m$ is a bijection and $l$ is a homomorphism, the set of branches $B_x$ with Euclidean label $x$ is the kernel of $l$, where $x = m((0))$ is the Euclidean label of the identity branch. The quotient $B/B_x$ is therefore isomorphic to the output group $G$, and two branches have the same output label if and only if the are in the same coset of $B_x$ in $B$.

The conditions of Theorem 5.2 are therefore satisfied for any group trellis that is combined with a bijective mapping $m$ from the output group to partition cells. The partition need not have any uniformity properties, and the mapping $m$ need not be isometric. The consistent permutation labeling defined by the theorem turns out to be the left regular representation of $G$ [49, p. 46].

5.4 Realizing existing codes as isometry codes

We now have the tools needed to realize a variety of trellis codes as isometry codes. Realizing an existing Euclidean code as an isometry code involves a certain amount of guesswork. The representation may or may not be unique, and the code was almost certainly not designed with a simple group structure in mind. Moreover, we conjecture that there exist geometrically uniform Euclidean codes that have no isometry code description.

Our approach uses the trellis section as the fundamental unit of analysis. We assume that the Euclidean code in question is described by a state-output realization or equivalently by a trellis diagram. The realization need not be minimal. If the Euclidean code is described in some other form, such as a set of parity check equations, then it must first be converted into a trellis. To find a isometry code that generates the Euclidean code, at each time $k$ the states of the Euclidean trellis are identified with the elements of a suitable state group, then the Euclidean branch labels are replaced by isometries.

The realization method proceeds along the following lines. The first step is to select a state-output realization of the Euclidean code to use for the rest of the procedure. This realization must satisfy the conditions of Theorem 5.1: if two states have one future in common then they have all futures in common; if they have one past in common then they have all pasts in common. Also, the state transition structure of the realization must correspond to that of a shift-register graph.
5.4. REALIZING EXISTING CODES AS ISOMETRY CODES

The second step is to verify that the partitioned signal sets used in the Euclidean code are geometrically uniform. The partitions must be generated by normal quotients $U/U'$ of isometries. If they are not, it may be possible to salvage the procedure by taking blocks of symbols several at a time or by selecting a different state-output realization.

The third step is to find a state group and branch group for the Euclidean trellis that satisfies the conditions of Theorem 5.2. This can be done either by inspection or through an analysis using derivative codes. The rotational invariance properties of the code and some simple group-theoretic arguments often rule out many candidate state groups, though the number of possibilities may still grow rapidly with the size of the trellis.

The final step in constructing an isometry code representation is to verify that the permutation labeling defined by Theorem 5.2 is isometric. If so, the permutations are replaced by cosets of isometries that have the same action on the partition cells.

We consider each of these steps in detail, using the V.32 code as a running example.

5.4.1 The V.32 code

The CCITT V.32 trellis code, described by Wei in [62], is used as a component in most high-speed commercial voice band modems. Buz [7] has shown that the code has a uniform distance profile over all code sequences. An isometry code realization will establish the stronger property of geometrical uniformity.

A reduced trellis section for the V.32 code is shown Figure 5.1. The trellis is past-induced, future-induced and minimal. The partition used in V.32 is the translate of $\mathbb{R}^2\mathbb{Z}/4\mathbb{Z}^2$ detailed in Example 4.7. The 8 subsets of the partition are labeled with the letters \{a,b,...,h\} as indicated in Figure 4.4 and in Table 4.1.

While the signal set specified in the V.32 standard is a 32 point subset of $\mathbb{R}^2 + (1,0)$ (V.32bis uses a 128-point subset), we will take the signal set to be infinite. Our analysis of the V.32 code is therefore approximate. The finite versions of the signal set used in V.32 and V.32bis are not geometrically uniform.

For convenience of reference, we have arbitrarily numbered the 8 states of the V.32 trellis 0,1,...,7. These labels appear along the left side of the
trellis in Figure 5.1. The state labels on the right side of the trellis are the elements of the dihedral group $D_4$; the motivation for this labeling will be evident shortly.

### 5.4.2 Shift register graphs

The state transition structure of a group trellis, ignoring output labeling, must correspond to a shift register graph in the sense defined in Section 3.4.2. Among other things, this implies that the number of branches leaving each state in a trellis section must be equal, and the number of branches entering each state must be equal. Further, if there are $n$ parallel branches from a state $s_k$ at time $k$ to a state $s_{k+1}$ at time $k + 1$, then all pairs of connected states at times $k$ and $k + 1$ must have $n$ parallel branches.

Ignoring parallel branches, finite state sets admit only finitely many distinct shift register trellis sections. These possibilities can be enumerated by
considering the decompositions of consecutive state sets into shared granules, taking care that the granule cardinalities divide the state set cardinalities. For time-invariant codes the test can usually be done by inspection.

Most trellis sections that appear in the literature are shift-register graphs. However, systems considered in the theory symbolic dynamics, such as trellis codes designed for channels with spectral nulls, do not in general have this regular structure. Such codes likely have no group representation in the sense developed here.

![Diagram](image)

Figure 5.2: Three time-invariant 8-state granule decompositions.

![Diagram](image)

Figure 5.3: Three 8-state shift-register trellis sections.

**Example 5.1:** Assume we wish to find all possible time-invariant 8-state irreducible trellis sections that are correspond to shift register graphs. We proceed as indicated in Figure 5.2. The figure indicates, in table form, three possible granule decompositions of the state sets. The trellis sections that correspond to these decompositions are shown in Figure 5.3. The time-$k$ entries of the rows of a table indicate the sizes of the granules $\Gamma_{[k-1,k]}$. 
\( \Gamma_{[k-1,k+1]} \) and \( \Gamma_{[k-1,k+2]} \). The rows are kept constant within a table for time invariance. The product of the granule sizes within the region indicated in each table gives the size of the state group at time \( k \). The product of the column entries at time \( k+1 \) is the number of edges leaving each state at time \( k \). It is not difficult to show that the tables describe the only three possible time invariant trellis sections with 8 states. \( \square \)

The trellis section of the V.32 code appears as Figure 5.3(b), hence the first test is passed.

5.4.3 Testing for geometrically uniform partitions

Once it is verified that the state-output realization of the Euclidean code corresponds to a shift-register graph, the next step is to check that the labeling of the Euclidean trellis at each time \( k \) induces geometrically uniform partitions of the signal set \( S \) at time \( k \). Specifically, let \( S' \) and \( S'' \) be the subsets \( S \) that appear as the Euclidean labels on any two sets of parallel branches. Then \( S' \) and \( S'' \) must be either equal or disjoint. When this condition is satisfied the parallel branches of the trellis induce a partition \( Y \) of \( S \).

We require that the partition be geometrically uniform. Similarly, the sets of labels that appear on edges leaving each state must be subsets of a geometrically uniform partition, as must the sets of labels that appear on edges entering each state. Normal subgroups appear in many places in an isometry code—each must induce a geometrically uniform partition.

As a practical matter, the partitions of integer lattices and PSK-type signal sets commonly used in trellis codes are almost always geometrically uniform. (See Forney [19].)

A partition \( Y \) of \( S \) is geometrically uniform if its partition symmetry group \( \Gamma(Y) \) is transitive on \( S \). To verify that the partition is normal, it is necessary and sufficient to show that the partition stabilizer \( \Gamma(Y)_Y \) is transitive on the points in each partition cell.

These conditions are met for the V.32 code, as established in Example 4.7. We therefore take the generating group \( U \) to be \( \Gamma(Y) \) and the normal subgroup \( U'' \) to be \( \Gamma(Y)_Y \). The quotient group \( U/U'' \) then has 64 elements.

Once the geometrically uniform partition induced by parallel transitions is identified, the Euclidean trellis is simplified to a reduced code by replacing parallel branches by a single branch labeled with a partition cell.
5.4.4 Finding a branch group

Finding a branch group $B$ that satisfies the conditions of Theorem 5.2 is the most difficult step in the realization process. There are two ways to proceed. The best approach is to guess the right answer then prove that it works. A second and more tiresome solution is to use the theory of group extensions and derivative codes (as in Loeliger and Mittelholzer [40]) to construct and test all possible branch groups against Theorem 5.2. The second approach must be used when proving that no valid branch group exists.

The V.32 trellis admits only a handful of possible branch groups, hence the “guess and test” method is most appropriate. We will demonstrate this method here. Enumerating group extensions and derivative codes is best done with aid of a computer, though even then a special case analysis may be required to reduce the number of possibilities to a manageable size.

The first step in both methods is to identify a branch of the Euclidean trellis as the identity element of the branch group $B$. Any branch will do, but to preserve time invariance a horizontal branch must be selected. We select the identity branch of the V.32 trellis to be the branch from state 0 to state 0 labeled with the partition cell $a$.

Choosing the identity branch has the side effect of determining the isometric labeling $m$ from the cosets of the output group $U/U'$ to the partition cells of $Y$. The isometry label on the identity branch of $B$ is always the identity coset $U'$, hence for the V.32 code $m(U') = a$. The map $m$ is therefore defined by $m(uU') = uU'a$ for all $uU' \in U/U'$. Equivalently, a branch with Euclidean label $x$ must have an isometry label that sends $a$ to $x$.

We are now ready to proceed with the “guess and test” method. Up to isomorphism, there are five possible state groups with 8 elements: $\mathbb{Z}_8$, $\mathbb{Z}_4 \times \mathbb{Z}_2$, $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_2$, the dihedral group $D_4$, and the quaternions $H$. We will eliminate possibilities one at a time.

We eliminate $\mathbb{Z}_8$ because a trellis section whose state group is $\mathbb{Z}_8$ must be either fully connected or uncontrollable. The binary linear state group $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_2$ has already been discarded as inadequate by Wei and others.

Let $(x, y)$ denote a branch from state $x$ to state $y$. We know from Theorem 5.2 that the four branches $(0,0), (1,2), (4,1)$ and $(5,3)$ labeled with the cell $a$ must be a subgroup $B_a$ of the branch group $B$. Up to isomorphism, $B_a$ is isomorphic to either $\mathbb{Z}_2 \times \mathbb{Z}_2$ or $\mathbb{Z}_4$. The possibility $B_a = \mathbb{Z}_4$ is eliminated by examining the perfect squares of the state group.

A element $g$ of a group $G$ is a perfect square if $g = xx$ for some $x \in G$. We
CHAPTER 5. ISOMETRY CODES

\[
\begin{array}{cccc|cccc}
\rho_0 & \rho_1 & \rho_2 & \rho_3 & \mu_1 & \mu_2 & \delta_1 & \delta_2 \\
\rho_0 & \rho_0 & \rho_1 & \rho_2 & \rho_3 & \mu_1 & \mu_2 & \delta_1 & \delta_2 \\
\rho_1 & \rho_1 & \rho_2 & \rho_3 & \rho_0 & \delta_2 & \delta_1 & \mu_1 & \mu_2 \\
\rho_2 & \rho_2 & \rho_3 & \rho_0 & \rho_1 & \mu_2 & \mu_1 & \delta_2 & \delta_1 \\
\rho_3 & \rho_3 & \rho_0 & \rho_1 & \rho_2 & \delta_1 & \delta_2 & \mu_2 & \mu_1 \\
\mu_1 & \mu_1 & \mu_2 & \mu_2 & \delta_2 & \rho_0 & \rho_2 & \rho_1 & \rho_3 \\
\mu_2 & \mu_2 & \delta_2 & \mu_1 & \delta_1 & \rho_2 & \rho_0 & \rho_3 & \rho_1 \\
\delta_1 & \delta_1 & \mu_2 & \delta_2 & \mu_1 & \rho_3 & \rho_1 & \rho_0 & \rho_2 \\
\delta_2 & \delta_2 & \mu_1 & \delta_1 & \mu_2 & \rho_1 & \rho_3 & \rho_2 & \rho_0 \\
\end{array}
\]

Table 5.1: Group table for \( D_4 \), the symmetries of the square.

Claim that the quaternions, \( D_4 \) and \( \mathbb{Z}_4 \times \mathbb{Z}_2 \) have only two perfect squares, one of which is the identity. The square of an element of order 2 is the identity, hence we need only examine the elements of order 4. The two elements of order 4 in \( D_4 \) are \( \rho_1 \) and \( \rho_3 \). The square of both elements is \( \rho_2 \). Similarly, the square of any element of \( \mathbb{Z}_4 \times \mathbb{Z}_2 \) is either \((2,0)\) or \((0,0)\), and the square of any element of the quaternions is either 1 or \(-1\).

Assume \( B_a \) is isomorphic to \( \mathbb{Z}_4 \). Then there must be a branch \((x,y)\) labeled a whose order is 4. Both endpoints \( x \) and \( y \) must also have order 4; if (say) \( x \) has order 2 then \((x,y)(x,y) = (0,yy)\) is a non-identity branch with label a that leaves state 0, a contradiction. The product \((xx,yy)\) of this branch with itself must therefore have non-identity endpoints of order 2. The paucity of perfect squares established in the previous paragraph implies that \( xx = yy \), hence \((xx,yy)\) is a non-identity horizontal branch labeled a. There is no such branch in the trellis. We conclude that the a-labeled branch group \( B_a \) is isomorphic to \( \mathbb{Z}_2 \times \mathbb{Z}_2 \).

Since \( B_a \) is isomorphic to \( \mathbb{Z}_2 \times \mathbb{Z}_2 \), every state that has an entering or exiting a-labeled branch must have order 2 or 1. By inspecting Figure 5.1, we see that the branches of \( B_a \) touch the six states \( \{0,1,2,3,4,5\} \). The state group \( \mathbb{Z}_4 \times \mathbb{Z}_2 \) has four elements of order 4, while the quaternions \( H \) have six elements of order 4. Both are therefore eliminated.

The group table of \( D_4 \), the only remaining possibility, is shown in Table 5.1.

Assigning the elements of the state group \( D_4 \) to the states of the trellis section determines the branch group \( B \). The state group can be assigned to the trellis in several ways that satisfy Theorem 5.2, though for this code all possibilities turn out to be equivalent up to automorphism.

State 0 is the identity state \( \rho_0 \). From the previous analysis, we know that
5.4. REALIZING EXISTING CODES AS ISOMETRY CODES

states 6 and 7 have order 4. Assign $\rho_1$ to state 6 and $\rho_3$ to state 7. The horizontal branch $(\rho_1, \rho_1)(\rho_1, \rho_1) = (\rho_2, \rho_2)$ must be $(1, 1)$, so we assign $\rho_2$ to state 1. Our next choice is unconstrained: assign $\mu_1$ to state 2. The set of branches leaving state $\rho_0$ must a subgroup, hence we assign $\mu_2$ to state 3.

According to Theorem 5.2 the branches in the left coset $(\mu_2, \mu_3)B_a$ of $B//B_a$ must all have label $h$, hence $(\mu_2, \mu_3)(\rho_2, \mu_1) = (\mu_1, \delta_1)$ has label $h$. We therefore assign $\delta_1$ to state 4 and $\delta_2$ to state 5.

The final assignment is displayed along the right edge of Figure 5.1. It can be verified that the branch group $B$ induced by this assignment is in fact a group: the product of any two branches is another branch.

To test $B$ against Theorem 5.2 we must show that two branches have the same Euclidean label if and only if they are in the same left coset of $B_a$ in $B$. The branch group $B$ is partitioned into left cosets of $B_a$ as follows, where $B_x$ denotes the set of edges that have subset label $x$:

\[
B_a = \{(\rho_0, \rho_0), (\rho_2, \mu_1), (\delta_1, \rho_2), (\delta_2, \mu_2)\}
\]
\[
B_b = (\rho_0, \rho_2)B_a = \{(\rho_0, \rho_2), (\rho_2, \mu_2), (\delta_1, \rho_0), (\delta_2, \mu_1)\}
\]
\[
B_c = (\rho_0, \mu_2)B_a = \{(\rho_0, \mu_2), (\rho_2, \rho_2), (\delta_1, \mu_1), (\delta_2, \rho_0)\}
\]
\[
B_d = (\rho_0, \mu_1)B_a = \{(\rho_0, \mu_1), (\rho_2, \rho_0), (\delta_1, \mu_2), (\delta_2, \rho_2)\}
\]
\[
B_e = (\rho_1, \rho_1)B_a = \{(\rho_1, \rho_1), (\rho_3, \delta_2), (\mu_1, \rho_3), (\mu_2, \delta_1)\}
\]
\[
B_f = (\rho_1, \rho_3)B_a = \{(\rho_1, \rho_3), (\rho_3, \delta_1), (\mu_1, \rho_1), (\mu_2, \delta_2)\}
\]
\[
B_g = (\rho_1, \delta_1)B_a = \{(\rho_1, \delta_1), (\rho_3, \rho_3), (\mu_1, \delta_2), (\mu_2, \rho_1)\}
\]
\[
B_h = (\rho_1, \delta_2)B_a = \{(\rho_1, \delta_2), (\rho_3, \rho_1), (\mu_1, \delta_1), (\mu_2, \rho_3)\}
\]

Comparison with Figure 5.1 establishes that two branches have the same Euclidean label if and only if they are in the same left coset of $B_a$ in $B$.

5.4.5 The permutation labeling

We can now find the consistent permutation labeling of $B$ defined in the last part of Theorem 5.2. It will remain to verify that the permutation labels are isometric.

It suffices to find permutation labels for a set of generators of the branch group $B$, so that every branch can be expressed as a finite product of generators and their inverses. For the current example, we choose as generators the four branches $(\rho_0, \rho_2), (\rho_0, \mu_1), (\rho_1, \rho_1)$, and $(\mu_1, \delta_2)$. The left translation action of these generators on the left cosets of $B_a$ is shown in Table 5.2. The entries along a row of the table indicate the action of a generator branch on the coset of the corresponding column. For example, we compute the entry
Table 5.2: The action of generator branches on the cosets of $B_a$

for row $(\mu_1, \delta_2)$ and column $B_e$ as

\[
(\mu_1, \delta_2)B_e = (\mu_1, \delta_2)(\rho_1, \rho_1)B_a \\
= (\delta_1, \mu_1)B_a \\
= B_c.
\] (5.1)
(5.2)
(5.3)

The remaining entries in the table are computed similarly.

If branch $b$ sends coset $B_y$ to coset $bB_y = B_z$, then it is assigned a permutation $l(b)$ that sends $y$ to $z$. Reading across the rows of Table 5.2, we conclude that

\[
l(\rho_0, \rho_2) = (a b)(c d)(e f)(g h), \quad (5.4) \\
l(\rho_0, \mu_1) = (a d)(b c)(e g)(f h), \quad (5.5) \\
l(\rho_1, \rho_1) = (a e c g)(b f d h), \quad \text{and} \quad (5.6) \\
l(\mu_1, \delta_2) = (a g)(b h)(c e)(d f), \quad (5.7)
\]

where the permutations are written in standard cycle notation.

For the sake of completeness we display in Table 5.3 the permutation labeling of every branch of $B$. Comparing the table with Figure 5.1 confirms that $l$ is consistent: a branch $b$ with Euclidean label $x$ is assigned a permutation $l(b)$ that sends $a$ to $x$.

The final step is to verify that the permutations used in the labeling are induced by isometries. Only the four generator branches need to be checked. The required isometries can be found by inspection. Referring to the signal set in Figure 4.4, translation by $(2, 2)$ has the same action on the subsets of $\mathbb{R}Z^2/4\mathbb{Z}^2$ as the permutation $l(\rho_0, \rho_2) = (a b)(c d)(e f)(g h)$, translation by $(2, 0)$ has the same action as $l(\rho_0, \mu_1) = (a d)(b c)(e g)(f h)$, rotation about the origin by 90 degrees clockwise has the same action as $l(\rho_1, \rho_1) = (a e c g)(b f d h)$, and reflection about the line $x = -y$ has the same action as $l(\mu_1, \delta_2) = (a g)(b h)(c e)(d f)$.

Since the permutation labels are isometric, to form the final isometry code we need only replace each permutation label with the coset of $\Gamma(Y)/\Gamma(Y)_Y$ that has the same permutation action on $Y$. 
Table 5.3: Permutation labeling for the 32 edges of the V.32 code.

<table>
<thead>
<tr>
<th>Edge</th>
<th>Label</th>
<th>Edge</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\rho_0, \rho_0)$</td>
<td>()</td>
<td>$(\mu_1, \delta_2)$</td>
<td>$(a g)(b h)(c e)(d f)$</td>
</tr>
<tr>
<td>$(\rho_0, \rho_2)$</td>
<td>$(a b)(c d)(e f)(g h)$</td>
<td>$(\mu_1, \delta_1)$</td>
<td>$(a h)(b g)(c f)(d e)$</td>
</tr>
<tr>
<td>$(\rho_0, \mu_1)$</td>
<td>$(a d)(b c)(e g)(f h)$</td>
<td>$(\mu_1, \rho_1)$</td>
<td>$(a f b e)(c h d g)$</td>
</tr>
<tr>
<td>$(\rho_0, \mu_2)$</td>
<td>$(a c)(b d)(e h)(f g)$</td>
<td>$(\mu_1, \rho_3)$</td>
<td>$(a e b f)(c g d h)$</td>
</tr>
<tr>
<td>$(\rho_1, \rho_1)$</td>
<td>$(a e c g)(b f d h)$</td>
<td>$(\delta_1, \mu_1)$</td>
<td>$(a c)(b d)$</td>
</tr>
<tr>
<td>$(\rho_1, \rho_3)$</td>
<td>$(a f c h)(b e d g)$</td>
<td>$(\delta_1, \mu_2)$</td>
<td>$(a d)(b c)(e f)(g h)$</td>
</tr>
<tr>
<td>$(\rho_1, \delta_2)$</td>
<td>$(a h d e)(b g c f)$</td>
<td>$(\delta_1, \rho_0)$</td>
<td>$(a b)(c d)(e g)(f h)$</td>
</tr>
<tr>
<td>$(\rho_1, \delta_1)$</td>
<td>$(a g d f)(b h c e)$</td>
<td>$(\delta_1, \rho_2)$</td>
<td>$(e h)(f g)$</td>
</tr>
<tr>
<td>$(\rho_2, \rho_2)$</td>
<td>$(a c)(b d)(e g)(f h)$</td>
<td>$(\mu_2, \delta_1)$</td>
<td>$(a e)(b f)(c g)(d h)$</td>
</tr>
<tr>
<td>$(\rho_2, \rho_0)$</td>
<td>$(a d)(b c)(e h)(f g)$</td>
<td>$(\mu_2, \delta_2)$</td>
<td>$(a f)(b e)(c h)(d g)$</td>
</tr>
<tr>
<td>$(\rho_2, \mu_2)$</td>
<td>$(a b)(c d)$</td>
<td>$(\mu_2, \rho_3)$</td>
<td>$(a h b g)(c f d e)$</td>
</tr>
<tr>
<td>$(\rho_2, \mu_1)$</td>
<td>$(e f)(g h)$</td>
<td>$(\mu_2, \rho_1)$</td>
<td>$(a g b h)(c e d f)$</td>
</tr>
<tr>
<td>$(\rho_3, \rho_3)$</td>
<td>$(a g c e)(b h d f)$</td>
<td>$(\delta_2, \mu_2)$</td>
<td>$(e g)(f h)$</td>
</tr>
<tr>
<td>$(\rho_3, \rho_1)$</td>
<td>$(a h c f)(b g d e)$</td>
<td>$(\delta_2, \mu_1)$</td>
<td>$(a b)(c d)(e h)(f g)$</td>
</tr>
<tr>
<td>$(\rho_3, \delta_1)$</td>
<td>$(a f d g)(b e c h)$</td>
<td>$(\delta_2, \rho_2)$</td>
<td>$(a d)(b c)$</td>
</tr>
<tr>
<td>$(\rho_3, \delta_2)$</td>
<td>$(a e d h)(b f c g)$</td>
<td>$(\delta_2, \rho_0)$</td>
<td>$(a c)(b d)(e f)(g h)$</td>
</tr>
</tbody>
</table>

The isometry code may also be characterized in purely geometric terms, as the set of all products (function compositions) of all shifts and inverses of the length-3 sequences of isometries displayed in Figure 5.4. The origin is indicated by a circle; successive terms in the sequence are separated by commas. Missing terms are the identity.

Figure 5.4: Isometry generator sequence for the V.32 code.

Several comments are in order. The V.32 Euclidean code is 3-observable and the isometry code is 1-observable. The 32 branches of the trellis each have a distinct isometry label, yet there are only 8 Euclidean cells in the geometrically uniform partition. The isometric labeling from isometries to cells is therefore many-to-one. Note, however, that the map from isometry sequences $c$ to Euclidean sequences $ca$ is a bijection.
Though Forney's original framework [19] cannot represent the V.32 code using the existing time axis, the picture changes if symbols are taken in blocks of 3 at a time. The resulting Euclidean code has a 6-dimensional signal set and is 1-observable. Distinct branches have distinct Euclidean labels, hence an isometric labeling for this code is necessarily one-to-one. In general, if \( \mu \) is the observability index of the Euclidean code, then many-to-one isometric labelings yield one-to-one isometric labelings when symbols are taken in blocks of length \( \mu \).

The isometry code realization makes the rotational invariance of the V.32 code explicit: there is a horizontal branch \( (\rho_1, \rho_1) \) labeled with 90-degree rotation. It can be shown that this is a sufficient condition for an isometry code to generate a 4-way rotationally invariant Euclidean code [57].

### 5.5 Two more examples

We close the chapter with two further examples of the process of realizing trellis codes as isometry codes. The first example demonstrates that a representation of a trellis code as a linear system over a finite field followed by a mapping to cells of a geometrically uniform partition does not automatically establish that the code is geometrically uniform. The mapping must be shown to be isometric.

The second example shows that a Euclidean trellis with a consistent permutation labeling may require permutations that are not isometries.

**Example 5.2:** Wei's 16-state 4-dimensional trellis code is described in [63]. The code is used in high-speed modems, and a minor variant is being considered for the forthcoming CCITT V.fast modem standard. We will prove that the code is geometrically uniform by constructing an isometry code representation.

Forney [19, p. 1242] implies that the code is geometrically uniform, but his conclusion appears to be based on the incorrect assumption that a binary linear code combined with a mod-2 lattice partition is a linear Euclidean-space code. This is true only if the mapping from encoder output bits to lattice codes is well behaved (isometric).

The signal set for the code is a translate of the four-dimensional integer lattice \( \mathbb{Z}^4 \). The symmetry group \( \Gamma(\mathbb{Z}^4) \) is the semidirect product of the group of translations by \( \mathbb{Z}^4 \) (the lattice group) and the group of coordinate permutations and sign changes (the point group).
The signal set is partitioned into 8 translates of \( \mathbf{R}D_4 = 2\mathbf{Z}^4 \cup (2\mathbf{Z}^4 + (1,1,1,1)) \). (Here \( D_4 \) denotes the four-dimensional checkerboard lattice, not the dihedral group.) The partition can be viewed as a coarsening of the mod-2 lattice partition of \( \mathbf{Z}^4 \) into 16 translates of \( 2\mathbf{Z}^4 \).

The symmetry group of the 16-way partition is equal to the symmetry group of the signal set, hence the partition is unambiguously denoted by \( Y = \mathbf{Z}^4/2\mathbf{Z}^4 \). The partition stabilizer \( \Gamma(Y)_Y \) is the semidirect product of the group of translations by \( 2\mathbf{Z}^4 \) and the group of \( 2^4 \) coordinate sign changes. The quotient group \( \Gamma(Y)/\Gamma(Y)_Y \) is therefore the semidirect product of the quotient translation group \( \mathbf{Z}^4/2\mathbf{Z}^4 \cong (\mathbf{Z}_2)^4 \) by the group of \( 4! \) coordinate permutations, for a total for \( 2^44! = 384 \) cosets of isometries. It is isomorphic to the group of symmetries of the 4-cube.

The partition stabilizer of the 8-way partition \( Y = \mathbf{Z}^4/\mathbf{R}D_4 \) is larger than the partition stabilizer of \( \mathbf{Z}^4/2\mathbf{Z}^4 \). It is the semidirect product of the group of translations by \( \mathbf{R}D_4 = 2\mathbf{Z}^4 \cup (2\mathbf{Z}^4 + (1,1,1,1)) \) and the group of \( 2^4 \) coordinate sign changes. The quotient group \( \Gamma(Y)/\Gamma(Y)_Y \) is therefore isomorphic to the semidirect product of \( \mathbf{Z}^4/\mathbf{R}D_4 \cong (\mathbf{Z}_2)^3 \) by the group of \( 4! \) coordinate permutations, for a total of 192 cosets. We will need only 8.

The code is described by a linear system over the finite field \( \text{GF}(2^3) \cong (\mathbf{Z}_2)^3 \) followed by a bijective mapping from output elements to partition cells. From the discussion at the end of Section 5.3, we know that the conditions of Theorem 5.2 are satisfied automatically. Furthermore, since the output group \( (\mathbf{Z}_2)^3 \) is abelian, to prove that the code is geometrically uniform we need only find one coset of \( \Gamma(Y)/\Gamma(Y)_Y \) for each element of \( (\mathbf{Z}_2)^3 \).

A sequence \( \mathbf{c} \in ((\mathbf{Z}_2)^3)^\mathbf{Z} \) is in the code if and only if it satisfies the binary linear parity check equation

\[
0 = (1 + D^3 + D^4)C_0(D) + DC_1(D) + (D^2 + D^3)C_2(D),
\]

where \( C(D) = (C_0(D), C_1(D), C_2(D)) \) is the \( D \)-transform of \( \mathbf{c} \).

The bijection from the output group \( (\mathbf{Z}_2)^3 \) to the cells of the partition \( \mathbf{Z}^4/\mathbf{R}D_4 + (1/2,1/2,1/2,1/2) \) is given in Table 5.4. The table is an adaptation of Table 1 of [63]. For convenience of reference the cells are numbered 0,1,\ldots,7.

This code is linear over a field, but it has an unusual mapping from field elements to cosets. We must show that the mapping is isometric. That is, for each element \( g \) of \( (\mathbf{Z}_2)^3 \) we must find an isometric permutation in \( \Gamma(Y)/\Gamma(Y)_Y \) whose action on the partition cells corresponds to the left translation action of \( g \) on \( (\mathbf{Z}_2)^3 \). It suffices to find isometric permutations for a set of generators of \( (\mathbf{Z}_2)^3 \).
Table 5.4: The correspondence between \((Z_2)^3\) and the partition cells of \(\mathbb{Z}^4/\mathbb{R}D_4\).

<table>
<thead>
<tr>
<th>label</th>
<th>partition cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(\mathbb{R}D_4 + (1/2)(1,1,1,1))</td>
</tr>
<tr>
<td>1</td>
<td>(\mathbb{R}D_4 + (1/2)(3,1,3,1))</td>
</tr>
<tr>
<td>2</td>
<td>(\mathbb{R}D_4 + (1/2)(3,3,1,1))</td>
</tr>
<tr>
<td>3</td>
<td>(\mathbb{R}D_4 + (1/2)(1,3,3,1))</td>
</tr>
<tr>
<td>4</td>
<td>(\mathbb{R}D_4 + (1/2)(3,3,3,1))</td>
</tr>
<tr>
<td>5</td>
<td>(\mathbb{R}D_4 + (1/2)(3,1,1,1))</td>
</tr>
<tr>
<td>6</td>
<td>(\mathbb{R}D_4 + (1/2)(1,1,3,1))</td>
</tr>
<tr>
<td>7</td>
<td>(\mathbb{R}D_4 + (1/2)(1,3,1,1))</td>
</tr>
</tbody>
</table>

Translation by \((0,0,1)\) has the following action on the output group:

\[
(0,0,0) \mapsto (0,0,1), \quad (0,0,1) \mapsto (0,0,0), \\
(0,1,0) \mapsto (0,1,1), \quad (0,1,1) \mapsto (0,1,0), \\
(1,0,0) \mapsto (1,0,1), \quad (1,0,1) \mapsto (1,0,0), \\
(1,1,0) \mapsto (1,1,1), \quad (1,1,1) \mapsto (1,1,0).
\] (5.9)

Referring to Table 5.4, the coset of isometries assigned to \((0,0,1)\) must therefore have the permutation action \((0\,1\,2\,3\,4\,5\,6\,7)\) on the partition cells. A similar computation shows that \((0,1,0)\) must be assigned the permutation \((0\,2\,1\,3\,4\,6\,5\,7)\), and that \((1,0,0)\) must be assigned the permutation \((0\,4\,1\,5\,2\,6\,3\,7)\).

Finding isometries that induce these permutations is not difficult. We take as a system of coset representatives for the quotient group \(\Gamma(Y)/\Gamma(Y)\) the set of 192 affine transformations \(x \mapsto Px + t\) such that \(P\) is a permutation matrix and \(t \in (Z_2)^4\) is a translation vector with a 0 in the last coordinate.

A computer search is the easiest way to find a representative isometry \(x \mapsto Px + t\) for each permutation: simply test all 192 representative isometries to see if any has the desired action. We will demonstrate a more constructive pencil-and-paper approach.

The isometry \(x \mapsto Px + t\) assigned to the permutation \((0\,1\,2\,3\,4\,5\,6\,7)\) must send points in cell 0 to points in cell 1. In particular, \(y = (1/2)(1,1,1,1)\) is sent to a point in \(\mathbb{R}D_4 + (1/2)(3,1,3,1)\). Since \(Py = y\) for any permutation matrix \(P\), the translation \(t \in (Z_2)^4\) must be \((1,0,1,0)\). More generally, if the isometry sends a point \(y\) to a cell \(\mathbb{R}D_4 + z\), then \(2Py\) must be a point in
2RD₄ + 2(z − t). Applying this equation to each cell in Table 5.4 yields

\[
P(1, 1, 1, 1) \in 2RD₄ + (1, 1, 1, 1),
\]
\[
P(3, 1, 3, 1) \in 2RD₄ + (3, 1, 3, 1),
\]
\[
P(3, 3, 1, 1) \in 2RD₄ + (3, 3, 1, 1),
\]
\[
P(1, 3, 3, 1) \in 2RD₄ + (1, 3, 3, 1),
\]
\[
P(3, 3, 3, 1) \in 2RD₄ + (1, 1, 3, 1) = 2RD₄ + (3, 3, 1, 3),
\]
\[
P(3, 1, 1, 1) \in 2RD₄ + (1, 3, 1, 1),
\]
\[
P(1, 1, 3, 1) \in 2RD₄ + (3, 3, 1, 1) = 2RD₄ + (1, 1, 1, 3),
\]
\[
P(1, 3, 1, 1) \in 2RD₄ + (3, 1, 1, 1).
\]

The unique permutation matrix that satisfies these constraints is

\[
P = \begin{bmatrix}
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
\end{bmatrix}.
\] (5.10)

The permutation matrix \(P\) exchanges the first two coordinates and exchanges the last two coordinates, hence the composite isometry \(x \mapsto Px + t\) is a 90 degree clockwise rotation about the origin in the first two and last two coordinates.

Similarly, the permutation \((0\ 2)(1\ 3)(4\ 6)(5\ 7)\) is induced by the isometry

\[
x \mapsto \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix} x + \begin{bmatrix}
1 \\
1 \\
0 \\
0 \\
\end{bmatrix},
\] (5.11)

which is a translation by \((1, 1, 0, 0)\), and the permutation \((0\ 4)(1\ 5)(2\ 6)(3\ 7)\) is induced by

\[
x \mapsto \begin{bmatrix}
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix} x + \begin{bmatrix}
1 \\
1 \\
1 \\
0 \\
\end{bmatrix},
\] (5.12)

which is a reflection across the line \(x = y\) in the first two coordinates followed by a translation by \((1, 1, 1, 0)\).

These three cosets of isometries generate a subgroup of \(\Gamma(Y)/\Gamma(Y)\) that contains 8 elements, is isomorphic to \((\mathbb{Z}_2)^2\), and is transitive on \(Y\). We have therefore found the desired isometric labeling.

Note that this labeling isn't a mod-2 labeling [16]. Different binary labelings of a mod-2 partition yield completely different classes of Euclidean code.
when combined with binary linear convolutional codes. In particular, Wei's code has no binary linear description with a mod-2 binary labeling, but it can be described using a nonlinear binary convolutional code.

The code is 4-way rotationally invariant because the horizontal branch labeled \((0,0,1)\) in the trellis section of the binary linear code is assigned a coset of isometries that contains 90 degree clockwise rotation in the first two and last two coordinates. □

Example 5.3: Chouly and Sari describe a six-dimensional 16-state "nonlinear" rotationally invariant code in [10]. The code is based on a translate of the 16-way lattice partition \(\mathbb{Z}^6/\mathbf{RD}_6\), where

\[
\mathbf{RD}_6 = (2\mathbb{Z}^6 + (0,0,0,0,0,0)) \cup (2\mathbb{Z}^6 + (0,0,1,1,1,1)) \cup (2\mathbb{Z}^6 + (1,1,0,0,1,1)) \cup (2\mathbb{Z}^6 + (1,1,1,1,0,0))
\]  

(5.13)

is the union of four translates of \(2\mathbb{Z}^6\). We find a branch group for the code that has a consistent permutation labeling, but the permutations are not isometric.

The code is described as the set of output sequences of a nonlinear binary encoder that has five input bits \((x_1,x_2,\ldots,x_5)\) and six output bits \((y_1,y_2,\ldots,y_6)\) at each time \(k\). In \(D\)-transform notation, the encoder is defined by the equations

\[
Y_1 = X_3 \oplus X_1D \oplus X_1D^2, \quad (5.14)
\]

\[
Y_2 = X_4, \quad (5.15)
\]

\[
Y_3 = X_2 \oplus X_2D \oplus X_1D \oplus X_1D^2 \oplus (X_1D \otimes X_1), \quad (5.16)
\]

\[
Y_4 = X_5, \quad (5.17)
\]

\[
Y_5 = X_3 \oplus X_2 \oplus X_2D \oplus X_1D \oplus (X_1D \otimes X_1), \quad (5.18)
\]

\[
Y_6 = X_4 \oplus X_5 \oplus X_1 \oplus X_1D \oplus X_1D^2 \oplus X_2D \oplus X_3D, \quad (5.19)
\]

where \(\oplus\) denotes binary addition and \(\otimes\) denotes binary multiplication.

Output bits are mapped to cells of \(\mathbb{Z}^6/\mathbf{RD}_6+(1/2)(1,1,1,1,1,1)\) as indicated in Table 5.5, where the vectors \((y_1,0,y_3,0,y_5,y_6)\), \((y_1,0,y_3,1,y_5,y_6\oplus1)\), \((y_1,1,y_3,0,y_5,y_6\oplus1)\), and \((y_1,1,y_3,1,y_5,y_6)\) map to the same translate of \(\mathbf{RD}_6\). The table is derived from Figure 1 of [10].

The equations describe a 16-state reduced trellis section with 8 branches leaving each state. We label the 16 states of trellis with binary 4-tuples such that the input sequence \(\mathbf{x} = (x_1,x_2,\ldots,x_5)\) drives the system to state \((x_{2,k},x_{3,k},x_{1,k-1},x_{1,k})\) at time \(k\). A pair of states denotes a branch of the
Table 5.5: The labeling of the cells of $\mathbb{Z}^6/\mathbf{R}D_6$.

trellis; for example, $(0000,0101)$ is the branch from state $(0,0,0,0)$ to state $(0,1,0,1)$.

Using Equations (5.14–5.19) and Table 5.5, we find that the eight trellis branches labeled with the partition cell 0 are

$$(0000,0000),\ (0100,0001),\ (1000,1001),\ (1100,1000),$$
$$(0010,1101),\ (0110,1100),\ (1010,0100),\ (1110,0101).$$

Similarly, we compute the set of branches leaving state $(0,0,0,0)$ as

$$(0000,0000),\ (0000,0001),\ (0000,0100),\ (0000,0101),$$
$$(0000,1000),\ (0000,1001),\ (0000,1100),\ (0000,1101).$$

We are now ready to find the branch group $B$. The zero state of the trellis must have a horizontal branch (self-transition). We select $(0,0,0,0)$ to be the zero state. Its partition label is 0.

The set of branches leaving the zero state is a subgroup of $B$. We guess that it is isomorphic to $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_2$. Under this assumption, every state reachable in one transition from the zero state has order 2. Also, if one endpoint (state) of a 0-labeled branch has order 2, then both must have order 2. The right endpoints of the 0-labeled branches for this code happen to be the
states reachable in one step from the zero state. All 0-labeled branches therefore have order 2, hence only the four states (0, 0, 1, 1), (0, 1, 1, 1), (1, 0, 1, 1), and (1, 1, 1, 1) that do not have an entering or exiting 0-labeled branch may have order greater than 2. A reasonable choice for the state group is thus the nonabelian group $D_4 \times Z_2$, which has exactly four elements of order 4.

Knowledge of the symmetries of the trellis diagram under rotation substantially simplifies the problem of assigning elements of the state group to trellis states. The states of the trellis are partitioned into orbits under 90 degree rotation as shown in Figure 5.5.

```
(0,0,0,0) -> (0,1,1,0) -> (1,0,0,1) -> (1,1,1,1)
(0,0,0,1) -> (1,1,1,0) -> (1,0,1,0) -> (0,1,1,0)
(0,0,1,0) -> (0,1,0,1) -> (1,0,1,0) -> (1,1,0,1)
(0,0,1,1) -> (1,1,0,0) -> (1,0,1,1) -> (0,1,0,0)
```

Figure 5.5: The orbits of states under 90 degree rotation.

A sufficient condition for rotational invariance is that the branch group contain a horizontal branch $(r, r)$ labeled with a coset of isometries that includes 90 degree rotation. The left translation action of the “rotator” state $r$ on the states of the trellis must induce the orbits given in Figure 5.5. In particular, the action of $r$ must send the zero state $(0, 0, 0, 0)$ to $r$, hence $r = (0, 1, 1, 1)$.

Elements of the candidate state group $D_4 \times Z_2$ are assignment to the states of the trellis as follows. Several different assignments are possible, though they all turn out to be equivalent up to automorphism of the state group.

We have already identified the identity state $(\rho_0, 0)$ as $(0, 0, 0, 0)$. Any element of order four may be assigned to the rotator state $r$; we assign $(\rho_1, 0)$ to $r = (0, 1, 1, 1)$. The left translation action of $r$ on $(0, 0, 0, 0)$ must induce the orbit given in Figure 5.5. This forces the assignment of $rr = (\rho_2, 0)$ to $(1, 0, 0, 0)$ and $rrr = (\rho_3, 0)$ to $(1, 1, 1, 1)$. Assigning an element of the state group to a representative trellis state from each of the three remaining orbits will completely specify the mapping from $D_4 \times Z_2$ to trellis states.

We continue to assume that the group of 0-labeled branches and the
5.5. TWO MORE EXAMPLES

group of branches leaving the zero state are isomorphic to \((Z_2)^3\). This can be achieved by assigning \((\mu_1, 0)\) to \((0, 0, 0, 1)\), \((\rho_0, 1)\) to \((0, 1, 0, 0)\), and \((\mu_1, 1)\) to \((0, 1, 0, 1)\). The rotation action of \(r\) then assigns \(r(\mu_1, 0) = (\delta_2, 0)\) to \((1, 1, 1, 0)\), \(rr(\mu_1, 0) = (\mu_2, 0)\) to \((1, 0, 0, 1)\), and \(rrr(\mu_1, 0) = (\delta_1, 0)\) to \((0, 1, 1, 0)\). We proceed similarly for the remaining two orbits.

The final assignment of group elements to states is shown in Table 5.6. The symbols \(\blacklozenge\), \(\lozenge\), \(\blacktriangleleft\), and \(\blacktriangleleft\) indicate the four disjoint orbits of states under rotation.

| \((\rho_0, 0)\) | \((0, 0, 0, 0)\) | \((\delta_1, 1)\) | \((0, 0, 1, 0)\) |
| \((\mu_1, 0)\) | \((0, 0, 0, 1)\) | \((\rho_1, 1)\) | \((0, 0, 1, 1)\) |
| \((\rho_0, 1)\) | \((0, 1, 0, 0)\) | \((\delta_1, 0)\) | \((0, 1, 1, 0)\) |
| \((\mu_1, 1)\) | \((0, 1, 0, 1)\) | \((\rho_1, 0)\) | \((0, 1, 1, 1)\) |
| \((\rho_2, 0)\) | \((1, 0, 0, 0)\) | \((\delta_2, 1)\) | \((1, 0, 1, 0)\) |
| \((\mu_2, 0)\) | \((1, 0, 0, 1)\) | \((\rho_3, 1)\) | \((1, 0, 1, 1)\) |
| \((\rho_2, 1)\) | \((1, 1, 0, 0)\) | \((\delta_2, 0)\) | \((1, 1, 1, 0)\) |
| \((\mu_2, 1)\) | \((1, 1, 0, 1)\) | \((\rho_3, 0)\) | \((1, 1, 1, 1)\) |

Table 5.6: The assignment of \(D_4 \times Z_2\) to trellis states.

It can now be verified that the branch group \(B\) determined by this state group assignment meets the conditions of Theorem 5.2. First, we find that the set \(B_0\) of 0-labeled branches is

\[
\{((\rho_0, 0), (\rho_0, 0)), ((\rho_0, 1), (\mu_1, 0)), ((\rho_2, 0), (\mu_2, 0)), ((\rho_2, 1), (\mu_2, 0)),
((\delta_1, 1), (\mu_2, 1)), ((\delta_1, 0), (\rho_2, 1)), ((\delta_2, 1), (\rho_0, 1)),
((\delta_2, 0), (\mu_1, 1))\}.
\]

As desired, \(B_0\) is a subgroup of \(B\) that is isomorphic to \((Z_2)^3\).

We next check that two branches have the same partition label if and only if they are in the same left coset of \(B_0\) in \(B\). This is indeed the case, as can be verified by hand or with the help of a computer. The consistent permutation labeling of a branch \(b\) is determined by the left translation action of \(b\) on the left cosets of \(B_0\) in \(B\); permutation labels for a set of six branches that generate \(B\) are displayed in Table 5.7.

The information in Table 5.7 completely determines the trellis diagram of the code. The group code may also be described as the set of products of shifts of three generator sequences, two of length 2 and one of length 3:

\[
g_1 = ((0 \ 2)(1 \ 3)(4 \ 6)(5 \ 7)(8 \ 10)(9 \ 11)(12 \ 14)(13 \ 15),
(0 \ 8)(1 \ 9)(2 \ 10)(3 \ 11)(4 \ 12)(5 \ 13)(6 \ 14)(7 \ 15)),
\]

\[
g_2 = ((0 \ 4)(1 \ 5)(2 \ 6)(3 \ 7)(8 \ 12)(9 \ 13)(10 \ 14)(11 \ 15),
\]

\[
(5.20)
\]
Table 5.7: The consistent permutation labeling of generator branches.

<table>
<thead>
<tr>
<th>branch</th>
<th>permutation label</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\rho_0, 0), (\mu_2, 0))</td>
<td>(0 12)(1 9)(2 14)(3 11)(4 8)(5 13)(6 10)(7 15)</td>
</tr>
<tr>
<td>((\rho_0, 0), (\rho_0, 1))</td>
<td>(0 2)(1 3)(4 6)(5 7)(8 10)(9 11)(12 14)(13 15)</td>
</tr>
<tr>
<td>((\rho_0, 0), (\mu_1, 0))</td>
<td>(0 8)(1 13)(2 10)(3 15)(4 12)(5 9)(6 14)(7 11)</td>
</tr>
<tr>
<td>((\rho_0, 1), (\rho_0, 0))</td>
<td>(0 8)(1 9)(2 10)(3 11)(4 12)(5 13)(6 14)(7 15)</td>
</tr>
<tr>
<td>((\rho_1, 0), (\rho_1, 0))</td>
<td>(0 7 8 15)(1 14 9 6)(2 5 10 13)(3 12 11 4)</td>
</tr>
<tr>
<td>((\mu_2, 0), (\delta_1, 1))</td>
<td>(0 3)(1 2)(4 7)(5 6)(8 11)(9 10)(12 15)(13 14)</td>
</tr>
</tbody>
</table>

\[ g_3 = \begin{pmatrix}
0 12 & 1 13 & 2 14 & 3 15 & 4 8 & 5 9 & 6 10 & 7 11 \\
0 8 & 1 13 & 2 10 & 3 15 & 4 12 & 5 9 & 6 14 & 7 11 \\
0 15 & 1 14 & 2 13 & 3 12 & 4 11 & 5 10 & 6 9 & 7 8 \\
0 14 & 1 3 & 2 12 & 4 10 & 5 7 & 6 8 & 9 11 & 13 15 
\end{pmatrix} \]  \hspace{1cm} (5.21)

The generator sequences \( g_1, g_2, \) and \( g_3 \) correspond to the Euclidean code sequences \((\ldots, 0, 2, 8, 0, \ldots), (\ldots, 0, 4, 12, 0, \ldots), \) and \((\ldots, 0, 8, 15, 14, 0, \ldots),\) respectively.

Though 8 branches are labeled with the partition cell 0, only the two branches \((\rho_0, 0)(\rho_0, 0)\) and \((\rho_2, 1), (\rho_2, 0)\) have an identity permutation label. It can be verified that the Euclidean code is 4-observable, while the group code is 2-observable.

The permutation \((0 7 8 15)(1 14 9 6)(2 5 10 13)(3 12 11 4)\) assigned to the rotator branch \((\tau, r) = (\rho_1, 0), (\rho_1, 0)\) has the same action on partition cells as the isometry

\[ x \mapsto \begin{bmatrix}
0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 \\
\end{bmatrix} x + \begin{bmatrix}
1 \\
0 \\
1 \\
0 \\
\end{bmatrix}, \]  \hspace{1cm} (5.23)

where omitted matrix entries are 0. As expected, this isometry is rotation by 90 degrees counterclockwise about the origin in the first two, second two, and last two coordinates. It can be shown that the existence of a horizontal branch with a rotation isometry label is a sufficient condition for rotational invariance, even when the other permutation labels are not isometric.

The permutation \((0 12)(1 9)(2 14)(3 11)(4 8)(5 13)(6 10)(7 15)\) is not isometric. The permutation sends the partition cell 0 to 12 and the partition cell 3 to 11. The minimal Euclidean distance between points in cell 0 and
points in cell 3 is 1, but the minimum distance between cell 11 and 12 is 3. The permutation does not preserve distances and therefore not an isometry.

Thus, we have a "nonlinear" trellis code that can be described as a group code with a permutation alphabet, but the permutations are not isometric. Our failure to find a isometry code representation does not, of course, prove that the code is not geometrically uniform. It may be possible to prove nonuniformity by showing that the nearest-neighbor profiles of paths starting at different states are different. □
Chapter 6

Conclusions and open problems

This dissertation only begins to explore the rich theory of group systems and isometry codes. We close with a selection of open problems.

Dual results in the theory of group systems are currently under investigation [24]. They include the construction of minimal syndrome formers, inverters, and encoders in observer canonical form. If the theory of group systems is to replace linear systems over fields, at least for trellis coding applications, standard tools must be reformulated in an accessible manner.

A convincing input-state-output theory for group systems will require a notion of sequence spaces with a noncomponentwise but local group structure, as in Kitchens [34]. Is there a notion of a causal sequence space?

Kitchens and Schmidt [34, 35] resolve a number of mathematical questions regarding the structure of compact shift-invariant group codes. Can we apply these results to isometry codes? The theory of symbolic dynamics and the theory of group systems needs to be better interconnected. The dynamics in symbolic dynamics arise from the shift operator; in our framework, dynamics are derived from the flow of time.

Benedetto, Garello, Mondin and Montorsi [3] have begun to construct new codes using the theory of group systems. Many classes of group code have yet to be explored.

It must be possible to relate the distance profile of a trellis code to its algebraic structure. This will lie at the heart of any constructive family of codes. Some simple results on the Hamming distance properties of group

121
codes are presented in [20].

Algebraic structure holds out the promise of constructive families of good codes. Presently, the only way to find high-complexity codes is with a computer search. What are the Reed-Solomon codes of Euclidean space? Can we use group representation theory to find them? This was attempted with little success with group codes for the Gaussian channel.

How many "nonlinear" Hamming-space codes are isometry codes? What about the Nordstrom Robinson code?

What are weaker forms of symmetry than geometric uniformity? Can they be studied using group systems?

An isometry code merely exhibits a transitive (often sharply transitive) subgroup of the symmetry group of a trellis code. How can we characterize the full symmetry group? What is the symmetry group of one of its Voronoi regions? More information about the structure of such polytopes may reveal whether they can asymptotically approximate spheres. Does there exist a family of trellis codes, used as quantizers, that achieves the rate-distortion function for a memoryless Gaussian source?

Do good trellis codes yield dense high-dimensional sphere packings that are as yet unknown? Can we use the symmetries of a trellis code to help calculate its normalized second moment? When are two trellis codes congruent?

Do isometry codes reach capacity? Is there any way to quantify the tradeoff between complexity and performance?

Sufficient conditions for rotational invariance are particularly easy to formulate for isometry codes [57]. The representational power afforded by non-abelian state groups provides a rich family of potentially useful rotationally invariant codes. Is there a theory of rotationally invariant trellis shaping regions?

A trellis code loses its symmetry with a poor choice of basis. Projected onto some (most, in fact) 2-dimensional hyperplanes in $\mathbb{R}^\infty$, a trellis code looks dense. Are there useful, coordinate-free performance measures for infinite-dimensional Euclidean codes? More importantly, can a trellis code be simplified by choosing the right basis?
Appendix A

Some elementary group theory

This appendix gives a brief summary of the group theory needed in the body of the dissertation. Its purpose is not so much to teach group theory or to make the work self-contained, but rather to show how little of elementary group theory is required. Fraleigh [25] is suggested as a good introductory text; Rotman [49] serves as a more advanced reference. Our only significant departure from standard group theory is the definition of infinite products of groups of sequences.

A group $G$ is a set that is closed under an associative binary operation $*$ and that has an identity element $e \in G$ and an inverse $g^{-1}$ for every $g \in G$. The order $|G|$ of a group $G$ is the number of its elements. A group $G$ is abelian if the operation $*$ is commutative; i.e., if $g*h = h*g$ for all $g, h \in G$. Hereafter we write the product $g*h$ as $gh$ and the identity $e$ as $0$.

A subgroup $H$ of $G$ is a subset of $G$ that is a group under the binary operation of $G$. A subset $H$ of $G$ is a subgroup if and only if $h'h^{-1} \in H$ for all $h, h' \in H$. A left coset of a subgroup $H$ in $G$ is a subset $gH$, where $g \in G$. Two left cosets of $H$ are either disjoint or equal, hence the set $G//H = \{gH : g \in G\}$ of left cosets of $H$ in $G$ is a partition of $G$. Any coset of a subgroup $H$ contains exactly $|H|$ elements, hence $|H|$ divides $|G|$ when $G$ is finite. The index $[G:H]$ of $H$ in $G$ is the number $|G//H| = |G|/|H|$ of left cosets of $H$ in $G$.

A right coset of $H$ is a subset $Hg$. In general, the sets of left and right cosets of $H$ in $G$ are not equal. A subgroup $H$ is normal (self-conjugate) in $G$ if the left coset $gH$ is equal to the right coset $Hg$ for all $g \in G$. A subgroup $H$ is normal in $G$ if and only if $g^{-1}hg \in H$ for all $g \in G$ and $h \in H$. Every subgroup of an abelien group is normal.
The intersection $H \cap J$ of two subgroups $H$ and $J$ of $G$ is itself a subgroup of $G$, the greatest common subgroup of $H$ and $J$. If $H$ is normal in $G$, then the *product* $HJ = \{ hJ : h \in H, j \in J \} = JH$ is a group, the least common supergroup of $H$ and $J$. The (external) *direct product* of two groups $H$ and $J$ is $H \times J = \{ (h, j) : h \in H, j \in J \}$, namely the Cartesian product of $H$ and $J$ with the group operation defined componentwise.

If $G$ and $Q$ are groups, then a map $f : G \to Q$ is a *homomorphism* if it takes the binary operation of $G$ to the binary operation of $Q$; that is, if for all $g, h \in G$,

$$f(gh) = f(g)f(h).$$

Homomorphisms are the closest analog in group theory to linear maps. The *kernel* of $f$ is the set of elements of $G$ that map to the identity of $Q$. A homomorphism sends the identity of $G$ to the identity of $Q$, inverses in $G$ to inverses in $Q$, and subgroups of $G$ to subgroups of $Q$. In particular, the whole group $G$ is sent to a subgroup of $Q$ called the *image* $f(G)$ of $G$ under $f$.

An *isomorphism* is a homomorphism that is both one-to-one and onto. A homomorphism $f : G \to G'$ is an isomorphism if and only if its kernel is the identity of $G$ and its image is $G'$. Then $G$ is isomorphic to $G'$, written $G \cong G'$. An isomorphism $f : G \to G'$ has an inverse isomorphism $f^{-1} : G' \to G$.

Given a homomorphism $f : G \to Q$, the *inverse image* of a subset $S$ of $Q$ is the set

$$f^{-1}(S) = \{ g \in G : f(g) \in S \}. \quad (A.2)$$

The kernel of a homomorphism $f$ is the inverse image $K = f^{-1}(0)$ of the identity of $Q$, and the inverse image $f^{-1}(q)$ of any $q \in f(G)$ is the coset $qK$.

The cosets of a normal subgroup $H$ in $G$ form a group under the group operation of $G$, modulo $H$, called the *quotient group* $G/H$. The group operation on cosets is $(gH)(g'H) = gg'H$ for $g, g' \in G$. The *natural map* $\phi : G \to G/H$ defined by $\phi(g) = gH$ is then a homomorphism with kernel $H$ and image $G/H$. Any normal subgroup $H$ of $G$ is thus the kernel of a homomorphism, namely the natural map. In general, $H \times (G/H)$ is not isomorphic to $G$. Moreover, $G/H$ need not be isomorphic to any subgroup of $G$.

A normal subgroup $H$ of a group $G$ is loosely analogous to a subspace $W$ of a vector space $V$. The quotient group $G/H$ then corresponds to the orthogonal space $W^\perp$ of $W$ in $V$. One must be careful not to stretch this analogy too far: for vector spaces, $V$ is equal to the direct sum of $U$ and $U^\perp$, but for groups, $G$ is not necessarily isomorphic to the direct product of $H$ and $G/H$. 
Given two groups $H$ and $J$, a group extension of $H$ by $J$ is a group $G$ such that $H$ is normal in $G$ and $G/H \cong J$. If $J$ is isomorphic to a subgroup $K$ of $G$ such that $HK = G$, then $G$ is a semidirect product of $H$ by $J$. The systematic classification of group extensions up to isomorphism is an open problem in group theory, though for small finite groups the classifications are known. Rotman [49, 137–152] considers group extensions in the special case where $H$ is abelian and semidirect products in general.

A product $HJ$ of two subgroups is an internal direct product if $H \cap J = \{0\}$ and if every element of $H$ commutes with every element of $J$. Then $HJ$ is isomorphic to the external direct product $H \times J$. If both $H$ and $J$ are normal in $G$ and $H \cap J = \{0\}$, then $HJ$ is an internal direct product.

Given an index set $I$ and a set of groups $\{G_k, k \in I\}$, the external direct product $W = \prod_{k \in I} G_k$ is the set of sequences $g = \{g_k \in G_k, k \in I\}$ with a componentwise group operation. The index set $I$ may be infinite, though in this work it is always countable. The set $W_k$ of sequences of $W$ that are zero except in the $k^{th}$ component is a normal subgroup of $W$ that is isomorphic to $G_k$, and $W$ is the internal direct product $W = \prod_{k \in I} W_k$.

More generally, if $\{H_j, j \in J\}$ is a collection of normal subgroups of $W$ indexed by a countable, ordered set $J$, then the product $\prod_{j \in J} H_j$ is defined as the set of all products $\prod_{j \in J} h_j$ with $h_j \in H_j$ for $j \in J$. If $J$ is infinite, then the product $\prod_{j \in J} H_j$ is taken to be meaningful only if for each product $\prod_{j \in J} h_j = \cdots h_{-1} h_0 h_1 \cdots$ and each $k \in I$, only finitely many of the sequences $h_j$ have a nonzero $k^{th}$ component $(h_j)_k$. All infinite products in this dissertation satisfy this condition.

The product $\prod_{j \in J} H_j$ is itself a normal subgroup of $W$. It is an internal direct product if and only if $H_j \cap (\prod_{j \neq j'} H_{j'}) = \{0\}$ for all $j \in J$, and then it is isomorphic to the external direct product of the groups $\{H_j, j \in J\}$.

A system of coset representatives $[G/H]$ for the cosets of a normal subgroup $H$ in $G$ is a subset of $G$ that includes exactly one element from each coset. We assume that the coset representative for the identity coset $H$ is 0. Every coset may then be uniquely written as $rH$ with $r \in [G/H]$, and any element $g \in G$ may be uniquely written as $g = rh$ with $h \in H$ and $r \in [G/H]$. The latter defines a one-to-one correspondence $m: G \to H \times [G/H]$ called a coset decomposition. Coset decompositions also exist when $H$ is not normal in $G$.

A normal series is a series of groups $G_0 \subseteq G_1 \subseteq \cdots \subseteq G_n$ such that each group is a normal subgroup of the next. A normal series defines a corresponding series of quotient groups $G_j/G_{j-1}$ for $j = 1, \ldots, n$. By induction, if
\[ [G_j/G_{j-1}] \text{ for } j = 1, \ldots, n \text{ is a series of systems of coset representatives, then} \]
\[ \text{every } g \in G_n \text{ may be uniquely written as } g = r_n r_{n-1} \cdots r_0 \text{ with } r_0 \in G_0 \text{ and} \]
\[ r_j \in [G_j/G_{j-1}] \text{ for } j = 1, \ldots, n. \text{ This is called a chain coset decomposition,} \]
\[ \text{and it establishes a one-to-one correspondence} \]
\[ G_n \leftrightarrow [G_n/G_{n-1}] \times \cdots \times [G_1/G_0] \times G_0, \]  
(A.3)

or equivalently a one-to-one correspondence
\[ G_n \leftrightarrow (G_n/G_{n-1}) \times \cdots \times (G_1/G_0) \times G_0. \]  
(A.4)

**Example 1.1:** Let \( \mathbb{Z} \) be the group of integers under ordinary addition. Then the set \( 8\mathbb{Z} \) of integer multiples of 8 is a subgroup of \( \mathbb{Z} \), and the quotient group \( \mathbb{Z}/8\mathbb{Z} \) is isomorphic to the group \( (\mathbb{Z}_8, +_8) \) of integers \( \{0, 1, \ldots, 7\} \) under modulo 8 addition.

The natural map \( \phi: \mathbb{Z} \rightarrow 8\mathbb{Z} \) is given by \( \phi(n) = 8\mathbb{Z} + n \). An obvious isomorphism \( \psi \) from \( \mathbb{Z}/8\mathbb{Z} \) to \( \mathbb{Z}_8 \) is \( \psi(8\mathbb{Z} + n) = n \mod 8 \); another is \( \psi(8\mathbb{Z} + n) = -n \mod 8 \).

The set \( 2\mathbb{Z}_8 = \{0, 2, 4, 6\} \) is a subgroup of \( \mathbb{Z}_8 \) that is isomorphic to \( \mathbb{Z}_4 \). The two cosets of \( 2\mathbb{Z}_8 \) in \( \mathbb{Z}_8 \) are \( \{0, 2, 4, 6\} \) and \( \{1, 3, 5, 7\} \). The group \( \mathbb{Z}_8/2\mathbb{Z}_8 \) is isomorphic to \( \mathbb{Z}_2 \). If we choose \( \{0, 1\} \) as a system of coset representatives, then \( \mathbb{Z}_8 \) has the partition
\[ \mathbb{Z}_8 = (2\mathbb{Z}_8) \cup (2\mathbb{Z}_8 +_8 1) \]  
(A.5)

and the elements of \( \mathbb{Z}_8 \) have unique coset decompositions
\[
\begin{align*}
0 &= 0 +_8 0, & 4 &= 4 +_8 0, \\
1 &= 0 +_8 1, & 5 &= 4 +_8 1, \\
2 &= 2 +_8 0, & 6 &= 6 +_8 0, \\
3 &= 2 +_8 1, & 7 &= 6 +_8 1. & \text{(A.6)}
\end{align*}
\]

It is impossible to choose a set of coset representatives for \( \mathbb{Z}_8/2\mathbb{Z}_8 \) that is a subgroup of \( \mathbb{Z}_8 \). \( \square \)

**Example 1.2:** The dihedral group \( D_3 \) is the smallest nonabelian group. The group \( D_3 \) is isomorphic to the set of symmetries of an equilateral triangle under function composition. Let \( \rho_0, \rho_1, \) and \( \rho_2 \) denote clockwise rotations by \( 0, 2\pi/3, \) and \( 4\pi/3 \) radians, and let \( \mu_1, \mu_2, \text{ and } \mu_3 \) denote reflections about axes that pass through the center of the triangle and the each of the vertices. The group table for \( D_3 \) is given in Table A.1.
Table A.1: Group table for $D_3$, the symmetries of a triangle.

<table>
<thead>
<tr>
<th></th>
<th>$\rho_0$</th>
<th>$\rho_1$</th>
<th>$\rho_2$</th>
<th>$\mu_1$</th>
<th>$\mu_2$</th>
<th>$\mu_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho_0$</td>
<td>$\rho_0$</td>
<td>$\rho_1$</td>
<td>$\rho_2$</td>
<td>$\mu_1$</td>
<td>$\mu_2$</td>
<td>$\mu_3$</td>
</tr>
<tr>
<td>$\rho_1$</td>
<td>$\rho_1$</td>
<td>$\rho_2$</td>
<td>$\rho_0$</td>
<td>$\mu_2$</td>
<td>$\mu_3$</td>
<td>$\mu_1$</td>
</tr>
<tr>
<td>$\rho_2$</td>
<td>$\rho_2$</td>
<td>$\rho_0$</td>
<td>$\rho_1$</td>
<td>$\mu_3$</td>
<td>$\mu_1$</td>
<td>$\mu_2$</td>
</tr>
<tr>
<td>$\mu_1$</td>
<td>$\mu_1$</td>
<td>$\mu_3$</td>
<td>$\mu_2$</td>
<td>$\rho_0$</td>
<td>$\rho_2$</td>
<td>$\rho_1$</td>
</tr>
<tr>
<td>$\mu_2$</td>
<td>$\mu_2$</td>
<td>$\mu_1$</td>
<td>$\mu_3$</td>
<td>$\rho_1$</td>
<td>$\rho_0$</td>
<td>$\rho_2$</td>
</tr>
<tr>
<td>$\mu_3$</td>
<td>$\mu_3$</td>
<td>$\mu_2$</td>
<td>$\mu_1$</td>
<td>$\rho_2$</td>
<td>$\rho_1$</td>
<td>$\rho_0$</td>
</tr>
</tbody>
</table>

The subgroup $H = \{\rho_0, \mu_1\}$ of $D_3$ is not normal. Its right cosets are

$$H\rho_0 = H\mu_1 = \{\rho_0, \mu_1\} \quad (A.7)$$
$$H\rho_1 = H\mu_3 = \{\rho_1, \mu_3\} \quad (A.8)$$
$$H\rho_2 = H\mu_2 = \{\rho_2, \mu_2\}, \quad (A.9)$$

while its left cosets are

$$\rho_0H = \mu_1H = \{\rho_0, \mu_1\} \quad (A.10)$$
$$\rho_1H = \mu_2H = \{\rho_1, \mu_2\} \quad (A.11)$$
$$\rho_2H = \mu_3H = \{\rho_2, \mu_3\}. \quad (A.12)$$

The partition of $D_3$ into left cosets

$$D_3 = (\rho_0H) \cup (\rho_1H) \cup (\rho_2H) \quad (A.13)$$

indicates one possible system of left coset representatives $\{\rho_0, \rho_1, \rho_2\}$.

Continuing the example, the set of rotations $R_3 = \{\rho_0, \rho_1, \rho_2\}$ is a normal subgroup of $D_3$, and that $\{\rho_0, \mu_1\}$ serves as a system of coset representatives.

$\Box$

The following are usually regarded as the basic isomorphism theorems:

**Theorem A.1 (First Isomorphism Theorem):** The kernel $K$ of a homomorphism $f: G \rightarrow Q$ is a normal subgroup of $G$, and $G/K \cong f(G)$. The isomorphism is given by the correspondence $gK \leftrightarrow f(g)$.

**Corollary A.2:** If $Q = f(G)$, then $G/K \cong Q$. 

If \( Q = f(G) \), then \( G \) is said to be an extension of \( K \) by \( Q \).

Any homomorphism \( f: G \rightarrow Q \) with kernel \( K \) and image \( f(G) \) may thus be written as the composition of the natural map \( \phi: G \rightarrow G/K \), the isomorphism between \( G/K \) and \( f(G) \), and the injection of \( f(G) \) into \( Q \):

\[
G \xrightarrow{\phi} G/K \rightarrow f(G) \rightarrow Q. \tag{A.14}
\]

The second isomorphism theorem applies when \( G \) has two subgroups \( H \) and \( J \), at least one of which is normal in \( G \). We need only the case in which both \( H \) and \( J \) are normal in \( G \). If \( H \) and \( J \) are both normal, then their product \( HJ \) is normal, and the second isomorphism theorem may be stated as follows:

**Theorem A.3 (Second Isomorphism Theorem, two normal subgroups):** If \( H \) and \( J \) are normal subgroups of \( H \), then \( HJ \) is a normal subgroup of \( G \), \( H \) and \( J \) are normal subgroups of \( HJ \), \( H \cap J \) is a normal subgroup of \( H \) and \( J \), and both \( HJ/H \cong J/(H \cap J) \) and \( HJ/J \cong H/(H \cap J) \).

Figure A.1 illustrates the second isomorphism theorem in this case.

![Diagram](image)

Figure A.1: The second isomorphism theorem with two normal subgroups.

**Corollary A.4:** If \( H \cap J = \{0\} \) then \( HJ/H \cong J \) and \( HJ/J \cong H \).

The third isomorphism theorem also involves normal subgroups of \( G \):
Theorem A.5 (Third Isomorphism Theorem): If $H$ and $J$ are normal subgroups of $G$ and $J \subseteq H$, then $J \subseteq H \subseteq G$ is a normal series, $H/J$ is a normal subgroup of $G/J$, and $(G/J)/(H/J) \cong G/H$.

The theorem is illustrated by the normal subgroups and corresponding quotient groups of Figure A.2.

![Diagram of normal subgroups and quotient groups](image)

Figure A.2: The third isomorphism theorem.

The correspondence theorem "could justifiably be called the fourth isomorphism theorem" [49]:

Theorem A.6 (Correspondence Theorem): If $K$ is a normal subgroup of $G$ and $\phi: G \to G/K$ is the natural map, the $\phi$ defines a one-to-one correspondence between the subgroups of $G$ that contain $K$ and the subgroups of $G/K$. If $H$ is a subgroup of $G$ that contains $K$, then $K$ is a normal subgroup of $H$, and the corresponding subgroup of $G/K$ is

$$\phi(H) = H/K.$$ 

(A.15)

If $J$ is another subgroup of $G$ that contains $K$, then $J$ is a subgroup of $H$ if and only if $\phi(J)$ is a subgroup of $\phi(H)$; $J$ is a normal subgroup of $H$ if and only if $\phi(J)$ is a normal subgroup of $\phi(H)$, and then $H/J \cong \phi(H)/\phi(J)$.

The correspondence theorem is illustrated in Figure A.3 for the case in which $K \subseteq J \subseteq H \subseteq G$ is a normal series. Of course, in the correspondence theorem, the natural map $\phi: G \to G/K$ may be replaced by any homomorphism $f: G \to Q$ with kernel $K$ and image $Q$. The correspondence theorem shows that the inverse image $f^{-1}(S)$ of a subgroup $S$ of $Q$ is a subgroup of $G$, that the inverse image of a normal subgroup of $Q$ is a normal subgroup of $G$, and that corresponding quotient groups are isomorphic.
Figure A.3: The correspondence theorem.

It follows from the correspondence theorem that if $H$ and $J$ are normal in $G$ and $K = H \cap J$, then

(a) $H$, $J$, $HJ$, and $K = H \cap J$ map to $H/K$, $J/K$, $HJ/K$, and $\{0\}$, respectively, under the natural map $\phi: G \to G/K$;

(b) $H/K$ and $J/K$ are normal in $G/K$;

(c) $(H/K)/(J/K) = H/J/K$ and $(H/K) \cap (J/K) = \{0\}$; so

(d) $HJ/K$ is the internal direct product of $H/K$ and $J/K$.

Finally, the following theorem is a corollary of the isomorphism theorems:

Corollary A.7 (Homomorphism/normal subgroup theorem): If $H$ and $J$ are normal subgroups of $G$ and $f: G \to Q$ is a homomorphism with kernel $K \subseteq J$, then $f(HJ)$ is a normal subgroup of $f(G)$, $f(J)$ is a normal subgroup of $f(HJ)$ and of $f(G)$, and

\[
\begin{align*}
f(G)/f(J) &\cong G/J; \\
f(HJ)/f(J) &\cong HJ/J \cong H/(H \cap J); \\
f(G)/f(HJ) &\cong G/HJ \cong (G/J)/(HJ/J).
\end{align*}
\]

This corollary is illustrated in Figure A.4.
Figure A.4: The homomorphism/normal subgroup theorem.
Bibliography


