The Dynamics of Group Codes: State Spaces, Trellis Diagrams, and Canonical Encoders

G. David Forney, Jr., Fellow, IEEE, and Mitchell D. Trott, Member, IEEE

Abstract—A group code \( C \) over a group \( G \) is a set of sequences of group elements that itself forms a group under a componentwise group operation. A group code has a well-defined state space \( \Sigma_k \) at each time \( k \). Each code sequence passes through a well-defined state sequence. The set of all state sequences is also a group code, the state code of \( C \). The state code defines an essentially unique minimal realization of \( C \). The trellis diagram of \( C \) is defined by the state code of \( C \) and by labels associated with each state transition. The set of all label sequences forms a group code, the label code of \( C \), which is isomorphic to the state code of \( C \). If \( C \) is complete and strongly controllable, then a minimal encoder in controller canonical (feedbackfree) form may be constructed from certain sets of shortest possible code sequences, called granules. The size of the state space \( \Sigma_k \) is equal to the size of the state space of this canonical encoder, which is given by a decomposition of the input groups of \( C \) at each time \( k \). If \( C \) is time-invariant and \( \nu \)-controllable, then \( |\Sigma_k| = \prod_{1 \leq j < \nu} |F_j/F_{j-1}| \), where \( F_0 \subseteq \cdots \subseteq F_\nu \) is a normal series, the input chain of \( C \). A group code \( C \) has a well-defined trellis section corresponding to any finite interval, regardless of whether it is complete. For a linear time-invariant convolutional code over a field \( \mathbb{G} \), these results reduce to known results; however, they depend only on elementary group properties, not on the multiplicative structure of \( \mathbb{G} \). Moreover, time-invariance is not required. These results hold for arbitrary groups, and apply to block codes, lattices, time-varying convolutional codes, trellis codes, geometrically uniform codes and discrete-time linear systems.

Index Terms—Linear codes, group codes, codes over groups, linear systems, group systems, systems over groups.

I. INTRODUCTION

A SEQUENCE SPACE \( W \) is defined by a discrete index set \( I \), sometimes called a time axis, and by a set \( \{A_k: k \in I\} \) of symbol alphabets \( A_k \). \( W \) is the set of all sequences \( \alpha = (a_k, k \in I) \) with components \( a_k \in A_k \) for \( k \in I \). In other words, \( W \) is the Cartesian product \( W = \prod_{k \in I} A_k \). If all symbols come from a common alphabet \( A \), then we write \( W = A^I \).

A code \( C \) is any subset of a sequence space \( W \). Ordinarily \( C \) is a proper subset of \( W \), and consequently there are dependencies within code sequences. A group sequence space \( W \) is a sequence space in which all symbol alphabets are groups \( G_k \). Then \( W = \prod_{k \in I} G_k \) is a direct product group under the componentwise group operation

\[
a \cdot b \triangleq (a_k \cdot b_k, k \in I),
\]

where \( a \cdot b \) denotes the product of \( a_k \) and \( b_k \) according to the group operation in \( G_k \).

A group code \( C \) is a subgroup of a sequence space \( W \). In other words, a group code is a set of sequences that forms a group under a componentwise group operation. If all code symbols are drawn from a common group \( G \), then \( W = G^I \), and \( C \) will be called a group code over \( G \) defined on \( I \).

A linear code over a field, vector space, ring, or module \( A \) is, at a more primitive level, a group code over the additive group of \( A \). Thus every conventional linear code (block, convolutional, lattice, or trellis) is a group code.

Most good Euclidean-space codes have been shown to be generated by group codes and thus to be geometrically uniform. If \( S \) is a signal set in a Euclidean space, if \( \Gamma \) is the symmetry group of \( S \), and if \( C \) is a group code over \( \Gamma \) with index set \( I \), then the orbit of an initial sequence \( \mathbf{x} = (x_k, k \in I) \) under \( C \),

\[
C(\mathbf{x}) = \{c(\mathbf{x}): c \in C\} = \{(c_k(x_k), k \in I): c \in C\},
\]

is a geometrically uniform code over \( S \). Usually the symmetry group \( \Gamma \) is nonabelian.

Willems has championed an approach to system theory in which a dynamical system is characterized by a time axis \( T \), a signal alphabet \( Y \), and a behavior set \( B \subseteq Y^T \), which is the set of all possible trajectories of the system. Thus, a discrete-time system \((T, Y, B)\) is precisely a code \( B \) over \( Y \) defined on \( T \). Conversely, any code \( C \) may be regarded as a discrete-time dynamical system of the Willems type. We shall therefore treat the terms “code” and “system” as synonyms.

A group system is a system in which the alphabet \( Y \) is a group, and the behavior set \( B \) forms a group under the componentwise group operation; i.e., a group system is a group code. Every conventional linear system is a group system.

In Willems’ approach, the behavior set \( (code) B \) is regarded as fundamental; all other system-theoretic constructs, such as states, inputs, or realizations, are to be deduced from \( B \). Willems successfully applies this approach to the important case of linear systems over real vector spaces.

This paper investigates the dynamical structure of group codes, regarded as group systems of the Willems type. We have multiple intersecting objectives.

1) We wish to analyze the dynamical (trellis) structure of ordinary linear codes as linear systems. Given an
ordering of the time axis \( I \), we show how to find a minimal state-space (trellis) representation of an arbitrary linear (or group) code \( C \). Trellis representations yield concrete measures of code complexity, and are useful for specifying decoding algorithms.

2) A convolutional code over a finite field is the set of output sequences of a linear time-invariant system (encoder). The algebraic structure of convolutional codes has traditionally been analyzed using a linear-system-theoretic approach. We are able to redevelop the principal results of this theory using only group structure, independent of 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 depends on time-invariance, so they apply to time-varying convolutional codes and block codes.

3) We wish to generalize these results to arbitrary groups, possibly nonabelian. This generalization is needed to analyze geometrically uniform codes.

4) We wish to show that Willems' results for linear systems can be obtained for a more general class of systems, namely group systems. We find that such linear-system-theoretic constructs as state spaces, state-transition (trellis) diagrams, and canonical minimal realizations can be developed using only elementary group theory, and that all of these constructs are essentially group-theoretic. We 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.

This paper is addressed to both coding theorists and system theorists. It uses the language of group theory, with which readers from either field may not be fully comfortable. We have, therefore, attempted to make the paper self-contained.

A partial bibliography of earlier related work would include:

1) Convolutional codes and linear systems: [11]-[18].
2) Trellis structure of linear codes: [9]-[12].
3) Euclidean-space codes based on group structure: [13]-[17].
4) Willems' approach to system theory: [18]-[21].
5) Systems over groups: [22]-[24].

For a recent survey of convolutional codes in the context of linear system theory, see [8]. For a beautiful summary of Willems' approach to system theory, see [20]. For the necessary group theory, see any group theory text; we have relied primarily on Rotman [25, ch. 2], and have summarized what we need (primarily the four isomorphism theorems) in the Appendix. It is remarkable that our development requires only these elementary results.

The closest prior work to ours is probably Kitchens' study of time-invariant systems over groups from a symbolic dynamics viewpoint [24]. The contemporaneous work of Loeliger and Mittelholzer [26] is closely related to ours and provides a complementary perspective.

In the next section, we give a preview of the paper via a motivating example: a simple linear time-invariant ring code over \( \mathbb{Z}_4 \). The problems that can arise even with such a simple code may be resolved by our group-theoretic approach.

Section III defines and discusses in more depth the basic elements of our study: sequence spaces, group codes, time axes, and group alphabets. It introduces projections and subcodes, our fundamental tools.

Section IV introduces our central concept: a group-theoretic definition of the state spaces of \( C \). The definition is justified by showing that the properties expected of state spaces are satisfied. Most importantly, the state space at time \( k \) of any minimal realization of a group code \( C \) must correspond to the group-theoretic state space \( \Sigma_k \) of \( C \) for every \( k \in I \). This leads to a canonical state realization of \( C \), and an associated canonical trellis diagram for \( C \).

In Section V, we study the state code of \( C \), namely the group code consisting of all possible state sequences of \( C \). We introduce the label code of \( C \), a reduced code that suffices to specify the dynamics of \( C \), and define the input groups of \( C \). These constructs yield a minimal encoder for \( C \) in input/state/output form.

In Section VI, we show that a complete, \( \nu \)-controllable code \( C \) is generated by the sequences in \( C \) of length \( \nu + 1 \) or less, and then construct a canonical minimal encoder that uses such sequences as generators. This encoder has a layered structure, finite memory, and other desirable properties. The state space sizes of \( C \) may be determined by counting the states of this encoder. It reduces to the "minimal encoder" of [3] or [6] for linear time-invariant convolutional codes over fields.

An alternative route to a similar canonical minimal encoder, for time-invariant group codes, is taken by Loeliger and Mittelholzer [26]. Their development is based on the observation that the state code of a \( \nu \)-controllable code \( C \) is \((\nu - 1)\)-controllable (see Section VI-A).

**Note on Terminology:** In earlier versions of this work, we spoke of "linear codes over groups" rather than "group codes." Several strong objections to this misuse of the term "linear" convinced us to drop it. Of the alternatives, "group code" and "group system" seemed best, despite a conflict with earlier terminology of Slepian [13], who used "group code for the Gaussian channel" to mean a spherical signal set \( S \) in real \( n \)-space \( \mathbb{R}^n \) generated as the orbit of a group of orthogonal transformations. Such a signal set \( S \) is not a group code in our sense; it is the image of a group "code" (of length one) acting on an initial signal point. We apologize for any confusion.

II. PREVIEW

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. [27] 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.

Fig. 1 shows the block diagram and trellis diagram of a four-state encoder for a rate-1/2 linear time-invariant convolutional
code $C$ over the ring $Z_4$. The time axis is $Z$, and the output alphabet is $Z_4 \times Z_4$. The encoder is linear over $Z_4$.

Since the encoder is linear, it may be characterized by a generator sequence

$$g = (\cdots, 00, 11, 13, 00, \cdots),$$

where “13” denotes the element $(1, 3)$ of the alphabet $Z_4 \times Z_4$. The encoder input is a sequence of elements of $Z_4$. The responses corresponding to the four possible inputs at time $k$ are

- $0 \rightarrow (00, 00);$
- $1 \rightarrow (11, 13);$
- $2 \rightarrow (22, 22);$
- $3 \rightarrow (33, 31).$

As a ring code, $C$ is the set of all linear combinations (over $Z_4$) of time shifts of the generator $g$. Alternatively, as a group code, $C$ is better characterized as the set of all sums of time shifts of these four responses, since “scalar multiplication” and “linear combination” are not defined for general groups.

A trellis diagram of a dynamical system is a state-transition diagram in which the state space $\Sigma_k$ is drawn separately for each time $k$. Output sequences correspond to paths through the trellis diagram.

In Fig. 1, the system has four states, so the trellis diagram shows four states at times $\cdots, k, k+1, k+2, \cdots$. In this system, there are state transitions connecting every state at time $k$ to every possible next state at time $k+1$. Transitions corresponding to all-zero outputs have been darkened. The system is time-invariant, so every “trellis section” is identical.

The problems with this simple encoder arise from the fact that there are multiple paths corresponding to infinite all-zero output sequences. As a result, the encoder is catastrophic and nonminimal.

A linear time-invariant encoder is catastrophic if there is an infinite-weight input sequence that generates a finite-weight output sequence. The encoder of Fig. 1 is catastrophic because the infinite-weight input sequence $(\cdots, 0, 2, 2, \cdots)$ generates the finite-weight code sequence $(\cdots, 00, 22, 00, \cdots)$. A catastrophic encoder cannot have a feedforward inverse. It is called catastrophic because it is possible for a finite number of channel errors to cause an infinite number of errors in the reconstructed input sequence.

The encoder of Fig. 1 is nonminimal because there exists a two-state encoder that also generates $C$, shown in Fig. 2. As we shall see, states “0” and “2” of Fig. 1 are equivalent, as are states “1” and “3”. Therefore, the four-state encoder of Fig. 1 can be collapsed to the two-state encoder of Fig. 2.

Fig. 2 illustrates the application of the basic results of this paper to the simple code generated by the encoder of Fig. 1. Our most fundamental result is a group-theoretic definition of the state spaces $\Sigma_k$ of a group code $C$.

In general, the zero state at time $k$ is the subset of $C$ consisting of:

- a) all code sequences whose components are zero (the identity) up to time $k$;
- b) all code sequences whose components are zero at time $k$ and later;
- c) all sums $a + b$ of a code sequence $a$ of type a) with a sequence $b$ of type b).

For the code generated by the encoder of Fig. 1, the zero state at time $k$ is the set of all code sequences that pass through either state “0” or state “2” at time $k$.

In general, the zero state at time $k$ is a subgroup of $C$. The state space $\Sigma_k$ is defined as the set of cosets of the zero state in $C$; in other words, $\Sigma_k$ is the set of equivalence classes of code sequences modulo the zero state. We shall see that the zero state is in fact always a normal subgroup of $C$; it then follows that the state space $\Sigma_k$ is itself a group, namely a quotient group (factor group).

For the code generated by Fig. 1, the state space $\Sigma_k$ at each time $k$ is isomorphic to $Z_2$. There is only one nonzero coset of the zero state at time $k$, namely the set of all code sequences that pass through either state “1” or state “3” at time $k$.

In general, the state spaces of a group code $C$ define an essentially unique minimal realizations of $C$. Fig. 2 is a simple example of a minimal canonical encoder of the type that we will develop in this paper.

The essence of our construction is the identification of a set of minimal-length generator sequences for the code. These generators then serve as the “impulse responses” of a minimal canonical encoder.

We start by asking: are there any code sequences of length 1? (The length of a nonzero sequence is the span from its first to last nonzero components.) In this case, there are: the length-one sequence $(\cdots, 00, 22, 00, \cdots)$ and all of its time shifts.
are code sequences. We therefore identify these sequences as length-one generators.

In general, the set of code sequences that are nonzero only at time $k$, plus the zero sequence, forms a group called the parallel transition group $C_{k,k}$. All code sequences in the parallel transition group $C_{k,k}$ are associated with transitions from the zero state at time $k$ to the zero state at time $k+1$. Furthermore, if there is any transition from a given state at time $k$ to another state at time $k+1$, then there must be a set of parallel transitions between these states corresponding to a coset of $C_{k,k}$.

In this example, the parallel transition group corresponds to the subgroup $\{00, 22\}$ of the output alphabet, and there are two transitions corresponding to a coset of this group connecting every pair of states at times $k$ and $k+1$.

We next ask: are there any length-two code sequences that are not generated by length-one generators? In this case, the length-two code sequences are the time shifts of

$$
(11, 13); \\
(22, 22); \\
(33, 31); \\
(33, 13); \\
(11, 31).
$$

The sequence $(22, 22)$ is generated by shifts of the length-one sequence $(22)$, but the remaining length-two sequences are not. We may choose any of these as our length-two generator—e.g., $(11, 13)$. All other length-two sequences can then be obtained by adding shifts of $(22)$ to shifts of $(11, 13)$.

Now all code sequences can be obtained as combinations of shifts of the generators $(22)$ and $(11, 13)$. In general, if a group code has controllability index $\nu$—i.e., if it is possible to join any past sequence to any future sequence via a $\nu$-step transition—then the code is generated by its sequences of length $\nu + 1$ or less. In this case, $\nu = 1$.

The encoder is constructed by associating inputs with these minimal-length generators. The input group $F_k$ of a group code $C$ is the set of all time-$k$ components of code sequences that are all-zero prior to time $k$. Here

$$
F_k = \{00, 11, 22, 33\}.
$$

This input group is isomorphic to $Z_4$, so we may equivalently take inputs from $Z_4$.

We shall see that in general $F_k$ is the set of outputs associated with transitions from the zero state. Furthermore, the set of outputs associated with transitions from any state is a coset of $F_k$. In this example, the set of outputs associated with transitions from the nonzero state is the coset $\{02, 13, 20, 31\}$ of $F_k$.

Every input in $F_k$ is the first nonzero component of a sum of minimal-length generators. In this example, $11$ is the first component of $(11, 13)$, $22$ is the first component of $(22)$, and $33$ is the first component of $(33, 13) = (22) + (11, 13)$. The four inputs are thus associated with four "impulse responses" as follows:

$$
00 \rightarrow (00) \\
11 \rightarrow (11, 13) \\
22 \rightarrow (22) \\
33 \rightarrow (33, 13).
$$

The appropriate components of all impulse responses due to current and past inputs are combined at the encoder output.

Now the encoder of Fig. 2 operates as follows. An input $f_k \in Z_4$ is decomposed into two bits, $f_{0,k}$ and $f_{1,k}$, such that

$$
f_k = f_{1,k} + 2f_{0,k}.
$$

The bit $f_{0,k}$ governs the selection of the length-one generator $(22)$, which requires no memory; the bit $f_{1,k}$ governs the selection of the length-two generator $(11, 13)$, which requires one memory element storing one bit. The output at time $k$ is given by

$$
c_k = f_{1,k}(11) + f_{1,k-1}(13) + f_{0,k}(22),
$$

which is indicated schematically in Fig. 2 as a multiplication of the binary 3-tuple $(f_{1,k}, f_{1,k-1}, f_{0,k})$ by a $3 \times 2$ matrix over $Z_4$.

In general, minimal canonical encoders will be constructed from a series of shift registers of lengths 1 to $\nu$. The register of length $j$ stores the components of the input that correspond to the length-$(j+1)$ generators. The output at time $k$ is the combination of the appropriate components of the generators determined by the current input and by the contents of the shift registers.

In the case of linear time-invariant convolutional codes over fields, the resulting minimal canonical encoders reduce to those of [3]. However, the construction here does not use multiplicative structure, polynomials, matrices, or the like. The same construction therefore works for linear time-varying codes, or for block codes. (The earliest construction of minimal convolutional encoders from "shortest independent generators" appears to be due to Roos [6]; see also Piret [7].)

For linear codes over rings, it may be impossible to construct a minimal encoder within the ring framework. There is no linear, feedbackfree, two-state encoder over $Z_4$ for this code (see Section V-D). For group codes, we find that canonical minimal encoders are generally nonlinear (nonhomomorphic), and that there exist group codes that have no linear (homomorphic) encoder (e.g., Example 2).

The time-$k$ trellis section $T_{k,k+1}$ of $C$ is defined as the subset of state-output–next state triples $(s_k, s_k, s_{k+1}) \in
\[ \Sigma_k \times G_k \times \Sigma_{k+1} \] that actually occur. In this example, \( T_{k, k+1} \) has 8 elements:

- \((0, 00, 0)\);
- \((0, 22, 0)\);
- \((0, 11, 1)\);
- \((0, 33, 1)\);
- \((1, 13, 0)\);
- \((1, 31, 0)\);
- \((1, 02, 1)\);
- \((1, 20, 1)\),

each corresponding to a transition in the trellis diagram of Fig. 2.

In general, \( T_{k, k+1} \) is a subgroup of the direct product group \( \Sigma_k \times G_k \times \Sigma_{k+1} \). Here \( T_{k, k+1} \) is a group isomorphic to \( Z_4 \times Z_2 \). Its projections onto its first, second and third components are the state group \( \Sigma_k \), the output group \( A_k \), and the next-state group \( \Sigma_{k+1} \), respectively, where the output group \( A_k \) is defined as the set of symbols in \( G_k \) that actually occur as time-\( k \) components of code sequences. Here, \( A_k = \{00, 11, 22, 33, 02, 13, 20, 31\} \) is a proper subgroup of \( G_k \) that is isomorphic to \( Z_4 \times Z_2 \). The parallel transition group \( C_{[k, k]} \) corresponds to elements of \( T_{k, k+1} \) of the form \((0, a_k, 0)\), or to the corresponding subgroup of \( A_k \). The input group \( F_k \) corresponds to elements of \( T_{k, k+1} \) of the form \((0, a_k, s_{k+1})\), or to the corresponding subgroup of \( A_k \). In group-theoretic terms, the trellis section \( T_{k, k+1} \) is a group extension (generalized product) of the state group \( \Sigma_k \) by the input group \( F_k \).

A trellis section \( T_{k, k+1} \) is what Willems calls an "evolution law;" it describes the local dynamical behavior of \( C \). The trellis diagram of \( C \) is the concatenation of the trellis sections \( T_{k, k+1} \) for all \( k \). Provided that \( C \) satisfies a technical condition called "completeness," the set of all sequences corresponding to paths through the trellis diagram of \( C \) is equal to \( C \); that is, \( C \) is determined by its local behavior.

Loeliger and Mittelholzer [20] consider group codes defined by group trellis sections ("group transition graphs"), and develop many results parallel to ours. Here, like Willems, we start from \( C \) itself, rather than from a local description of \( C \).

The trellis can be simplified further by observing that the parallel transition group \( C_{[k, k]} \) has no effect on the dynamical structure of \( C \). Therefore, it is natural to factor it out. This is done by replacing the cosets of \( C_{[k, k]} \) in the output group \( A_k \) by the elements of any label group \( Q_k \) that is isomorphic to the quotient group \( A_k / C_{[k, k]} \).

In this case, \( A_k / C_{[k, k]} \) is isomorphic to \( Z_2 \times Z_2 \), so we may replace cosets of \( C_{[k, k]} \) by elements of the label group \( Q_k = Z_2 \times Z_2 \), as follows:

- \((00, 22) \rightarrow 00;\)
- \((11, 33) \rightarrow 10;\)
- \((13, 31) \rightarrow 01;\)
- \((02, 20) \rightarrow 11.\)

Since the set of outputs corresponding to the set of transitions from any state \( s_k \) to any next state \( s_{k+1} \) is isomorphic to \( Z_2 \times Z_2 \), we may replace this set of transitions by a single branch from \( s_k \) to \( s_{k+1} \) labeled by the appropriate label \( q_k \in Q_k \). The code \( q(C) \) that results from this substitution is called the label code of \( C \). In this case, the label code of \( C \) is a rate-1/2 linear time-invariant binary convolutional code with output alphabet \( Z_2 \times Z_2 \), which has the encoder and trellis diagram shown in Fig. 3.

The dynamics of the label code \( q(C) \) are the same as those of \( C \). However, in the trellis diagram of \( q(C) \), pairs of states are connected by a single branch, rather than by a pair of parallel transitions. As a result, there is a one-to-one correspondence (in fact an isomorphism) between the label code \( q(C) \) and the set of all trellis paths, or equivalently the set \( \sigma(C) \) of all state sequences of \( C \) (the state code of \( C \)).

We, therefore, say that all codes that have the same label code are dynamically equivalent. Thus, the ring code of Fig. 1 is dynamically equivalent to the rate-1/2 linear time-invariant binary convolutional code of Fig. 3.

III. GROUP CODES, PROJECTIONS AND SUBCODES

In this section, we recapitulate the definition of a group code (or group system), and discuss in detail its key components: time axis, symbol alphabets, sequences. We define the output alphabets and the output sequence space of a group code \( C \). Finally, we define projections and subcodes of \( C \), which are fundamental to its algebraic structure, and which play dual roles.

A. Group Codes

Group codes (or group systems) are subgroups of group sequence spaces.
Definition 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 the integers \( \mathbb{Z} \), and the symbol alphabets \( G_k, k \in I \), are arbitrary groups. A group code (or group system) is any subgroup \( C \) of a group sequence space. If all symbol alphabets \( G_k \) are equal to a common group \( G \), then the sequence space is denoted by \( W = G^I \), and \( C \) is called a group code over \( C \) defined on \( I \).

Since the dynamical properties of a code \( C \) generally depend on the time axis \( I \) being ordered, there is little essential loss of generality in requiring that \( I \) be a subset of \( \mathbb{Z} \). Some static properties of \( C \) (e.g., the state spaces defined by two-way partitions of \( I \)) are well defined when \( I \) is unordered. The study of codes defined over partially ordered index sets, such as \( \mathbb{Z}^+ \), is a subject for further investigation.

We may regard any code \( C \) as being defined over \( Z \) by adding trivial symbol alphabets \( G_k = \{0\} \) for \( k \notin I \), so for notational simplicity we may let \( I = Z \).

On the other hand, by focusing on finite index sets \( I \) we can avoid some issues (e.g., “completeness”; see Section IV-G) that arise with finite time axes. For codes of length \( N \), we will usually let \( I = \{0, 1, \ldots, N - 1\} \).

We use conventional notation for time intervals: for example, if \( m \leq n \),

\[
[m, n) = \{ k \in I : m \leq k < n \}.
\]

The interval \([m, n)\) starts at time \( m \), ends at time \( n - 1 \), and has length \( n - m \). A length-one interval \([m, m + 1)\) may be denoted by \([m, m + 1) \) or simply \( \{m\} \).

If \( J \subset I \), then the complement of \( J \) in \( I \) is denoted by \( I - J \). The complement of \([m, n)\) is \( I - [m, n) \). The complement of \( I \) itself is the empty set \( \emptyset \).

We use special notations for the past and future with respect to time \( m \):

\[
m^- = \{ k \in I : k < m \};
\]

\[
m^+ = \{ k \in I : k \geq m \}.
\]

Of course, \( m^- \) and \( m^+ \) are archetypal complements. Note the asymmetry in the definition; we are thinking of a “cut” between times \( m - 1 \) and \( m \).

A code \( C \) is time-invariant if \( I = \mathbb{Z} \) and \( DC = C \), where \( D \) is the shift operator. Thus a code defined on a finite time axis cannot be time-invariant. Time-invariance is not assumed in this paper.

The granularity of the time axis 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 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, the time-invariant code over \((\mathbb{Z}_k)^5 \) of Fig. 1 may be regarded as a time-varying code over \( \mathbb{Z}_k \).

A most useful modification of a time axis \( I \) is obtained by regarding subsets of \( I \) as individual time intervals. For example, if \( I \) is partitioned into two complementary subsets, \( J \) and \( I - J \), then we may regard \( C \) as a length-two code defined on the time axis \( I' = \{ J, I - J \} \). When we focus on a finite time interval \([m, n)\), it is helpful to regard \( C \) as a length-three code defined on the time axis \( I' = \{ m^-, [m, n), n^+ \} \).

We wish to emphasize that no restrictions are placed on the symbol groups \( G_k \). In particular, they may be nonabelian. 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 “zero.” The trivial group is denoted by \( \{0\} \), and the zero sequence is denoted by \( 0 \).

A sequence is an element \( g = (g_k, k \in I) \) of a group sequence space \( W = \prod_{k \in I} G_k \). A sequence is finite if the number of its nonzero elements (its weight) is finite. A sequence 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 \( k > n \); its span is then the interval \([m, n]\). The length is \( n - m + 1 \). We sometimes say that it is active during \([m, n]\). Note that the zero sequence 0 is finite, but never active.

Because our motivation and examples are drawn primarily from coding theory, readers may develop the impression that our results are restricted to finite groups, or at least to discrete groups. We do not believe this to be the case. It is true that with continuous groups, such as real \( n \)-space \( \mathbb{R}^n \), topological issues arise that lie outside the domain of group theory. A striking illustration is that, as an additive group, \( \mathbb{R}^n \) is isomorphic to \( R \) [25, p. 247]. We do not deal with such issues in this paper, but we believe that they can be successfully addressed by restricting attention to only those subgroups of \( \mathbb{R}^n \) that are also subspaces. For a “universal algebra” approach, see Loeliger and Mittelholzer [26].

It is always possible to find a group \( G \) that contains isomorphic copies of all symbol groups \( G_k \); for example, \( G = W \). Thus every group code may be regarded as being 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 actually occur in \( C \). When \( C \) is a group code, this set is a group.

Definition 2: The time-\( k \) output group of a group code \( C \) is the set of all symbols that actually occur in some code sequence:

\[
A_k \triangleq \{ c_k : c \in C \}.
\]

The output sequence space of \( C \) is the (external) direct product

\[
V \triangleq \prod_{k \in I} A_k.
\]

The code is trim if \( V = W \), and free if \( C = V \).

Note that \( C \subseteq V \subseteq W \) in general. A group code is free, if and only if it is the direct product of its output groups. A group code \( C \subseteq W \) is trim if and only if its output sequence space \( V \) is \( W \). For example, the code of Section II is not trim as a code over \((\mathbb{Z}_k)^3 \), but it may be made trim if it is regarded as a code over its 8-element output group. (Willems [20] defines “free” as what we would call “free and trim”—i.e., \( C = W \)—but we prefer to keep these two concepts logically independent.)

The support of a code \( C \) defined on \( I \) is the subset of times \( k \in I \) for which its output alphabet \( A_k \) is nontrivial.
B. Projections and Subcodes

In geometry, a projection of a set $S$ is the set of values that $S$ takes in given coordinates when its values in the other coordinates are free, while a cross-section is the set of values that $S$ takes in given coordinates when its values in the remaining coordinates are fixed. Projections and cross-sections are geometric duals.

Here, we will be interested in projections $P_J(C)$ of a group code $C$ onto a subset $J$ of the time axis $I$, and in subcodes $C_J$ that may be characterized as cross-sections of $C$ in $J$ when the values of $C$ in the remaining coordinates, $I - J$, are fixed to zero.

Definition 3: The projection map $P_J: W \rightarrow W$ sends a sequence $g \in W$ to the sequence $h \in W$ defined by

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

Given a group code $C \subseteq W$ defined on $I$ and a subset $J \subseteq I$, the projection $P_J(C) = \{P_J(c) : c \in C\}$ of $C$ onto $J$ is the image of $C$ under the projection map $P_J$.

The projection map $P_J$ acts by "zeroing out" the components of a sequence outside of $J$. It is a homomorphism, since $P_J(gk) = P_J(g)P_J(k)$. Since $C$ is a group, its image $P_J(C)$ under the projection map $P_J$ is also a group. The projection $P_J(C)$ is thus a group code defined on $I$ with support contained in $J$.

We shall not distinguish sharply between a code defined on $I$ with support $J$, such as $P_J(C)$, and the equivalent code 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-one interval $[k, k]$; depending on context, we may refer to either $A_k$ or $P_{[k, k]}(C)$ as the output group at time $k$. Similarly, the output sequence space $V = \prod_{k \in J} A_k$ may equally well be defined as the internal direct product $V = \prod_{k \in J} P_{[k, k]}(C)$.

We shall occasionally apply projection maps to sequence spaces that are not necessarily groups; then $P_J(g)$ must be interpreted as a sequence defined on $J$:

$$P_J(g) = (g_k, k \in J).$$

Definition 4: The subcode $C_J$ of $C$ with support $J$ is the set

$$C_J = \{c \in C : c_k = 0 \text{ for } k \notin J\}.$$ 

Thus, the subcode $C_J$ is the kernel of the projection map $P_{I - J}$ restricted to $C$:

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

By the fundamental homomorphism theorem, it follows that $C_J$ is a normal subgroup of $C$, and that

$$C/C_J \cong P_{I - J}(C).$$

Thus $C_J$, like $P_J(C)$, is a group code defined on $I$ with support contained in $J$.

Alternatively, $C_J$ is the set of all $c \in C$ that are invariant under $P_J$:

$$C_J = \{c \in C : P_J(c) = c\}.$$ 

This is the intersection of $C$ and $P_J(C)$:

$$C_J = C \cap P_J(C).$$

If $C_J$ and $C_{J'}$ are two subcodes of $C$, then the product subcode $C_JC_{J'}$ is a normal subgroup of $C$, since products of normal subgroups are normal. Moreover, if $J$ and $J'$ are disjoint, then $C_J \cap C_{J'} = \{0\}$, so $C_JC_{J'}$ is a direct product.

IV. STATE SPACES AND CANONICAL REALIZATIONS

In this section, following Willems, we first review what is meant by a state realization of a system (code) $C$. We then show that when $C$ is a group code, there is a natural definition of the state spaces of $C$ as quotient groups such that $C$ has a canonical realization with these state spaces, and every minimal realization of $C$ is equivalent to this canonical realization. We conclude that the state spaces $\Sigma_k$ so defined are essentially unique and are justifiably identified as the state spaces of $C$. It therefore appears to us that the state spaces of linear systems are essentially group-theoretic constructs.

The canonical realization of $C$ has a natural group structure compatible with that of $C$. The trellis diagram of $C$ is defined by its canonical realization.

An encoder for $C$ is a realization of $C$ that is driven by inputs. In later sections, we shall develop minimal encoders for a group code $C$ whose underlying state structure is that of the canonical realization of $C$.

A. State Realizations

As in Willems [20], a state realization of a system $C$ defined on $I$ with symbol alphabets $G_k$ is a system $B$ defined on $I$ with Cartesian product alphabets $G_k \times X_k$, where the new alphabets $X_k$ are state spaces, such that 1) the projection of $B$ onto $W = \prod_{k \in I} G_k$ is $C$, and 2) $B$ satisfies the axiom of state (Markovian property).

The trajectories in $B$ are pairs $(c, x)$, where $c$ is a sequence in $C$ and $x$ is an associated state sequence. $B$ satisfies the axiom of state if whenever two trajectories $(c, x)$ and $(c', x')$ in $B$ pass through the same state $x_k = x'_k$ at time $k$, then the concatenation of the past $P_k(c, x)$ of one with the future $P_{k+1}(c', x')$ of the other is a valid trajectory in $B$. This is the defining property of states: given the state at time $k$, the past and future are conditionally independent.

The dynamics of a realization of $C$ depend on the time axis $I$ on which $C$ is defined. As discussed in Section III, the ordering and granularity of the code time axis may be adjusted in various ways. Any such adjustment will in general affect the state spaces of $C$, as well as other structural properties of realizations of $C$.

A state realization is sometimes called a state/output (S/O) system. We reserve the term encoder for an input/state/output (I/S/O) system, in which the trajectories are triples of input, state and output sequences. Input sequences are generally required to be free (or at least memoryless; see Sections IV-C and IV-G). An encoder, stripped of its input sequences, is a state realization of $C$. Given a state realization of an arbitrary system, there may be no satisfactory way of making it into an
B. The State Spaces of C

Illustration of the code sequences associated with the states at time \( k \).

encodes; however, we shall see that with group systems this is always possible.

A principal goal of this paper is the construction of canonical minimal state realizations and encoders for a group code \( C \). Loosely, a realization or encoder is minimal if its state spaces are as small as possible for each time \( k \).

The State Space Theorem

Often in system theory the state space of a system \( C \) at time \( k \) is defined abstractly as a set of equivalence classes of sequences in \( C \). When \( C \) is a group code, we shall see that the state space of \( C \) at time \( k \) may be defined in a natural manner as the set of cosets of a normal subgroup of \( C \), which we identify as the zero state. Thus state spaces are quotient groups.

It is natural to expect that any sequence in \( C \) that is all-zero prior to time \( k \) ought to pass through the zero state at time \( k \), including the zero sequence \( 0 \). The set of all such sequences is the future subcode \( C_{k^+} = \{ c \in C : P_k^+(c) = 0 \} \). Similarly, any code sequence that is all-zero at time \( k \) and later ought to have been in the zero state at time \( k \). The set of all such sequences is the past subcode \( C_{k^-} = \{ c \in C : P_k^-(c) = 0 \} \).

By the group property of \( C \), any combination of any sequence in \( C_{k^-} \) with any sequence in \( C_{k^+} \) is a code sequence which ought also to be in the zero state at time \( k \).

The zero state \( C \) at time \( k \) is therefore defined to be the product \( C_{k^-} C_{k^+} \). The zero state is depicted graphically at the top of Fig. 4.

Since the subcodes \( C_{k^-} \) and \( C_{k^+} \) are kernels of \( P_k^+ \) and \( P_k^- \), they are normal in \( C \), so their product \( C_{k^-} C_{k^+} \) is a normal subgroup of \( C \). The cosets of \( C_{k^-} C_{k^+} \) in \( C \) thus form a quotient group \( C / (C_{k^-} C_{k^+}) \), which we define as the state space \( \Sigma_k \) of \( C \) at time \( k \).

A typical state \( c C_{k^-} C_{k^+} \), denoted by \( \sigma_k(c) \), is depicted graphically at the bottom of Fig. 4. The set of parts of sequences in \( C_{k^-} C_{k^+} \) is

\[
P_k^- (c C_{k^-} C_{k^+}) = P_k^+(c) C_{k^-} \triangleq \sigma_k^-(c),
\]

a coset of the past subcode \( C_{k^-} \) that will be called the past-induced state.

The set of futures is the future-induced state

\[
P_k^+ (c C_{k^-} C_{k^+}) = P_k^-(c) C_{k^+} \triangleq \sigma_k^+(c),
\]

a coset of the future subcode \( C_{k^+} \). The state \( \sigma_k(c) \), a coset of the zero state \( C_{k^-} C_{k^+} \), is the set \( \sigma_k^-(c) \sigma_k^+(c) \) of combinations of parts in \( \sigma_k^-(c) \) with futures in \( \sigma_k^+(c) \).

Thus if two sequences are in the same state \( \sigma_k(c) = c C_{k^-} C_{k^+} \) at time \( k \), then the concatenation of the past of one with the future of the other is a sequence in \( C \) that is also in \( c C_{k^-} C_{k^+} \). The equivalence classes defined by the state space \( \Sigma_k = C / (C_{k^-} C_{k^+}) \) therefore satisfy the axiom of state.

The state space of a code \( C \) at time \( k \) arises from a "cut" of the time axis \( I \) between times \( k-1 \) and \( k \), which partitions \( I \) into the future \( k^+ \) and past \( k^- \) with respect to time \( k \). More generally, a state space of \( C \) may be defined by any two-way partition of \( I \) into complementary subsets \( \{ I, I-J \} \), as follows.

Definition 5: Given a group code \( C \) defined on \( I \) and a subset \( J \subseteq I \), the state space of \( C \) induced by the partition of \( I \) into \( J \) and \( I-J \) is the quotient group \( \Sigma_J = C / (C_J C_{I-J}) \). In the particular case \( J = k^- \) (or \( k^+ \)), the state space of \( C \) at time \( k \) is \( \Sigma_k = C / (C_{k^-} C_{k^+}) \).

Note that since \( C_J \) and \( C_{I-J} \) have disjoint support, their product \( C_J C_{I-J} \) is a direct product, isomorphic to \( C_J \times C_{I-J} \).

Example 2: Let \( C \) be the length-two code over \( Z_4 \) consisting of the eight sequences

\[
C = \{(0, 0), (0, 2), (1, 1), (1, 3),
(2, 0), (2, 2), (3, 1), (3, 3)\},
\]

defined on \( J = \{0, 1\} \). Then \( C_{I-J} = \{(0, 0), (2, 0)\} \), \( C_{I+} = \{(0, 0), (0, 2)\} \), and \( C_{I-J} = \{(0, 0), (0, 2), (2, 0), (2, 2)\} \). The state space \( \Sigma_I \) at time \( 1 \) consists of the zero state \( C_{I+} \) and its coset \( \{1, 1, 1, 3, 3, 1, 3, 3\} \); it is isomorphic to \( Z_2 \).

C. The State Space Theorem

Fig. 4 shows that there are natural one-to-one correspondences between the sets of states, past-induced states, and future-induced states of a group code \( C \). The following fundamental theorem shows that in fact these three state spaces are isomorphic. Again, for generality, we consider state spaces induced by an arbitrary two-way partition of \( I \).

State Space Theorem: Given a group code \( C \) defined on \( I \) and a subset \( J \subseteq I \), the subcodes \( C_J \) and \( C_{I-J} \) are normal subgroups of \( P_J(C) \) and \( P_{I-J}(C) \), respectively, and

\[
P_J(C)/C_J \simeq P_{I-J}(C)/C_{I-J} \simeq C / (C_J C_{I-J}) = \Sigma_J.
\]

For any \( c \in C \), the corresponding cosets under this isomorphism are \( \sigma_J(c) = P_J(c) C_J \), \( \sigma_{I-J}(c) = P_{I-J}(c) C_{I-J} \), and \( \sigma_{I-J}(c) \sigma_J(c) = c C_J C_{I-J} \), respectively.

Proof: Since the kernels of the maps

\[
P_J : C \rightarrow P_J(C);
P_J : C \rightarrow C_J \rightarrow C_J
\]

are both equal to \( C_{I-J} \), it follows from the correspondence theorem that \( C_J \) is a normal subgroup of \( P_J(C) \), that \( P_J(C)/C_J \simeq C / (C_J C_{I-J}) \), and that the coset \( c C_J C_{I-J} \) of \( C_J C_{I-J} \) in \( C \) corresponds to the coset \( P_J(c) C_J \) of \( C_J \) in \( P_J(C) \). Interchanging \( J \) and \( I-J \) yields the remaining isomorphism and correspondences. \( \square \)
The $J$-induced state space is the set of cosets of $C_J$ in $P_J(C)$, which forms the quotient group $P_J(C)/C_J$; the $(I-J)$-induced state space is the set of cosets of $C_{I-J}$ in $P_{I-J}(C)$, which forms the quotient group $P_{I-J}(C)/C_{I-J}$; and the two-sided state space is the set of cosets of $C_{J^{-1}J}$ in $C$, which forms the quotient group $C/(C_{J}C_{J^{-1}J})$. In view of the state space theorem, these three state spaces are isomorphic, so any of them characterizes the state space of $C$ induced by the partition of $I$ into $[I, J^{-1}J]$. For definiteness, we take this state space to be $\Sigma_J = C/(C_{J}C_{J^{-1}J})$.

Fig. 5 illustrates the relationships involved in the state space theorem.

Using the state space $\Sigma_J$ as a common index set for all three state spaces, it follows that $P_J(C)$ is the disjoint union of the cosets $\{\sigma_J(s), s \in \Sigma_J\}$ of $C_J$ (the $J$-induced states), $P_{I-J}(C)$ is the disjoint union of the cosets $\{\sigma_{I-J}(s), s \in \Sigma_J\}$ of $C_{I-J}$ (the $(I-J)$-induced states), and $C$ is the disjoint union of the cosets $\{\sigma_J(s)\sigma_{I-J}(s), s \in \Sigma_J\}$ of the zero state $C/C_{I-J}$ (the two-sided states). Fig. 4 illustrates these correspondences for the case $J = k$.

From the state space theorem, it is clear that the state of a code sequence $c$ at time $k$ is determined by either its past $P_k^-(c)$ or its future $P_k^+(c)$. Given a past projection $P_k^-(c)$, all information relevant to predicting the future is captured in the past-induced state $\sigma_k^+(c) = P_k^-(c)C_k^+$; this is the essence of the concept of "state." In statistical terms, the past-induced state $\sigma_k^+(c)$ is a sufficient statistic for the past with respect to any estimate of the future. Equally, given a future projection $P_k^+(c)$, the future-induced state $\sigma_k^-(c) = P_k^+(c)C_k^-$ captures all information relevant to retrodicting the past. These two determinations of the state $\sigma_k(c)$ are equivalent, and indeed redundant.

Given two code sequences $c, c' \in C$, the state space theorem implies that the concatenation $P_k^-(c)P_k^+(c')$ of the past of one with the future of the other is in $C$ if and only if the two sequences are in the same coset of $C_k^-$ in $C$; that is, if and only if the past-induced state $P_k^-(c)C_{k^-}$ and the future-induced state $P_k^+(c')C_k^+$ correspond under the isomorphism of the theorem.

Independence between past and future is expressed in system theory by the notion of memorylessness. From the state space theorem, the following are equivalent for a group code $C$:

1) the state space at time $k$ is trivial; i.e., $|\Sigma_k| = 1$;
2) $P_k^-(C) = C_{k^-}$;
3) $P_k^+(C) = C_{k^+}$;
4) $C = C_{k^-} - C_{k^+}$;
5) $C = P_k^-(C)P_k^+(C)$.

Under any of these equivalent conditions, we say that $C$ is memoryless at time $k$. If $C$ is memoryless at all times $k \in I$, then $C$ is memoryless. Obviously, if $C$ is free, then it is memoryless, but the converse need not hold; see Section IV-G.

Notes: Similar definitions of the state of a linear dynamical system have been introduced in [19, I, theorem 9], in the context of linear systems over vector spaces; in [11, Appendix], in the context of determining the trellis diagrams of block codes and lattices; and in [27]-[29], in the context of determining minimal encoders for convolutional codes over rings. It would be surprising if the literature of algebraic linear systems theory did not contain further similar definitions.

Willems shows [20, proposition 2.4(b)] that for linear systems over real vector spaces, state spaces are essentially uniquely defined. Although his notion of linearity covers only real vector space alphabets, his proof is essentially group-theoretic, as here.

In group theory, a subgroup of a direct product group is called a subdirect product. The first theorem about subdirect products in the classic text of Hall [30] is essentially the state space theorem.

D. The Canonical Realization

The state $\sigma_k(e)$ of a code sequence $c$ at time $k$ is determined by the natural map

$$\sigma_k: C \to C/(C_{k^-} - C_{k^+}) = \Sigma_k,$$

which is a homomorphism. There is therefore a well-defined state sequence $\sigma(e) = (\sigma_k(e), k \in I)$ associated with each $e \in C$. The state sequence map $\sigma: C \to \Pi_{k \in I} \Sigma_k$ that sends $e$ to $\sigma(e)$ is a Cartesian product of homomorphisms, and is therefore a homomorphism. The image of this map is the state code $\sigma(C)$ of $C$.

Definition 6: The canonical realization of a group code $C$ is the state realization with state spaces $\Sigma_k$ defined by the set of trajectories

$$B = \{(e, \sigma(e)): e \in C\},$$

where $\sigma(e)$ is the state sequence of $e$.

The canonical realization is in fact a realization of $C$, since the set of its output sequences is $C$, and since its state spaces $\Sigma_k$ satisfy the axiom of state, as shown in Section IV-B.

Since the state sequence map is a homomorphism, the trajectory code $B$ and the state code $\sigma(C)$ are both group codes. If $e \in C$ passes through the state $s_k \in \Sigma_k$ and $d \in C$ passes through $t_k \in \Sigma_k$ at time $k$, then $ed \in C$ passes through the state $s_kt_k \in \Sigma_k$. The dynamical structure of $C$ thus naturally reflects its group structure.

The next theorem establishes the canonical realization as the essentially unique minimal realization of a group code.

Minimal Realization Theorem: Any state realization of a group code $C$ may be collapsed to the canonical realization by deleting, merging and relabeling states.
Proof: Let $B' = \{ (e, x): e \in C \}$ be a realization of $C$ with state spaces $\{ X_k: k \in I \}$. First delete any states in $X_k$ that do not occur in any trajectory in $B'$ to obtain a trim state space $X'_k$. Since $B'$ satisfies the axiom of state, if two trajectories pass through state $x_k \in X'_k$ then the concatenation of the past of one with the future of the other is in $B'$. But the state space theorem implies that the concatenation of the past of one code sequence with the future of another is in $C$ if and only if the two sequences are in the same coset of $C_k \times C_{k+1}$. All code sequences that reach $x_k$ must then be in the same state of $\Sigma_k$. This defines a surjective map $m_k: X'_k \rightarrow \Sigma_k$ from the trim state spaces $X'_k$ onto the state spaces $\Sigma_k$ of $C$, for all $k$. Now, $\{ (e, m(x)): e \in C \}$ is the canonical realization of $C$.

In view of this theorem, a state realization of $C$ will be called minimal if there is a one-to-one correspondence $X_k \leftrightarrow \Sigma_k$ between its state spaces $X_k$ and the state spaces $\Sigma_k$ of $C$, for all $k$. Thus, all minimal realizations of $C$ are equivalent to the canonical realization of $C$, up to state space relabeling.

Willems gives a nice summary of the conditions under which an arbitrary code $C$ has an essentially uniquely defined minimal state realization in this sense. (There are other possible definitions of minimal realizations; e.g., see [31].) In paraphrase, he shows that the following conditions are equivalent:

1) all minimal state realizations are equivalent;
2) past equivalence is equivalent to future equivalence, or to two-sided equivalence;
3) the past-induced realization is equivalent to the future-induced realization;
4) there exists a state realization that is both past-induced and future-induced;
5) if two pasts have one future in common, then they have all their futures in common;
6) if two futures have one past in common, then they have all their pasts in common.

For group codes, the state space theorem proves (2), (5), and (6); the canonical realization satisfies (3) and (4); and (1) has just been proved.

E. The Trellis Diagram of $C$

The trellis diagram of a state realization is a directed graph that depicts its possible state sequences and corresponding output sequences. A trellis diagram is composed of trellis sections, which display the possible connections between states at times $k$ and $k+1$, with their corresponding outputs. Examples appear in Figs. 1–3.

The trellis diagram of the canonical realization of $C$ is clearly a canonical trellis diagram for $C$, since all minimal realizations of $C$ must have this trellis diagram, up to state space relabeling. We call it simply the trellis diagram of $C$. Conversely, any state space system whose trellis diagram is graph-isomorphic to the trellis diagram of $C$ is the trellis diagram of a minimal realization of $C$.

The trellis section $T_{k+1}$ of $C$ at time $k$ is the subset of transitions $(s_k, a_k, s_{k+1}) \in \Sigma_k \times A_k \times \Sigma_{k+1}$ that actually occur in the trellis diagram of $C$, where $\Sigma_k$ and $\Sigma_{k+1}$ are the state spaces of $C$ at times $k$ and $k+1$, and $A_k \cong P_k(C)$ is the output group of $C$ at time $k$. A transition is associated with a branch $(s_k, s_{k+1}) \in \Sigma_k \times \Sigma_{k+1}$ and an output symbol $a_k \in A_k$.

Algebraically, $T_{k+1}$ is the image of $C$ under the Cartesian product map

$$\sigma_k \times P_k \times \sigma_{k+1}: C \rightarrow \Sigma_k \times A_k \times \Sigma_{k+1},$$

where $\sigma_k$ and $\sigma_{k+1}$ are the state maps at times $k$ and $k+1$, respectively. Since a Cartesian product of homomorphisms is a homomorphism, $T_{k+1}$ is a subgroup of the direct product group $\Sigma_k \times A_k \times \Sigma_{k+1}$.

The kernel of this map is the set of all code sequences that pass through the zero states of $\Sigma_k$ and $\Sigma_{k+1}$ via a zero symbol, which is the direct product $C_k \times C_{k+1}$. By the fundamental homomorphism theorem, therefore,

$$T_{k+1} \cong C/(C_k \times C_{k+1}).$$

The projection of $T_{k+1}$ onto $\Sigma_k \times \Sigma_{k+1}$ is the branch space $\Sigma_{k+1}$ of $C$, consisting of all state pairs $(s_k, s_{k+1})$ that actually occur in the trellis diagram of $C$. The branch space is the image of the Cartesian product homomorphism

$$\sigma_k \times \sigma_{k+1}: C \rightarrow \Sigma_k \times \Sigma_{k+1},$$

thus $\Sigma_{k+1}$ is a subgroup of the direct product group $\Sigma_k \times \Sigma_{k+1}$.

The kernel of this map is the set of all code sequences that pass through the zero states of $\Sigma_k$ and $\Sigma_{k+1}$; namely, the intersection of $C_k \times C_{k+1}$ and $C_{k+1} \times C_k$, which is the direct product $C_k \times C_{k+1}$. Therefore,

$$\Sigma_{k+1} \cong C/(C_k \times C_{k+1}).$$

Alternatively, the branch space $\Sigma_{k+1}$ may be obtained by projecting the trellis section $T_{k+1}$ onto $\Sigma_k \times \Sigma_{k+1}$. The kernel of this projection is $\{ 0 \} \times C_{k+1}$, so

$$\Sigma_{k+1} \cong T_{k+1}/\{ 0 \} \times C_{k+1}.$$

The parallel transition group at time $k$ is the set of output symbols $a_k$ that are associated with transitions $(0, a_k, 0)$ between zero states. Since

$$P_k(C_k \times C_{k+1} = C_{k+1},$$

the parallel transition group at time $k$ is $C_{k+1}$. Depending on context, we may regard $C_{k+1}$ as a subgroup either of $A_k$ or of $P_k(C)$.

Given a branch $(s_k, s_{k+1})$, the set of output symbols $a_k$ associated with the set of parallel transitions $(s_k, a_k, s_{k+1})$ is then a coset of $C_{k+1}$ in $P_k(C)$. These cosets may be designated by labels in a label group $Q_k \cong P_k(C)/C_{k+1}$ (see Section V-B). If each set of parallel transitions is replaced by a single labeled branch $(s_k, q_k, s_{k+1})$, then we obtain a labeled trellis section. A labeled trellis section is a subset of $\Sigma_k \times Q_k \times \Sigma_{k+1}$ that is isomorphic to the branch space $\Sigma_{k+1}$. 
Finally, trellis sections may be linked to form a trellis diagram for $C$ by identifying states of $\Sigma_{k+1}$ in the trellis sections $T_{k, k+1}$ and $T_{k+1, k+2}$ for all $k$.

Loeliger and Mittelholzer [26] start by defining a time-invariant group ("convolutional") code $C$ by a trellis section ("transition graph") that generates it. They endow trellis sections with group structure, as above (or, more generally, with a module or vector space structure), and then derive the properties of $C$ from this algebraic structure.

$F.$ The State Space Theorem and Length-Two Codes

A helpful interpretation of the state space theorem results from regarding a group code $C$ defined on $I$ as a length-two code defined on the time axis $I' = \{J, I - J\}$.

With respect to this time axis, $C$ remains a group code; the output groups of $C$ are $P_J(C)$ and $P_{I-J}(C)$, and the parallel transition groups are $C_J$ and $C_{I-J}$.

A length-two group code has three state spaces. The initial state space $\Sigma_0$ and the final state space $\Sigma_0$ are trivial, since they arise from the partitions of $I'$ into $\{I', \emptyset\}$ and $\{\emptyset, I'\}$, respectively; thus

$$\Sigma_0 = C/(C_P C_O) = C/C = \{0\},$$

and $\Sigma_0$ is identical to $\Sigma_0$, since it is induced by the same partition. With respect to the original time axis $I$, the initial and final state spaces will be denoted by $\Sigma_{-\infty}$ and $\Sigma_{\infty}$, respectively.

The intermediate state space $\Sigma_1$ is the state space $\Sigma_J$ of $C$, since

$$\Sigma_1 = C/(C_J C_{I-J}) = \Sigma_J.$$

The trellis diagram of $C$ is shown in Fig. 6.

Every branch in the trellis corresponds to a coset of a parallel transition group, $C_J$ or $C_{I-J}$. Every path through the trellis corresponds to a coset of $C_J C_{I-J}$. Fig. 6 thus illustrates clearly the three characterizations of the state space $\Sigma_J$ and the isomorphisms of the state space theorem.

As a length-two code defined on $I' = \{J, I - J\}$, a group code $C$ is thus dynamically equivalent (see Section V-B) to a repetition code of length two over $\Sigma_J$.

$G.$ Completeness and the Local Behavior of Codes

Having defined the trellis diagram of a group code $C$, we naturally ask whether $C$ is generated by its trellis diagram; that is, whether $C$ is equal to the set of all output sequences corresponding to paths through the trellis diagram of $C$.

The answer is again given by Willems [20], as follows. If $C$ is complete, then $C$ is fully characterized by its trellis diagram; i.e., by its local behavior. Otherwise, given the trellis diagram of an incomplete code $C$, it may be necessary to specify $C$ further via global constraints. (For pointing out the necessity of completeness, we thank Loeliger and Mittelholzer [private communication].)

Definition 7 [20]: A code $C$ is complete if the fact that $P_J(g) \in P_J(C)$ for every finite interval $J \subseteq I$ implies that $g \in C$.

In other words, a code $C$ is complete if membership of an arbitrary sequence $g \in W$ in $C$ can be checked by examining the projections of $g$ onto finite intervals $J$; i.e., if there are no global constraints on $C$.

Examples of group codes that are not complete are the following.

1. The set $W_1$ of all finite sequences in a group sequence space $W = G^Z$.
2. The set $W_L$ of all one-sided sequences (formal Laurent series) in $W = G^Z$.
3. The set $W_{V^2}$ of all square-summable sequences in $W = R^Z$.

Each of these codes is memoryless, but not free. More generally, the set of all finite, one-sided, or square-summable sequences in a complete group code $C$ is typically an incomplete group code with nontrivial dynamics.

On the other hand, it can be shown that a group code is complete under any of the following conditions [32]:

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

This implies, for example, that if $C$ is complete, then the products $C_{k-} C_{k+}$ and $\prod_{k \in I} C[k, k]$ are complete for all $k \in I$.

However, as shown in [32], products of complete codes are not guaranteed to be complete in general; for example, even if $C$ is complete, $C_{k+1} C_{k+}$ or $\prod_{k \in I} C[k, k+1]$ may not be complete.

In symbolic dynamics, completeness is called closure; indeed, completeness amounts to topological closure in the product topology of the discrete (Hamming distance) topology on the subgroups. Completeness ensures that any infinite product of code sequences (as defined in the appendix) is in $C$. For further discussion of topological considerations, see [26] or [32].

Kitchens [24] shows that a complete linear time-invariant code over a finite group $G$ must have finite state spaces.

The following example, due to Loeliger [33], illustrates that an incomplete group code $C$ has a well-defined trellis diagram. The example shows that the dynamics of the completion $C'$ of $C$ (the least complete code that contains $C$) may be simpler than the dynamics of $C$.

Example 3: Let $C_{even}$ be the linear time-invariant binary group code consisting of all finite binary sequences in $\{Z_2\}^Z$ with even Hamming weight. $C_{even}$ is incomplete. The state space of $C_{even}$ at any time $k$ is isomorphic to $Z_2$, since $C_{even}$, $k^-$, the set of all finite even-weight past sequences, is a proper subgroup of $P_k(\text{-} C_{even})$, which includes also the odd-weight past sequences. Since the extension of any even-weight sequence by a "0" (resp. "1") gives an even-weight (resp., odd-
V. STATE CODES, LABEL CODES, AND INPUTS

Section IV shows that, given a choice of time axis \( I \), every group code \( C \) has an essentially unique minimal realization and trellis diagram. The dynamics of \( C \) are characterized by the state code \( \sigma(C) \) of \( C \), which is itself a group code.

In this section we examine the dynamical structure of \( C \) in further detail. The dynamics of \( C \) may alternatively be characterized by a reduced code \( \eta(C) \), called the label code of \( C \), which is also a group code. Moreover, the label code \( \eta(C) \) is isomorphic to the state code \( \sigma(C) \). Either one specifies the trellis diagram of \( C \).

We give a natural definition of the input groups \( F_k \) of a group code \( C \) at each time \( k \), such that the input sequence space is memoryless and drives the state transitions of a minimal encoder. This leads to next-state and output maps for \( C \), which specify a minimal encoder realization in input/state/output form.

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

A. The State Code and Parallel Transition Code of \( C \)

As we have seen, a code sequence \( c \) in a group code \( C \) passes through a well-defined \textit{state sequence} \( \sigma(c) = (\sigma_k(c), k \in Z) \), with states \( \sigma_k(c) \) in the state spaces \( \Sigma_k \) of \( C \), which are determined by the natural maps

\[
\sigma_k: C \rightarrow C/(C_k - C_{k+1}) = \Sigma_k.
\]

The state sequence \( \sigma(c) \) is thus an element of the state sequence space \( \prod \Sigma_k \). The state sequence map \( \sigma \) of \( C \), namely

\[
\sigma: C \rightarrow \prod \Sigma_k,
\]

is a homomorphism that sends a code sequence \( c \) to its state sequence \( \sigma(c) \). The image \( \sigma(C) \) is the \textit{state code} of \( C \). Because \( \sigma \) is a homomorphism, the state code \( \sigma(C) \) is itself a group code.

The kernel of this map, the subcode consisting of all code sequences that map to the zero state sequence, is the \textit{parallel transition code} \( C_0 \) of \( C \). By the fundamental homomorphism theorem, \( \sigma(C) \) is isomorphism to \( C/C_0 \).

By the definition of a zero state, \( C_0 = \bigcap_k C_k - C_{k+1} \).

\textbf{Parallel Transition Code Theorem:} The parallel transition code of a complete group code \( C \) is the direct product \( C_0 = \prod_k [C_{[k, k)} \) of the parallel transition groups \( C_{[k, k)} \).

\textbf{Proof:} By the results of [32] cited in Section IV-E, both \( \bigcap_k C_k - C_{k+1} \) and \( \prod_k [C_{[k, k)} \) are complete. It suffices to show that for any finite interval \( [m, n) \), \( P_{[m, n)}(C_0) = P_{[m, n)}(\prod_k [C_{[k, k)} \). Obviously,

\[
P_{[m, n)}(C_{[k, k)} = \prod_{k \in [m, n)} C_{[k, k)}.
\]

As noted in Section IV-E, the intersection of \( C_k - C_{k+1} \) and \( C_{k+1} - C_{k+1+} \) is \( C_k - C_{[k, k)} C_{k+1+} \); more generally, by induction,

\[
\bigcap_{k \in [m, n)} C_k - C_{k+1} = C_{m-} \{ \prod_{k \in [m, n)} C_{[k, k)} \} C_{n+},
\]

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

Because \( \prod_k [C_{[k, k)} \) is a sequence space, the parallel transition code \( C_0 \) of a complete code \( C \) is free. Indeed, \( C_0 \) is then the maximal free subcode of \( C \). When \( C \) is not complete, \( C_0 \) is the maximal memoryless subcode of \( C \). We call \( C_0 \) the \textit{nondynamical component} of the code \( C \), and \( C_{[k, k)} \) the \textit{nondynamical component} of the output group \( A_k \approx P_{[k, k)}(C) \).

\textbf{Notes:} In an earlier version of this paper [34], the state code \( \sigma(C) \) was called the “metacode” of \( C \). Loeliger and Mittelholzer [26] refer to the state sequence map \( \sigma \) by the suggestive name of “derivative.”

B. The Label Code of \( C \)

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 groups \( C_{[k, k)} \) may be “factored out” without affecting the dynamical structure of \( C \).

Consider the partition of \( Z \) into the single index \( \{ k \} \) and its complement \( Z - \{ k \} \). The state space of \( C \) that corresponds to this partition is the \( \{ k \} \)-induced state space

\[
\Sigma_{[k, k)} = P_{[k, k)}(C)/C_{[k, k)}.
\]

(Caution: \( \Sigma_{[k, k)} \neq \Sigma_k \).) The cosets of the parallel transition group \( C_{[k, k)} \) in the output group \( P_{[k, k)}(C) \) may therefore be labeled by the states in \( \Sigma_{[k, k)} \), or by the elements of any group \( Q_k \) that is isomorphic to \( \Sigma_{[k, k)} \).

\textbf{Definition 8:} The label space at time \( k \) of a group code \( C \) is its \( \{ k \} \)-induced state space \( \Sigma_{[k, k)} \). A label group \( Q_k \) at time \( k \) is any group isomorphic to \( \Sigma_{[k, k)} \). The label map \( \alpha \) of \( C \) at time \( k \) is then the composition

\[
\alpha_k: C \rightarrow P_{[k, k)}(C) \rightarrow P_{[k, k)}(C)/C_{[k, k)} = \Sigma_{[k, k)} \rightarrow Q_k
\]

of the projection \( P_{[k, k)} \), the natural map from \( P_{[k, k)}(C) \) to \( P_{[k, k)}(C)/C_{[k, k)} = \Sigma_{[k, k)} \), and the isomorphism from \( \Sigma_{[k, k)} \) to \( Q_k \). The label sequence space is \( \prod Q_k \), and the Cartesian product of the label maps is the label sequence map.
The label code is the image \( q(C) \) of the label sequence map.

A label map \( q_k \) is a homomorphism whose image \( q_k(C) \) is the label group \( Q_k \) and whose kernel is the parallel transition group \( C_{[k, k]} \) at time \( k \). The label sequence map \( q \) is a homomorphism whose image is the label code \( q(C) \) and whose kernel is the parallel transition code \( C_0 \).

Therefore, by the fundamental homomorphism theorem, the label code \( q(C) \) is a group code isomorphism to \( C/C_0 \), like the state code \( \sigma(C) \). Indeed, two code sequences have the same label sequence if and only if they have the same state sequence; the label sequence map \( q \) and the state sequence map \( \sigma \) send a given coset \( c C_0 \) of \( C_0 \) to \( q(c) \) and \( \sigma(c) \), respectively. It follows that the state code \( \sigma(C) \) also serves as a state code \( \sigma(q(C)) \) for the label code \( q(C) \) (by a mild abuse of notation), that the map from label sequences \( q(e) \) to state sequences \( \sigma(q(e)) \) is an isomorphism, and that the parallel transition code of \( q(C) \) is trivial.

**Label Code Decomposition Theorem:** A group code \( C \) has a coset decomposition into its parallel transition code \( C_0 \) and a label code \( q(C) \) whose dynamics are the same as the dynamics of \( C \): for any \( c \in C \), the state sequence \( \sigma(q(c)) \) of the label sequence \( q(c) \) is equal to \( \sigma(c) \). For the label code \( q(C) \), the state sequence map is an isomorphism, \( \sigma(q(C)) \cong q(C) \); therefore the parallel transition code of \( q(C) \) is trivial.

In group-theoretic terms, \( C \) is an extension of the label code \( q(C) \) by the parallel transition code \( C_0 \).

The trellis section of the label code \( q(C) \) at time \( k \) is the labeled trellis section introduced in Section IV-E, a subgroup of \( \Sigma_k \times Q_k \times \Sigma_{k+1} \), which is isomorphic to the branch space \( \Sigma_{k, k+1} \). The trellis diagram of \( q(C) \) is the same as that of \( C \), except that sets of parallel transitions are replaced by single labeled branches.

Since a label group \( Q_k \) is essentially the \( \{ k \} \)-induced state space \( \Sigma_{[k, k]} \) of \( C \), it follows that the label \( q_k(c) \) of a code sequence \( c \) at time \( k \) contains all of the information in the symbol \( c_k \) that is relevant to predicting all other components of \( c \). In other words, the components in \( Z - \{ k \} \) are independent of the nondynamical component \( C_{[k, k]} \) of the output group. For example, if \( C \) is free, then all label groups are trivial, signifying that all output groups \( A_k = C_{[k, k]} \) are independent.

Different codes can have the same label code. The dynamics of the two codes are identical, but the codes are not. Two codes whose label codes or state codes are identical, up to componentwise isomorphism, are *dynamically equivalent*.

For example, all free codes defined on the same time axis are dynamically equivalent. In Section II, we saw that the ring code over \( (Z_4)^2 \) of Fig. 1 is dynamically equivalent to the binary code of Fig. 3. From Section IV-F, we see that if a group code \( C \) is regarded as a length-two code defined on \( \{ J, I - J \} \), then \( C \) is dynamically equivalent to a length-two repetition code over the state group \( \Sigma_J \).

The label code concept is well established in Euclidean-space coding theory, where a signal space code is usually specified by a group label code \( q(C) \) and a map from label groups \( Q_k \) to subsets of a partitioned signal set \( S \). If \( S \) corresponds to a group \( G \) and the subsets correspond to cosets of a normal subgroup \( K \) of \( G \) such that \( Q_k \cong G/K \) for all \( k \), then the signal space code is called a "coset code" [15].

Coset codes are examples of a general method of constructing a group code \( C \) by extension of a group label code \( C' \subset \prod Q_k \) by a direct product code \( C_0 = \prod C_{[k, k]} \). Each label group \( Q_k \) is first extended by a parallel transition group \( C_{[k, k]} \) to form an output group \( A_k \) such that \( A_k/C_{[k, k]} \cong Q_k \). Then each component of each label code sequence in \( C' \) is replaced by the corresponding coset of \( C_{[k, k]} \) in \( A_k \) to form \( C \). The trellis diagram of \( C \) is the same as that of \( C' \), except that its branches are "fleshed out" by replacing labels by cosets.

**Example 4:** The binary \( (8, 4, 4) \) extended Hamming (or first-order Reed-Muller) code may be viewed as the code over \( (Z_2)^2 \) whose 4-state trellis diagram is shown in Fig. 8 [11]. The Gosset \((E_8)\) lattice may be constructed as a "mod-2" ("Construction A" [34]) lattice by replacing each label in \( (Z_2)^2 \) by a corresponding coset of \( 2Z^2 \) in \( Z^2 \); since \( Z^2/2Z^2 \cong (Z_2)^2 \). Thus this version of the \( E_8 \) lattice has the same trellis structure as the binary \( (8, 4, 4) \) code. (In lattice theory, this type of construction is called "gluing," with the label code being regarded as the "glue" [35].)

**C. Inputs**

Willems' definition of a dynamical system is based only on the set \( C \) of its possible output sequences. We have seen that in group systems the state spaces of \( C \) are essentially uniquely defined at each time \( k \). In this section we show that in group systems there is also a group-theoretic definition of inputs.

If a code sequence \( c \in C \) passes through the zero state at time \( k \), i.e., if \( \sigma_k(c) = 0 \), then the set of its possible future continuations is the future subcode \( C_{k+} \). The set of possible outputs at time \( k \) is therefore the projection \( P_{[k, k]}(C_{k+}) \). We define the input group \( F_k \) of a group code \( C \) at time \( k \) as this "first-output group;"

\[
F_k \triangleq P_{[k, k]}(C_{k+})
\]

The input sequence space of \( C \) is then \( \prod F_k \).

Again, depending on context, we may regard an input group \( F_k \) as a subgroup either of \( P_{[k, k]}(C) \) or of the symbol output group \( A_k \).

By the state space theorem, the set of code sequences that agree with an arbitrary code sequence \( c \in C \) on the past \( k \) is \( c C_{k+} \). The set of possible outputs at time \( k \) leaving a state \( \sigma_k(c) \) is therefore the projection

\[
P_{[k, k]}(c C_{k+}) = P_{[k, k]}(c) F_k,
\]

which is a coset of \( F_k \) in \( P_{[k, k]}(C) \cong A_k \).
Thus, there is a one-to-one correspondence between the input group $F_k$ and the set of possible next outputs from $P_k(c)$. This defines an output map from the Cartesian product $\Sigma_k \times F_k$ onto the output group $A_k$. (Caution: the output map is not necessarily a homomorphism from the direct product group $\Sigma_k \times F_k$ onto $A_k$; see Section V-D.)

The next state $P_{k+1}(c)$ is determined by the state $P_k(c)$ and the output $q_k(c)$. Equivalently, since the previous paragraph shows that, given $P_k(c)$, there is a one-to-one correspondence between $F_k$ and the set $P_{k+1}(c)F_k$ of possible next outputs, the next state may be determined by $P_k(c)$ and an element of the input group $F_k$. This yields a next-state map from $\Sigma_k \times F_k$ onto $\Sigma_{k+1}$. (Same caution.)

Furthermore, since all outputs $q_k(c)$ with the same label $q_k(c_f)$ lead to the same next state, the next-state map may be reduced to a map from $\Sigma_k \times q_k(F_k)$ onto $\Sigma_{k+1}$, where $q_k(F_k)$ is the first-label group.

The last-output group at time $k$ is defined analogously to the input group $F_k$ as $L_k \triangleq P_{k+1}(C_{k+1})$. The last-output group becomes the input group if the ordering of the time axis is reversed. Similarly, the last-label group at time $k$ is $q_k(L_k)$.

Since, given $P_k(c)$, there is a one-to-one correspondence between $F_k$ and the set $P_{k+1}(c)F_k$ of possible next outputs, every code sequence $c$ uniquely determines an input sequence in the input sequence space $\prod F_k$. In other words, the minimal encoder for $C$ defined by the output and next-state maps is invertible.

The input space is defined as the image of this inverse map, a subset of the input sequence space $\prod F_k$. If there are no global constraints, then the input space is equal to $\prod F_k$; i.e., the input space is free. Otherwise, the input space is a proper subset of $\prod F_k$, and is only memoryless. This suggests that a group code $C$ might be characterized by its trellis diagram and by a memoryless input space that reflects the global constraints. In Example 3, for instance, $C_{even}$ may be characterized as the set of all output sequences that correspond to finite input sequences; i.e., the input space is the set of all finite sequences in $(\mathbb{Z}_2)^2$.

An invertible encoder for $C$ is catastrophic if there exist two code sequences that differ in finitely many components whose corresponding input sequences differ in infinitely many components.

**Theorem:** A minimal encoder for a group code $C$ is non-catastrophic.

**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 $[m, n]$. \hfill $\square$

**D. A Minimal Encoder Construction**

We shall now see that the domain of the output and next-state maps should be viewed not as the direct product $\Sigma_k \times F_k$, but rather as the trellis section $T_{k,k+1}$, which is an extension of $\Sigma_k$ by $F_k$. This viewpoint leads to a minimal encoder that is “almost linear.” Loeliger and Mittelholzer first arrived at this idea; see [26].

**Trellis Section Theorem:** A trellis section $T_{k,k+1} \subset \Sigma_k \times A_k \times \Sigma_{k+1}$ of a group code $C$ has a normal subgroup $\bar{F}_k$ isomorphic to the input group $F_k$ such that $T_{k,k+1}/\bar{F}_k \cong \Sigma_k$.

i.e., $T_{k,k+1}$ is an extension of $\Sigma_k$ by $F_k$.

**Proof:** Consider the projection of $T_{k,k+1}$ onto its first component. The image of this projection is $\Sigma_k$, and the kernel is

$$\bar{F}_k \triangleq \{(s_k, c_k, s_{k+1}) \in T_{k,k+1} : s_k = 0\},$$

the set of transitions that start from the zero state. Thus $T_{k,k+1}/\bar{F}_k \cong \Sigma_k$. The projection of $\bar{F}_k$ onto its second component is the input group $F_k$; the kernel of this projection is the set of elements of $T_{k,k+1}$ of the form $(0, 0, s_{k+1})$, but this is the trivial group, because starting from the zero state in $\Sigma_k$, if the next output symbol is 0, then the next state must be 0. Thus, $\bar{F}_k \cong F_k$.

The fact that $T_{k,k+1}$ is an extension of $\Sigma_k$ by $F_k$ implies that there is a one-to-one correspondence $\Sigma_k \times F_k \leftrightarrow T_{k,k+1}$. The output map and next-state maps of $C$ may be defined by the composition of this correspondence with projections onto the second and third components of $T_{k,k+1}$.

The input/state/output (I/S/O) system shown in Fig. 9 is thus a minimal encoder for a general group code $C$. At time $k$, the trellis section $T_{k,k+1}$ is obtained as an extension of the state in $\Sigma_k$ by an input in $F_k$; the output in $A_k$ and the next state in $\Sigma_{k+1}$ are then obtained as projections of $T_{k,k+1}$.

Is this encoder linear? In a group-theoretic context, an encoder is “linear” if all maps are homomorphisms. Since projections are homomorphisms, the only possibly nonlinear operation in this encoder structure is the extension of $\Sigma_k$ by $F_k$.

A natural definition of a linear encoder is one that induces a homomorphic map from the input sequence space to the output sequence space. A group extension is not in general a homomorphism, and the encoder above is not necessarily linear in this sense. (To find a linear encoder for an arbitrary group code requires a notion of an input sequence space that is not a direct product group under a componentwise group operation; such groups are considered by Kitchens [24].)
Example 1 (cont.): The two encoders shown in Fig. 10 are both minimal encoders for the example code of Section II. For this code, the subgroup $F_k$ of $T_k, k+1$ consists of the four triples $\{(0, 00, 0); (0, 11, 1); (0, 22, 0); (0, 33, 1)\}$ and is isomorphic to $Z_4$. The two encoders result from choosing $(1, 13, 0)$ or $(1, 02, 1)$ as coset representatives of the nonzero coset of $F_k$ in $T_k, k+1$. The impulse responses in the two cases are

\[
\begin{align*}
0 &\rightarrow (00) \\
1 &\rightarrow (11, 13) \\
2 &\rightarrow (22) \\
3 &\rightarrow (33, 13)
\end{align*}
\]

Thus, the first encoder has finite input (controller) memory, but its I/O map is nonlinear; the second encoder has a linear I/O map, but infinite input memory.

The second encoder illustrates problems that can arise when the input memory is infinite. The all-zero input sequence can produce either $\cdots, 00, 00, \cdots$ or $\cdots, 02, 02, \cdots$ as an output sequence, depending on the initial state at time $-\infty$. Thus the input-output sequence map is many-to-one. Furthermore, the output at time $k$ is in general the sum of an infinite number of nonzero elements of past input responses, and infinite sums are not necessarily well defined. These problems are customarily avoided by fixing the initial state to zero and allowing only one-sided input sequences (formal Laurent series); however, under such restrictions the encoder may not generate the entire code. For instance, it cannot then generate any bi-infinite periodic sequence other than $0$. None of these problems arises when the input memory is finite; therefore we prefer finite-memory encoders.

Example 2 (cont.): Consider again the following simple length-two group code $C$ over $Z_4$:

\[
C =\{(0, 0), (1, 1), (2, 2), (3, 3), (0, 2), (1, 3), (2, 0), (3, 1)\}.
\]

$C$ is isomorphic to $Z_4 \times Z_2$; its output groups $A_1$ and $A_2$ are isomorphic to $Z_4$; its parallel transition groups are $C_{[0, 0]} = \{(0, 0), (2, 0)\} \cong Z_2$ and $C_{[1, 1]} = \{(0, 0), (0, 2)\} \cong Z_2$; its nontrivial state group $\Sigma_1$ is isomorphic to $Z_2$; and its input groups $F_0$ and $F_1$ are isomorphic to $Z_4$ and $Z_2$, respectively. The trellis section $T_{1, 2} = \{(0, 0, 0), (1, 1, 0), (0, 2, 0), (1, 3, 0)\}$ is isomorphic to $Z_4$, whereas $\Sigma_1 \times F_1 \cong Z_2 \times Z_2$. There can therefore be no homomorphic output map from $\Sigma_1 \times F_1$ to $A_1$, since there is no homomorphism from $Z_2 \times Z_2$ to $Z_4$.

The system of Fig. 9 may be refined as in Fig. 11, by decomposing the input in $F_k$ into a label in the first-label group $q_k(F_k)$ and a nondynamical component in $C_{[k, k]}$. The branch space $\Sigma_{k+1}$ can then similarly be shown to be an extension of $\Sigma_k$ by $q_k(F_k)$. Projections of $\Sigma_{k+1}$ and a label in $Q_k = q_k(A_k)$ yield the next state in $\Sigma_{k+1}$ and a label in $Q_k = q_k(A_k)$. The actual output symbol is determined by the output label and the nondynamical input component; i.e., by an extension of $Q_k$ by $C_{[k, k]}$. Again, the only possibly nonlinear operations are cost decompositions and group extensions.

Fig. 12 illustrates the sequential progression of state spaces and trellis sections, or alternatively branch spaces, in the encoders of Figs. 9 and 11, or indeed in any minimal encoder. The figure shows alternating cycles of expansion, as $\Sigma_k$ is extended to $T_k, k+1$ or $\Sigma_k, k+1$ by $F_k$ or $q_k(F_k)$, and compression, as $T_k, k+1$ or $\Sigma_k, k+1$ is projected onto $\Sigma_{k+1}$.

Similarly, it can be shown that $T_k, k+1$ is an extension of $\Sigma_{k+1}$ by the last-output group $L_k$, and $\Sigma_k, k+1$ is an extension of $\Sigma_{k+1}$ by the last-label group $q_k(L_k)$. Thus the cycles of Fig. 12 can also be run in the reverse time direction, with the groups $L_k$ or $q_k(L_k)$ as the "inputs."
A. Controllability

A repetition code is an example of projection time axis. A code path of length code the future of any other code sequence. In other words, length-two time axis of a code is necessarily a group code. Our encoder is based on a chain coset decomposition of ical minimal encoder with memory.

A code sequence 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 in the construction of a finite-input-memory encoder, and finally prove that the resulting encoder is minimal.

V. CANONICAL MINIMAL ENCODERS

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

We first define $\nu$-controllability in Section VI-A, again following Willems [20]. Section VI-B gives two results concerning $\nu$-controllable group codes that may be used to construct a layered canonical minimal encoder. One such construction will be given here; an alternative construction appears in [26].

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 in the construction of a finite-input-memory encoder, and finally prove that the resulting encoder is minimal.

A. Controllability

With Willems [20], we define controllability as a property of a code $C$, not of a realization of $C$.

Definition 9: A code $C$ is $[m, n)$-controllable if for any $e, e' \in C$, there exists a code sequence $e'' \in C$ with $P_m(e'') = P_m(e)$ and $P_n(e'') = P_n(e')$. A code $C$ is $\nu$-controllable if $C$ is $[k, k+\nu)$-controllable for all $k \in Z$. A code $C$ is strongly controllable if $C$ is $\nu$-controllable for some integer $\nu$. The minimum such $\nu$ is the controllability index (controller memory) of $C$.

In other words, a code $C$ is $\nu$-controllable if there exists a path of length $\nu$ connecting the past of any code sequence to the future of any other code sequence. A group code $C$ is $[m, n)$-controllable if and only if

$$ P_{[m, n)}(C) = P_m(C)P_n(C). $$

In other words, $C$ is $[m, n)$-controllable, if and only if the projection $P_{[m, n)}(C)$ is free as a code defined on the length-two time axis $J' = \{m^-, n^+\}$. This definition makes sense for a length-0 interval $[k, k)$: a code is $[k, k)$-controllable if $C = P_k(C)P_k(C)$. Thus,

1. $C$ is $[k, k)$-controllable if and only if $C$ is memoryless at time $k$;

2. $C$ is $0$-controllable if and only if $C$ is memoryless.

An example of an uncontrollable group code is a repetition code $C_r$ over an arbitrary group $G$ defined on the bi-infinite time axis $Z^*$:

$$ C_r \triangleq \{\ldots, g, g, g, \ldots: g \in G\}. $$

A repetition code is an example of an “autonomous” system [20], whose trajectories are entirely determined by “initial conditions” at time $-\infty$.

More generally, a code defined on $Z$ is uncontrollable whenever there are initial conditions whose effects persist indefinitely. The trellis diagram of such a code is then a reducible (disconnected) graph. If a code is regarded as a stochastic system, then strong controllability is an ergodic (mixing) property.

B. Strong Controllability Theorems

This section gives two fundamental theorems concerning $\nu$-controllable codes that suggest two different roads to minimal canonical encoders.

We have seen that the state code $\sigma(C)$ of a group code $C$ is itself a group code. If $C$ is strongly controllable, then the controllability index of $\sigma(C)$ may be determined as follows.

State Code Controllability Theorem: If a group code $C$ is $\nu$-controllable with $\nu \geq 1$, then the state code $\sigma(C)$ of $C$ is $(\nu - 1)$-controllable.

Proof: If $C$ is $\nu$-controllable, then for any time $k$ and any two sequences $e, e' \in C$, there exists a sequence $e'' \in C$ such that $P_k(e'') = P_k(e)$ and $P_{k+\nu}(e'') = P_{k+\nu}(e')$. This implies that there is a state sequence $\sigma(e'')$ connecting any past-induced state at time $k$ to any future-induced state at time $k+\nu$. But this implies that there is a state sequence $\sigma(e')$ that connects any past state sequence $P_{k+\nu}[\sigma(e')]$ to any future state sequence $P_{k+\nu}[\sigma(e')]$, which implies that the state code $\sigma(C)$ is $(\nu - 1)$-controllable. \hfill \Box

Note that the proof of this theorem depends only on the equivalence of past-induced and future-induced states. Therefore, it applies to any strongly controllable code $C$, not necessarily a group code, for which the state code of $C$ is essentially uniquely defined.

In view of this theorem, a logical and attractive approach to the analysis of a strongly controllable group code $C$ is to determine the series of state codes $C^{(0)} = C, C^{(1)} = \sigma(C), C^{(2)} = \sigma(C^{(1)}), \ldots$ (the “higher derivatives” of $C$). The $j$th derivative code $C^{(j)}$ is then $(\nu - j)$-controllable. The series terminates in the 0-controllable (memoryless) code $C^{(0)}$. This approach is pursued by Loeliger and Mittelholzer [26], who use it to derive a layered canonical minimal encoder for $C$.

This paper develops an alternative approach that arrives at a similar layered canonical minimal encoder. The two approaches give complementary insights into the dynamical structure of $C$. Our approach depends on a second theorem, which, unlike the previous theorem, depends in an essential way on the group property of $C$. It is helpful first to state a general lemma.

Lemma: If $C$ is a $\nu$-controllable group code, then any $e \in C$ may be written as a product $e_{(k+\nu)'} - e_{k+\nu}$ for any time $k$, where $e_{(k+\nu)'} = C_{(k+\nu)}$ and $e_{k+\nu} = C_k$.\hfill \Box

Proof: Since $C$ is $\nu$-controllable, there exists a sequence $b \in C_{k+\nu}$ that agrees with 0 on $k^-$ and with $e$ on $(k + \nu)^+$. It follows that $a = cb^{-1}$ is a code sequence in $C_{(k+\nu)}$ that agrees with $e$ on $k^-$ and with 0 on $(k + \nu)^+$, and that $e = ab$.\hfill \Box
Strong Controllability Theorem: A complete group code $C$ is $\nu$-controllable if and only if $C$ is the product of its code sequences of length $\nu + 1$ or less; i.e., if $C = \prod C[k,k+\nu]$.

Proof: Assume that $C = \prod C[k,k+\nu]$. In order to show that $C$ is $\nu$-controllable, we must show that for all $e, e' \in C$ and all times $j$, there exists a sequence $e'' \in C$ that agrees with $e$ on $j^-$ and with $e'$ on $(j+\nu)^+$. Since $C = \prod C[k,k+\nu]$, we may write $e = \prod e_k$ and $e' = \prod e'_k$, where $e_k, e'_k \in C[k,k+\nu]$ for each $k$. Then,

$$e'' = (\prod e_k)(\prod e'_k)$$

agrees with $e$ on $j^-$ and with $e'$ on $(j+\nu)^+$. The completeness of $C$ ensures that $e''$ is a sequence in $C$.

For the converse, assume that $C$ is $\nu$-controllable. In order to show that $C = \prod C[k,k+\nu]$, we need only show that any $e \in C$ may be written as a product $e = \prod e_k$ with $e_k \in C[k,k+\nu]$ for each $k$, since the completeness of $C$ ensures that $\prod C[k,k+\nu] \subseteq C$. By the lemma, we may first decompose $e$ into a product $c(k+v) - c(k+v-1)$, where $c(k+v-1)$ agrees with $e$ on $(j+\nu)^+$. This process may be iterated to decompose $c(k+1)$ into a sequence in $C(k+1)$ and $c(k)$ into $C(k)$, and so forth, by transfinite induction.

The practical import of the strong controllability theorem is that a strongly controllable code may be characterized by finite-length sequences.

C. 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 group $F_k$ of the state code $\sigma(C)$, or of the label code $q(C)$.

The $j$-controllable subcode $C_j$ of a group code $C$ is defined as the set of combinations of code sequences of length $j + 1$ or less:

$$C_j \triangleq \prod C[k,k+j]$$

If $C$ is complete, then $C_j$ is in fact a subcode of $C$. The 0-controllable subcode $C_0 = \prod C[k,k]$ is then the parallel transition code of $C$. If $C$ is complete and $\nu$-controllable, then, by the strong controllability theorem, $C_\nu = C$.

If $C$ is complete, then $C_j$ is clearly 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 $C[k,k+j]$. Thus,

$$C_0 \subseteq C_1 \subseteq \cdots \subseteq C_\nu = C$$

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

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

The quotient groups $C_j/C_{j-1}$, $1 \leq j \leq \nu$, may be further decomposed as follows.

Definition 10: For $k \in Z$ and $0 \leq j \leq \nu$, the granule $\Gamma_{k+k+j}$ of $C$ is defined as

$$\Gamma_{k+k+j} \triangleq C[k,k+j]/(C[k,k+j]C[k,k+j]).$$

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

The subcode $C[k,k+j]$ is a part of $C_j$, and the subcodes $C[k,k+j]$ and $C[k,k+j]$ are parts of $C_{j-1}$ that are subcodes of $C_j$. Indeed, if $C_j/k+k+j$ denotes the subcode of $C_j$ with support $[k, k+j]$,

$$C_j/k+k+j \triangleq \Gamma_{k+k+j}/C_j(k,k+j).$$

then, for $j \geq 1$,

$$C_j/k+k+j = C[k,k+j];$$

$$\Gamma_{k+k+j} = C[k,k+j]/C_{j-1}(k,k+j);$$

$$\Gamma_{k+k+j} = C[k,k+j]/C_{j-1}(k,k+j).$$

The granule $\Gamma_{k+k+j}$ thus represents what is needed to generate the sequences in $C[k,k+j]$ that are not in $C_{j-1}$.

Code Granule Theorem: If $C_j$ is the $j$-controllable subcode of a group code $C$, and $C_j$, $C_j$, and $C$ are complete, then $C_j/C_{j-1}$ is isomorphic to the direct product $\prod \Gamma_{k+k+j}$.

Proof (see Fig. 13): By the definition of $C_j$, $C_j = C_j/C_{j-1}$. From the strong controllability theorem, $C_j = \prod C[k,k+j]$.

Thus,

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

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

$$C[k,k+j]C_{j-1} \simeq C[k,k+j]/(C[k,k+j]C_{j-1}) = \Gamma_{k+k+j}.$$

Definition 11: If $C$ is a $\nu$-controllable group code, its $j$th input group at time $k$ is the image $F_{k,j} = P_{k,j}(C[k,k+j])$ of $C[k,k+j]$ under the projection $P_{k,j}$, for $0 \leq j \leq \nu$. 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,

$$F_{k,\nu} = P_{k,k}(C[k,k]) = F_k.$$

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

State Granule Theorem: Let \( \sigma_k(C_j) \) be the \( j \)-controllable state space at time \( k \) of a complete, \( \nu \)-controllable, group code \( C \), for \( 1 \leq j \leq \nu \) and \( k \in \mathbb{Z} \). Then the quotient group \( \sigma_k(C_j)/\sigma_k(C_j-1) \) is isomorphic to the direct product

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

**Proof (see Fig. 15):** The kernel of the state space map \( \sigma_k \): 

\[ C_j, [k-j, k+j], C_j-1, C_k- C_k+ \]

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

\[ C_j, [k-j, k+j], C_j-1, C_k- C_k+ \]

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

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

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

\[
\sigma_k(C_j)/\sigma_k(C_j-1) \simeq (C_j, C_k- C_k+)/ (C_j-1, C_k- C_k+)
\]

\[
\simeq C_j, [k-j, k+j] / C_j-1, [k-j, k+j].
\]

Finally, replacing \( C \) by \( C_j, [k-j, k+j] \) in the code granule theorem shows that

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

The state granule theorem thus defines a decomposition of the state space \( \Sigma_k \) into a Cartesian product (not necessarily a direct product) of direct products of granules:

\[ \Sigma_k \leftrightarrow \bigotimes_{1 \leq j \leq \nu} \Gamma_{i+i+j}. \]

From this decomposition, we can determine the sizes of the state spaces of \( C \).
The label input group \(G_k \subseteq G_{k+1} \subseteq \cdots \subseteq G_{k+\nu} = G_k\), and the granules \(G_k, G_{k+1}, \ldots, G_{k+\nu}\) of \(G\) are finite, for \(1 \leq j \leq \nu\), then the size of the state space \(\Sigma_k\) of \(G\) at time \(k\) is

\[
|\Sigma_k| = \prod_{1 \leq j \leq \nu} \prod_{i \in [k-j, k-1]} |G_{[k, i+j]}| = \prod_{1 \leq j \leq \nu} \prod_{i \in [k-j, k-1]} |F_{i,j}/F_{i,j-1}|.
\]

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

\[
|\Sigma_k| = \prod_{1 \leq j \leq \nu} |F_j/F_{j-1}|^2.
\]

Finally, consider the label code \(q(G)\). Since \(G_0\) is the kernel of the label map \(q\), the \(j\)-controllable subcodes of the label code \(q(G)\) are equal to \(q(G_{[k,j+1]}), 1 \leq j \leq \nu\). Of course the 0-controllable subcode of \(q(G)\) is \(q(G_0) = \{0\}\). The quotients \(q(G_{[k,j+1]})/q(G_{[k,j+1]})\) are isomorphic to the quotients \(G_{[k,j+1]}/G_{[k,j+1]}, 1 \leq j \leq \nu\). By a simple extension of the code granule theorem, we have the following theorem.

**Label Granule Theorem:** The code granules of the label code \(q(G)\) are isomorphic to the code granules of \(G\) for \(1 \leq j \leq \nu\):

\[
q(G_{[k,j+1]})/q(G_{[k,j+1]}) \cong G_{[k,j+1]}.
\]

The label input chain is the image of the input chain under the label map \(q_k\):

\[
\{0\} = q_k(G_{[k,k]}), 1 \leq j \leq \nu.
\]

Since \(G_{[k,k]}\) is the kernel of \(q_k\), corresponding quotient groups of the input and label input chains are isomorphic. Therefore, from the input granule theorem,

\[
q_k(G_{[k,j+1]})/q_k(G_{[k,j+1]}) \cong G_{[k,j+1]}.
\]

The label input group \(q_k(G_k)\) thus has a chain coset decomposition into components isomorphic to the dynamical granules \(\{G_{[k,j+1]}\}, 1 \leq j \leq \nu\).

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

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

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

\[
|\Sigma_k| = \prod_{1 \leq j \leq \nu} |\Gamma_{[k, j+1]}/\Gamma_{[k, j+1]}|^2.
\]

Finally, consider the label code \(q(C)\). Since \(G_0\) is the kernel of the label map \(q\), the \(j\)-controllable subcodes of the label code \(q(C)\) are equal to \(q(G_{[k,j+1]}), 1 \leq j \leq \nu\). Of course the 0-controllable subcode of \(q(C)\) is \(q(G_0) = \{0\}\). The quotients \(q(G_{[k,j+1]})/q(G_{[k,j+1]})\) are isomorphic to the quotients \(G_{[k,j+1]}/G_{[k,j+1]}, 1 \leq j \leq \nu\). By a simple extension of the code granule theorem, we have the following theorem.

**Label Granule Theorem:** The code granules of the label code \(q(C)\) are isomorphic to the code granules of \(C\) for \(1 \leq j \leq \nu\):

\[
q(G_{[k,j+1]})/q(G_{[k,j+1]}) \cong G_{[k,j+1]}.
\]

The label input chain is the image of the input chain under the label map \(q_k\):

\[
\{0\} = q_k(G_{[k,k]}), 1 \leq j \leq \nu.
\]

Since \(G_{[k,k]}\) is the kernel of \(q_k\), corresponding quotient groups of the input and label input chains are isomorphic. Therefore, from the input granule theorem,

\[
q_k(G_{[k,j+1]})/q_k(G_{[k,j+1]}) \cong G_{[k,j+1]}.
\]

The label input group \(q_k(G_k)\) thus has a chain coset decomposition into components isomorphic to the dynamical granules \(\{G_{[k,j+1]}\}, 1 \leq j \leq \nu\).

**Generator Theorem:** If \(G_C = \{G_{[k,k+\nu]}\}, k \in \mathbb{Z}, 0 \leq j \leq \nu\) is a set of representatives for the granules of a complete, \(\nu\)-controllable group code \(C\) whose \(j\)-controllable subcodes are also complete, then every code sequence can be uniquely expressed as a product \(e = \prod_{0 \leq j \leq \nu} \prod_{k \in \mathbb{Z}} e(\gamma_{k,j})\) of generators in \(G_C\).

A canonical minimal encoder may be constructed from such a set \(G_C\) as follows. We think of the granule representatives as the "impulse responses" of the encoder.

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 \(\Gamma_{[k,k]} = \Gamma_{[k,k]}\) and input granules \(\gamma_{k,j} \in \Gamma_{[k,k+1]}, 1 \leq j \leq \nu\). 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 combination of the generator components \(e(\gamma_{k,j}), 0 \leq j \leq \nu, k-j \leq i \leq k\). This encoder structure is illustrated in Fig. 16.

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

A granule \(\Gamma_{[k,k+\nu]}\) corresponds to an input granule at time \(k\). If \(j \geq 1\), it is active as a code granule during the interval \([k, k+1]\). It is active as a state granule only during \([k+1, k+j]\); that is, its length as a state granule is one less
than its length as a code granule. This observation can be used to prove the state code controllability theorem for group codes.

The encoder is minimal, because by construction there is a one-to-one correspondence between its state space \( X_k \) at time \( k \) and \( \Sigma_k \):

\[
X_k \leftrightarrow \bigotimes_{1 \leq j \leq l \leq k-j, k-1} \Gamma_{[k,i+j]} \rightarrow \Sigma_k.
\]

The trellis diagram of any minimal encoder is the trellis diagram of \( C \). Thus, the trellis diagram of this encoder is the trellis diagram of \( C \).

The encoder is layered, in the sense that if the input granules \( \gamma_k, \gamma_{k'} \) are set equal to zero for \( j+1 \leq j' \leq \nu \) and all \( k \in \mathbb{Z} \), 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 will have 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 \( \Sigma_{\text{eternal}} \), where \( \Sigma_{\text{cont}} \) is the controllable subcode of \( C \). In this case, \( \Sigma_{\text{eternal}} \) will then consist of the union of \( \Sigma_{\text{cont}} \) cosets of \( C_{\text{cont}} \), none reachable from any other. An autonomous group system corresponds to an uncontrollable code \( C \) with \( C_{\text{cont}} = \{0\} \), and is dynamically equivalent to an infinite repetition code over \( \Sigma_{\text{eternal}} \).

Since \( \Gamma_{[k,k+j]} \approx F_{k,j}/F_{k,j-1} \), as shown in the proof of the input granule theorem by correspondence under projection by \( F_{k,j} \), it is always possible to take the first outputs \( c_k(\gamma_{k,j}) \) of the generators \( \{c_k(\gamma_{k,j}) : \gamma_{k,j} \in \Gamma_{[k,k+j]}\} \) 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 called monic; the first nonzero output of the encoder is the first nonzero input. We may then write the sets of generators as \( \{f_k(j) : f_k(j) \in [F_{k,j}]/F_{k,j-1}\} \).

**Example 1 (cont.):** Consider once again the time-invariant code \( C \) over \( (Z_4)^2 \) whose encoder is shown in Fig. 1. Obviously the controllability index of \( C \) is \( \nu = 1 \). The shortest nonzero code sequences are the shifts of the parallel transition group \( C_{[k,k+1]} = \Gamma_{[k,k+1]} \), which is \( \{00, 22\} \). The subcode \( C_{[k+1,k+1]} \) has eight elements, including the four sequences in \( C_{[k+1,k+1]} \) generated by \( C_{[k,k+1]} \). The granule \( \Gamma_{[k,k+1]} \) is trivially \( 3 \)-controllable.

**Example 4 (cont.):** The four nontrivial granules of the linear \( (8, 4, 4) \) binary block code \( C \) of Fig. 5 are each two-element groups, with representatives as follows:

\[
\begin{align*}
\Gamma_{[1,2]} &= C_{[1,2]} = \{0, 11, 11, 00, 00\}; \\
\Gamma_{[2,3]} &= C_{[2,3]} = \{0, 00, 11, 11, 00\}; \\
\Gamma_{[3,4]} &= C_{[3,4]} = \{0, 00, 11, 11, 11\}; \\
\Gamma_{[4,5]} &= C_{[4,5]}/C_{[1,2]}C_{[2,3]}C_{[3,4]}; \\
\end{align*}
\]

This yields the trellis diagram of Fig. 8. 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 Fig. 8; and \( C \) is trivially \( 3 \)-controllable, \( C_3 = C_{[1,4]} = C \).

**VII. CONCLUSION**

For any code \( C \) that has a group property—a block code, convolutional code, lattice, trellis code, geometrically uniform code, or whatever—the results of this paper give a consistent method of characterizing \( C \) by a minimal encoder or trellis diagram, once the time axis of \( C \) is specified. Such descriptions can replace the ad hoc constructions used in previous work.

Consequently, the “complexity” of a linear code \( C \) is a well-defined concept. Usually the size of the state space is taken as a measure of code complexity, although the size of the branch space is more meaningful in the context of Viterbi algorithm decoding. One can then use “performance versus complexity” as a code design criterion. For example, for Euclidean-space codes, one is more interested in “coding gain” (packing density) versus complexity than versus dimension, which is the usual measure in the sphere-packing literature.

For a linear time-invariant convolutional code \( C \) over a field, these results replicate known results, but do not invoke either the multiplicative structure of the field or time invariance. All that is required is a set of shortest code sequences that generate \( C \), which in turn specify its minimal encoder and trellis diagram.

For linear system theory, these results give a surprisingly elementary answer to the question of finding a minimal state realization for a set \( C \) of trajectories that has the group property.

Subsequent papers will develop dual minimal observers/syn- drome-formers, study dual codes, and consider controllability and observability in group systems.

For coding purposes, both the distance properties and the group properties of a group code \( C \) are important. A related paper gives some elementary results on the Hamming distance properties of group codes [36].

The results of this paper show that all group codes may be constructed by starting with a label code as a skeleton and using group extensions to flesh it out. Although the subject of group extensions is vast and unfinished, this insight should be helpful in the construction of new codes.
In summary, the elementary results presented here probably only scratch the surface of a rich research area.

**APPENDIX**

**SOME ELEMENTARY GROUP THEORY**

This appendix gives a brief summary of the group theory needed in this paper. Its purpose is not so much to teach group theory as to show how little of elementary group theory is required.

A group \( G \) is a set that is closed under an associative binary operation \( \ast \) and that has an identity element \( e \in G \) and an inverse \( g^{-1} \) for every \( g \in G \). The group \( G \) is abelian if the operation \( \ast \) is commutative; i.e., if \( g \ast h = h \ast g \) for all \( g, h \in G \).

Given a map \( f : G \rightarrow Q \), the kernel of \( f \) is the set of elements of \( G \) that map to inverses in \( Q \), and the inverse image \( f^{-1}(q) \) of any element \( q \) of \( Q \) is a normal subgroup of \( G \) that includes \( e \) and \( f^{-1}(q) \) as 0. We sometimes refer to the product \( g \ast h \) as the combination \( g + h \) of \( g \) and \( h \) or as a sum when \( G \) is abelian.

If \( G \) and \( Q \) are groups, then a map \( f : G \rightarrow Q \) is a homomorphism if it takes the binary operation of \( G \) to the binary operation of \( Q \); that is, if, for all \( g, h \in G \), \( f(gh) = f(g)f(h) \). The kernel of \( f \) is the set of elements of \( G \) that map to the identity of \( Q \). A homomorphism sends the identity of \( Q \) to inverses in \( Q \) and inverses in \( G \) to inverses in \( Q \), and subgroups of \( G \) to subgroups of \( Q \). In particular, the group \( G \) is sent to a subgroup of \( Q \) called the image \( f(G) \) of \( G \).

An isomorphism is a homomorphism that is both one-to-one and onto. A homomorphism \( f : G \rightarrow G' \) is an isomorphism if and only if its kernel is the identity of \( G \) and its image is \( G' \). Then, \( G \) is said to be isomorphic to \( G' \), written \( G \cong G' \). An isomorphism \( f : G \rightarrow G' \) has an inverse isomorphism \( f^{-1} : G' \rightarrow G \).

A subgroup \( H \) of \( G \) is normal (self-conjugate) in \( G \) if its left cosets \( gH \) are equal to its right cosets \( Hg \). The cosets of \( H \) in \( G \) then form a group under the group operation of \( G \), modulo \( H \), called the quotient group \( G/H \). The natural map \( \phi : G \rightarrow G/H \) defined by \( \phi(g) = gH \) is then a homomorphism with kernel \( H \) and image \( G/H \). Every normal subgroup \( H \) of \( G \) is thus the kernel of a homomorphism, namely, the natural map.

Given a map \( f : G \rightarrow Q \), the inverse image of a subset \( S \) of \( Q \) is the set \( f^{-1}(S) = \{ g \in G : f(g) \in S \} \). The kernel of a homomorphism \( f \) is the inverse image \( K = f^{-1}(0) \) of the identity of \( Q \), and the inverse image \( f^{-1}(g) \) of any \( g \in f(G) \) is the coset \( gK \).

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 of each coset, so that there is a one-to-one correspondence \([G/H] \leftrightarrow G/H\). Every coset may then be uniquely written as \( Hr \) or \( hr \) with \( r \in [G/H] \), and any element \( g \in G \) may be uniquely written as \( g = hr \) with \( h \in H \) and \( r \in [G/H] \). The latter defines a one-to-one correspondence \( m : G \rightarrow H \times [G/H] \) (not an isomorphism), called a coset decomposition, or equivalently a one-to-one correspondence \( m' : G \rightarrow H \times (G/H) \). The inverse one-to-one map \( f : H \times (G/H) \rightarrow G \) defined by \( f(h, rH) = hr \) is called an extension of \( G/H \) by \( H \). (Most authors would say "an extension of \( H \) by \( G/H \)," but in this paper the quotient group is generally the more basic building block.)

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 \( 1 \leq j \leq n \). By induction, if \( f([G_j/G_{j-1}] : 1 \leq j \leq n) \) is a corresponding series of systems of coset representatives, then every \( g \in G_n \) may be uniquely written as \( g = r_0 r_1 \cdots r_n \) with \( r_0 \in G_0 \) and \( r_j \in [G_j/G_{j-1}] \), \( 1 \leq j \leq n \). A normal series defines a corresponding series of homomorphisms called a normal series of subgroups of \( G \). In particular, the group \( G \) is sent to a subgroup of \( Q \) called the image \( f(G) \) of \( G \).

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 \( 1 \leq j \leq n \). By induction, if \( f([G_j/G_{j-1}] : 1 \leq j \leq n) \) is a corresponding series of systems of coset representatives, then every \( g \in G_n \) may be uniquely written as \( g = r_0 r_1 \cdots r_n \) with \( r_0 \in G_0 \) and \( r_j \in [G_j/G_{j-1}] \), \( 1 \leq j \leq n \). This is called a chain coset decomposition, and establishes a one-to-one correspondence

\[ G_n \leftrightarrow G_0 \times [G_1/G_0] \times \cdots \times [G_n/G_{n-1}] \]

or equivalently a one-to-one correspondence \( G_n \leftrightarrow G_0 \times (G_1/G_0) \times \cdots \times (G_n/G_{n-1}) \).

The following are usually regarded as the basic isomorphism theorems.

**First Isomorphism Theorem (Fundamental Homomorphism 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:** If \( Q = f(G) \), then \( G/K \cong Q \).

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 \( Q \); i.e.,

\[ G \xrightarrow{\phi} G/K \rightarrow f(G) \xrightarrow{\iota} Q. \tag{1} \]

The second isomorphism theorem applies when \( G \) has two subgroups \( H \) and \( J \), at least one of which is normal in \( G \). The intersection \( H \cap J = \{ g \in G : g \in H, g \in J \} \) is then a group, the greatest common subgroup of \( H \) and \( J \). If \( H \) is normal in \( G \), then the product \( HJ = \{ hJ : h \in H, J \} = \{ Hj : j \in J \} = \{ jH : j \in J \} = JH \) is a group, the least common supergroup of \( H \) and \( J \).

In this paper, only the case in which both \( H \) and \( J \) are normal in \( G \) is needed. If \( H \) and \( J \) are both normal, then their product \( HQ \) is normal, and the second isomorphism theorem may be stated as follows.

**Second Isomorphism Theorem (Two Normal Subgroups):** If \( H \) and \( J \) are normal subgroups of \( G \), then \( HQ \) is a normal subgroup of \( G \), if \( H \) and \( J \) are normal subgroups of \( JH \), then \( H \cap J \) is a normal subgroup of \( H \) and \( J \), \( HJ/H \cong J/(H \cap J) \), and \( HJ/J \cong H/(H \cap J) \).

Fig. 17 illustrates the second isomorphism theorem in this case.
The third isomorphism theorem also involves normal subgroups of $G$. It is illustrated by the normal subgroups and corresponding quotient groups of Fig. 18.

**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 $(G/J)/(H/J) \simeq G/H$.

![Fig. 18. Third isomorphism theorem.](image)

Corollary: If $H \cap J = \{0\}$, then $HJ/H \simeq J$ and $HJ/J \simeq H$.

A product $HJ$ 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 $H \times J = \{(h, j) : h \in H, j \in J\}$, namely the Cartesian product of $H$ and $J$, with the group operation defined componentwise. If both $H$ and $J$ are normal in $HJ$, then $HJ$ is an internal direct product if and only if $H \cap J = \{0\}$.

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 Cartesian product set of all sequences $g = (g_k \in G_k, k \in I)$, with the group operation defined componentwise by the group operations of the groups $G_k$. The index set may be infinite. In this paper it is always assumed to be countable. The set $W_k$ of $g \in W$ such that $g_k = 0$ for $k \neq k'$ is a normal subgroup of $W$ that is obviously 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 $\prod_{j \in J} J$ 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 finite, then the product $\prod_{j \in J} H_j$ is taken to be meaningful only if for each product $\prod_{j \in J} h_j$ and each $k \in I$, only finitely many of the sequences $h_j$ have a nonzero $k$th component $h_{jk}$. All infinite products in this paper 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 $\{H_j : j \in J\}$.

The third isomorphism theorem also involves normal subgroups of $G$. It is illustrated by the normal subgroups and corresponding quotient groups of Fig. 18.

**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

$$f(G)/f(J) \simeq G/J;$$

$$f(HJ)/f(J) \simeq HJ/J \simeq H/(H \cap J);$$

$$f(G)/f(HJ) \simeq G/HJ \simeq (G/J)/(H/J).$$

This theorem is illustrated in Fig. 20.
ACKNOWLEDGMENTS

The authors have benefited greatly from continuing dialogues with H.-A. Loeliger, J. L. Massey, and T. Mittelholzer, who have been exploring similar topics. Although we have tried to reference their work appropriately, we wish to say that as a result of our free and open interchange, it has become difficult to disentangle credit; where results overlap, we wish to say that as a result of our free and open interchange, it has become difficult to disentangle credit; where results overlap, we wish to say that as a result of our free and open interchange, it has become difficult to disentangle credit; where results overlap, we wish to say that as a result of our free and open interchange, it has become difficult to disentangle credit; where results overlap, we wish to say that as a result of our free and open interchange, it has become difficult to disentangle credit; where results overlap, we wish to say that as a result of our free and open interchange, it has become difficult to disentangle credit; where results overlap.
学霸图书馆
www.xuebalib.com

本文献由“学霸图书馆-文献云下载”收集自网络，仅供学习交流使用。

学霸图书馆（www.xuebalib.com）是一个“整合众多图书馆数据库资源，提供一站式文献检索和下载服务”的24小时在线不限IP图书馆。

图书馆致力于便利、促进学习与科研，提供最强文献下载服务。

图书馆导航:
图书馆首页 文献云下载 图书馆入口 外文数据库大全 疑难文献辅助工具