

MULTILINGUAL DICTIONARY

Dave Forney
Motorola
20 Cabot Boulevard
Mansfield, MA 02048
Luse27@email.mot.com

Brian Marcus
IBM K53-802
650 Harry Rd.
San Jose, CA 95120
marcus@almaden.ibm.com

N.T. Sindhushayana
Qualcomm Inc.
6455 Lusk Blvd.
San Diego, CA, 92121
nsindhushayana@qualcomm.com

Mitchell Trott
Department of Electrical Engineering
MIT 35-213
Cambridge, MA, 02139
trott@lids.mit.edu

Introduction:

System theory, coding theory, symbolic dynamics and automata theory are subjects which have a lot in common, but have developed somewhat independently. So, it is not surprising that very similar concepts with very different terminology have arisen in these subjects. This dictionary is an attempt to bring together the various terms for these concepts.

We have divided the material into several sections. Within each section, the items are displayed in an $n \times 4$ table; each row represents a concept, and the columns represent the four subjects of this dictionary. Following the table, you will find, for each concept, a definition together with (sometimes) a brief discussion. The definition is usually given in terms of system theory, but occasionally in the more natural mother tongue. As with natural languages, often something is lost in the translation; in the discussion, we try to explain how the terms for each concept differ (if they do) as well as to indicate some relations among the different concepts. Most of the terms in this dictionary are fairly well established, but some were made up to meet the needs of the dictionary.

There is an *Index of Terms*, in alphabetical order, at the end of the Dictionary.

For the purposes of this dictionary, system theory is interpreted to mean the behavioral theory of systems. Coding theory is taken to mean coding theory and information theory. And within symbolic dynamics, we have included terminology from the field of constrained coding (which has adopted some terms from the theory

of state machines popular in the '50's and '60's).

While we have not made any specific references in this paper, we drew heavily from the papers listed in the bibliography of the article by B. Marcus in this volume.

We are happy to thank Natasha Jonoska, Andi Loeliger, Jan Willems and Susan Williams for their helpful comments on the Dictionary. We thank Yvonne Marcus for help with the index, useful remarks and various wisecracks.

An earlier version of this dictionary was presented at the IEEE-IT Workshop on Coding, System Theory and Symbolic Dynamics, held in Mansfield, MA., October 18-20, 1993. We thank the National Science Foundation and National Security Agency for their support of that workshop.

Before we begin, we would like to present one instance where the same term has been used for different concepts in two of the subjects:

code (*kōd*) *n.* 1. [Coding Th.] A set of sequences. 2. [Symbolic Dyn.] A rule for mapping one set of sequences to another set of sequences.

Actually, the usage in symbolic dynamics corresponds to an old usage in coding theory; "Morse code" refers not just to a set of signal sequences, but also a rule for mapping text into signal sequences. In modern coding theory, a mapping rule is called an "encoder."

TABLE OF CONTENTS:

I. Basic concepts of system	4
II. Concepts of memory	9
III. Concepts of State Representation	13
IV. Properties of state representation (state realization) . . .	18
V. Concepts of input/output	25
VI. Concepts of input/output machine	30
VII. Concepts of linearity	33
VIII. Other graph concepts	38
IX. Entropy	41
Index of Terms	42

KEY:

ST = System theory (behavioral)

CT = Coding theory

SD = Symbolic dynamics (including constrained coding)

AT = Automata theory

I. Basic concepts of system

Sect.	System Theory	Coding Theory	Symbolic Dynamics	Automata Theory
I-1	Dynamical system	Code	—	Language
I-2	Complete	—	Closed (compact)	—
I-3	Completion	—	Closure	—
I-4	Time-invariant	Time-invariant	Shift-invariant	—
I-5	Trim	—	—	—
I-6	Finite alphabet, discrete time, complete, time- invariant system	—	Shift space (subshift)	—

I-1.
 ST. Dynamical system (system)
 CT. Code
 SD. —
 AT. Language

Defn: A *dynamical system* (or *system*) is a triple (T, W, \mathcal{B}) where T is a *time axis*, W is a *signal alphabet* (or simply *alphabet*), and the *behavior* $\mathcal{B} \subseteq W^T$ is a set of sequences with index set T and alphabet W .

For instance, when $T = \mathcal{Z}$, the set of integers, the behavior \mathcal{B} is a set of bi-infinite sequences.

In system theory, usually $T = \mathcal{Z}$ (discrete time), or $T = \mathcal{R}$, the set of real numbers (continuous time), while W and \mathcal{B} are arbitrary. In linear system theory, W is usually

a finite dimensional real vector space and \mathcal{B} is a subspace of W^T .

In coding theory, usually $T = \mathcal{Z}$ (corresponding to convolutional codes, defined in VII-1) or T is a finite subinterval of \mathcal{Z} (corresponding to block codes). A *code* is a set of sequences defined on one of these time axes T . Usually, W is finite, or at least discrete, and has some algebraic structure such as vector space, module or group structure. Then \mathcal{B} is usually an algebraic sub-object (subspace, submodule, subgroup) of W^T .

In symbolic dynamics, usually $T = \mathcal{Z}$ and W is finite. The behavior \mathcal{B} must satisfy some additional conditions (see I-6 below). But there has been significant work done for time axes $T = \mathcal{Z}^+, \mathcal{R}, \mathcal{Z}^n$ and for infinite alphabets as well.

In automata theory, $T = \mathcal{Z}$, W is finite, and the behavior is defined by a *language* \mathcal{L} , which is a set of finite sequences (of arbitrary lengths) over a finite alphabet A ; more precisely, given a language \mathcal{L} , introduce a symbol e , not in A , and define \mathcal{B} to be the set of all bi-infinite sequences over $W = A \cup \{e\}$ of the form

$$\dots eeeweee\dots, w \in \mathcal{L}.$$

where the ‘time 0’ coordinate can appear anywhere. In automata theory, the focus is on the language \mathcal{L} , not the behavior \mathcal{B} . There has also been significant work done in automata theory on infinite sequences.

Useful Notation: For a subset $J \subseteq T$ and $x \in W^T$, $x|_J$ denotes the sequence x restricted to the indices in J . In particular, when $T = \mathcal{Z}$ and $i \leq j$, $x|_{[i,j]}$, called a *block* or *word*, denotes the restriction of the sequence x to the indices $k : i \leq k \leq j$. For $J \subseteq T$, we have the *projection*

$$\mathcal{B}|_J = \{x|_J : x \in \mathcal{B}\}.$$

Whenever the alphabet W of a system (T, W, \mathcal{B}) is decomposed as a direct product $W = W_1 \times W_2$, we denote

$$P_{W_1}\mathcal{B} = \{w_1 : (w_1, w_2) \in \mathcal{B} \text{ for some } w_2\}.$$

and similarly for $P_{W_2}\mathcal{B}$.

- I-2.
- ST. Complete
- CT. —

SD. Closed or Compact

AT. —

Defn: A system is *complete* if

$$x \in \mathcal{B} \text{ iff for all finite } J \subset T, x|_J \in \mathcal{B}|_J.$$

Suppose that $T = \mathcal{Z}$, W is finite and we endow W^T with any metric d such that $d(x, y)$ is ‘small’ iff for ‘large’ n , $x|_{[-n, n]} = y|_{[-n, n]}$; for instance, let ρ be any metric on W , and define

$$d(x, y) = \sum_{k=-\infty}^{\infty} \frac{\rho(x_k, y_k)}{2^{|k|}}.$$

Then complete = closed = compact (as subsets of the metric space W^T).

I-3.

ST. Completion

CT. —

SD. Closure

AT. —

Defn: The *completion* $(T, W, \bar{\mathcal{B}})$ of a system (T, W, \mathcal{B}) is the smallest complete system that contains the system.

Equivalently, $(T, W, \bar{\mathcal{B}})$ is the set of all $x \in W^T$ such that for all finite $J \subset T$ $x|_J \in \mathcal{B}|_J$.

I-4.

ST. Time-invariant

CT. Time-invariant

SD. Shift-invariant

AT. —

Defn: Suppose that T has some notion of addition (e.g., $T = \mathcal{Z}, \mathcal{Z}^+, \mathcal{R}, \mathcal{Z}^n, \mathcal{R}^n$). Define the shift map $\sigma_i : W^T \rightarrow W^T$ by $(\sigma_i(x))_k = x_{i+k}$, $k, i \in T$. A system is *time-invariant* if $\forall i \sigma_i(\mathcal{B}) = \mathcal{B}$.

Some prefer inclusion rather than equality in the definition of time-invariance.

Observe that for a time-invariant system, $\forall k \in T, J \subset T$, we have $\mathcal{B}|_{k+J} = \mathcal{B}|_J$. Note that when $T = \mathbb{Z}$ a system is time-invariant iff $\sigma_1(\mathcal{B}) = \mathcal{B}$.

In coding theory, the delay operator $D = \sigma^{-1}$. In symbolic dynamics, σ_1 is denoted simply σ .

I-5.
ST. Trim
CT. —
SD. —
AT. —

Defn: A system (T, W, \mathcal{B}) is *trim* if each element of W appears in some element of \mathcal{B} .

I-6.
ST. Finite-alphabet, complete, time-invariant system, with $T = \mathbb{Z}$
CT. —
SD. Shift space
AT. —

Defn: A *shift space* is a finite-alphabet, complete, time-invariant system with $T = \mathbb{Z}$.

Note on classical dynamical systems: The notion of dynamical system that we have given here differs from the classical notion. A classical (discrete-time) dynamical system is a pair (X, f) where X is a set and $f : X \rightarrow X$ is an invertible mapping; usually, X has some structure and f respects that structure. Given a partition P of X , we get a time-invariant system in the sense of this dictionary by setting $T = \mathbb{Z}$, $W = P$ and \mathcal{B}_P to be the subset of $P^{\mathbb{Z}}$ which describes the orbits of f - i.e., \mathcal{B}_P is the set of all sequences $\dots p_{-1}p_0p_1\dots$ for which there is an $x \in X$ such that $f^i(x) \in p_i$. Symbolic dynamics models classical systems (X, f) using finite partitions P , by replacing (X, f) with a shift space \mathcal{B} that is roughly (but not exactly) the completion of \mathcal{B}_P . Noninvertible f are also considered (giving rise to systems with $T = \mathbb{Z}^+$).

For ease of exposition, we hereafter make the following

STANDING ASSUMPTIONS:

1. **Discrete time** ($T = \mathcal{Z}$)
2. **Time-Invariance**
3. **Trim**

This leaves out some very important cases: continuous-time systems in system theory, block codes in coding theory, and multidimensional systems. However, many of the concepts in the dictionary can be extended to these settings.

II. Concepts of memory

Sect.	System Theory	Coding Theory	Symbolic Dynamics	Automata Theory
II-1	L -complete system	—	L -step Shift of Finite Type (L -step SFT)	—
II-2	Strongly complete system	—	Shift of Finite Type (SFT)	—
II-3	L -observable system	—	—	—
II-4	Strongly observable system	—	—	—
II-5	Memoryless system	—	—	—
II-6	Free system	Sequence space	Fullshift	—
II-7	Controllable system	—	Irreducible (Topologically transitive) shift space	—
II-8	Strongly controllable system	Finite-constraint length code	Mixing shift space	—

II-1.
 ST. L -complete system
 CT. —

SD. L -step Shift of Finite Type (abbr., L -step SFT)

AT. —

Defn: A system is L -complete if

$$x \in \mathcal{B} \text{ iff } \forall i \in T \ x|_{[i, i+L]} \in \mathcal{B}|_{[0, L]}.$$

In other words, if a sequence looks like an element of \mathcal{B} through all windows of length $L + 1$, then it is in fact in \mathcal{B} .

The term ‘ L -step SFT’ usually assumes finite alphabet.

Fact: L -complete \Rightarrow complete.

Note that completeness is invariant under reordering of the time axis; L -completeness is not.

II-2.

ST. Strongly complete

CT. —

SD. Shift of finite type (abbr., SFT) or Finite memory system

AT. —

Defn: A system is *strongly complete* if it is L -complete for some L .

The term ‘SFT’ usually assumes finite alphabet. The term ‘finite memory system’ is used in constrained coding.

II-3.

ST. L -observable system

CT. —

SD. —

AT. —

Defn: A system is L -observable if whenever

$$x, y \in \mathcal{B} \text{ and } x|_{[0, L]} = y|_{[0, L]}$$

then

$$x|_{(-\infty, 0)} y|_{[0, \infty)} \in \mathcal{B}.$$

Fact: L -observable + complete = L -complete.

An L -observable system is sometimes called L -finite memory.

II-4.

ST. Strongly observable system

CT. —

SD. —

AT. —

Defn: A system is *strongly observable* if it is L -observable for some L . The least such L is called the *observability index* or *observer memory* of the system.

Fact: Strongly observable + complete = Strongly complete. So, a shift of finite type is the same as a strongly observable shift space.

II-5.

ST. Memoryless system

CT. —

SD. —

AT. —

Defn: A system is *memoryless* if it is 0-observable.

II-6.

ST. Free system

CT. Sequence space

SD. Full shift

AT. —

Defn: A system is *free* if it is 0-complete.

The term 'Full shift' usually assumes finite alphabet.

Fact: Free = Memoryless + Complete

Fact: A free system is the Cartesian product W^T .

Example: Let W be a group with identity element e . Let $W_f^{\mathcal{Z}}$ be the system which consists of all sequences $x \in W^{\mathcal{Z}}$ such that $x_n = e$ for all sufficiently large $|n|$ (the subscript ‘ f ’ stands for ‘finite’ indicating that $W_f^{\mathcal{Z}}$ really represents the set of all finite sequences over the alphabet W). Observe that $W_f^{\mathcal{Z}}$ is memoryless, but unless W is the trivial group it is not complete and therefore not free.

II-7.

ST. Controllable system

CT. —

SD. Irreducible shift space or Topologically transitive shift space

AT. —

Defn: A system is *controllable* if $\forall x, y \in \mathcal{B}, \exists M \in \mathcal{Z}^+, z \in \mathcal{B}$ such that $z|_{(-\infty, 0]} = x|_{(-\infty, 0]}$ and $z|_{[M, \infty)} = y|_{[0, \infty)}$.¹

In symbolic dynamics, a shift space X is *irreducible* if whenever u, v are blocks that appear in elements of X , then there is a block w such that uvw appears in some element of X . This agrees with controllability for the special (but distinguished) class of sofic shifts defined in III-6 below.

II-8.

ST. Strongly controllable system

CT. Finite-constraint length code

SD. Mixing shift space

AT. —

Defn: A system is *strongly controllable* if $\exists M \in \mathcal{Z}^+$ satisfying: $\forall x, y \in \mathcal{B}, \exists z \in \mathcal{B}$ such that $z|_{(-\infty, 0]} = x|_{(-\infty, 0]}$ and $z|_{[M, \infty)} = y|_{[0, \infty)}$. The least such M is called the *controllability index* or *controller memory*.

Fact: 0-controllable = 0-observable = memoryless

The term ‘mixing’ assumes shift space and agrees with strong controllability for sofic shifts, defined in III-6 below.

¹This definition does not appear to generalize in a straightforward way to the time-varying case.

III. Concepts of State Representation

Sect.	System Theory	Coding Theory	Symbolic Dynamics	Automata Theory
III-1	State space system	Realization	—	—
III-2	External behavior	Code generated by a realization	—	—
III-3	State behavior	State code generated by a realization	—	—
III-4	Evolution law	Trellis section	Labeled graph (Finite state transition diagram)	automaton
III-5	State space system of an evolution law	Trellis	—	—
III-6	External behavior of an evolution law	Code generated by a trellis	Sofic shift (sofic system) (constrained system)	Regular language
III-7	—	—	Graph shift (Edge sequence shift)	—
III-8	Finite state, complete, state behavior	—	Topological Markov shift (State sequence shift)	—

III-1.
 ST. State space system
 CT. Realization
 SD. —
 AT. —

Defn: A *state space system* (T, W, X, \mathcal{B}_s) is a system $(T, W \times X, \mathcal{B}_s)$ that satisfies the *Axiom of State* – i.e., whenever

$$(w, x), (w', x') \in \mathcal{B}_s \text{ and } x_0 = x'_0,$$

then

$$(w, x)|_{(-\infty, 0)}(w', x')|_{[0, \infty)} \in \mathcal{B}_s.$$

III-2.
 ST. External behavior
 CT. Code generated by a realization
 SD. —
 AT. —

Defn: The *external behavior* of a state space system (T, W, X, \mathcal{B}_s) is the system $(T, W, P_W \mathcal{B}_s)$.

III-3.
 ST. State behavior
 CT. State code generated by a realization
 SD. —
 AT. —

Defn: The *state behavior* of a state space system (T, W, X, \mathcal{B}_s) is the system $(T, X, P_X \mathcal{B}_s)$.

III-4.
 ST. Evolution law
 CT. Trellis section
 SD. Labeled graph or Finite state transition diagram (FSTD)
 AT. Finite automaton

Defn: An *evolution law* is a quadruple (T, W, X, ∂) where T is a time axis, W is an alphabet, X is a set called the *state space*, and $\partial \subseteq X \times W \times X$, called the *next state relation*.

In linear system theory, the alphabet, state space and next state relation are usually finite-dimensional vector spaces. In coding theory, symbolic dynamics and automata theory, they are usually finite sets.

A *trellis section* is essentially the same as an evolution law, but the emphasis is on finite state spaces. The states are sometimes called nodes. Elements of the next state relation are sometimes called branches.

A *labeled graph* is a finite-alphabet, finite-state evolution law (although, strictly speaking, a labeled graph may have multiple copies of the same (x, w, x')). In symbolic dynamics, a labeled graph is usually defined as a pair (G, \mathcal{L}) . Here, G is a *graph* ('finite directed graph' in graph theory) and \mathcal{L} is a *labeling*; that is, $G = (\mathcal{V}, \mathcal{E}, s, t)$ consists of a finite set \mathcal{V} of vertices (states), a finite set \mathcal{E} of edges, initial state, terminal state functions $s, t : \mathcal{E} \rightarrow \mathcal{V}$, and \mathcal{L} is a function on \mathcal{E} ; the next state relation is then $\{(s(e), \mathcal{L}(e), t(e)) : e \in \mathcal{E}\}$. One usually makes the harmless assumption that each state has at least one outgoing edge and one incoming edge (roughly corresponding to the 'trim' concept). A *path* or *edge sequence* in a graph is a sequence (finite, bi-infinite, or semi-infinite) of edges e_i such that $t(e_i) = s(e_{i+1})$. A *state sequence* is the sequence of states of a path.

Finite-state transition diagram (FSTD) is the term used in constrained coding.

A *finite automaton* is a labeled graph, usually endowed with a specific subset of states called the *initial states* and a specific subset of states called the *terminal states*. However, the properties in section IV of finite automata are properties of the underlying labeled graph and do not involve the specific initial state and terminal states.

III-5.

ST. State space system of an evolution law

CT. Trellis or Trellis diagram

SD. —

AT. —

Defn: The *state space system* (T, W, X, \mathcal{B}_s) of an evolution law (T, W, X, ∂) is

defined by

$$\mathcal{B}_s = \{(w, x) \in W^T \times X^T : (x_k, w_k, x_{k+1}) \in \partial \text{ for each } k \in T\}.$$

The term *trellis* usually assumes finite-state.

III-6.

ST. External behavior of an evolution law

CT. Code generated by a trellis

SD. Sofic shift or sofic system or constrained system

AT. Regular language

Defn: The *external behavior of an evolution law* is the external behavior of the state space system of the evolution law.

A *sofic shift* is the external behavior of a finite-alphabet, finite-state evolution law – equivalently, the set of label sequences of bi-infinite paths in a labeled graph. A sofic shift is automatically a shift space. ‘Sofic’ is a Hebrew word meaning finite. The terms ‘sofic shift,’ ‘sofic system,’ and ‘constrained system’ (a term from constrained coding) are synonyms except that ‘constrained system’ is often taken to mean the set of label sequences of finite paths in a labeled graph.

A *regular language* is the language consisting of all label sequences of finite paths in a finite automaton that begin with an initial state of the automaton and end with a terminal state of the automaton.

The set of finite sequences that occur in a sofic shift is a special kind of regular language: it is closed under subwords and prolongable in the sense that for every word w that occurs there are symbols a and b such that awb also occurs.

Fact: The state space system of an evolution law is complete (and conversely).

Fact: The external behavior of an evolution law need not be complete, but the external behavior of a finite-state evolution law is always complete (see also VII-3 and VII-8).

III-7.

ST. —

CT. —

SD. Graph shift or Edge sequence shift

AT. —

(Recall the notions of graph and labeled graph from III-4).

Defn: A *graph shift* is the set of all bi-infinite paths of a graph – in other words, a sofic shift presented by a labeled graph in which all the edges are labeled distinctly.

III-8.

ST. Finite-state, complete state behavior

CT. —

SD. Topological Markov shift or State sequence shift

AT. —

Defn: A *topological Markov shift* is the set of all bi-infinite state sequences of a graph – equivalently, a finite-state complete state behavior.

Fact: Every graph shift is a topological Markov shift, every topological Markov shift is an SFT, every SFT is a sofic shift, and every sofic shift is a shift space; these inclusions are all proper.

IV. Properties of state representation (state realization)

Sect.	System Theory	Coding Theory	Symbolic Dynamics	Automata Theory
IV-1	Point controllable state behavior	—	Irreducible graph	—
IV-2	Strongly point controllable state behavior	—	Primitive graph	—
IV-3	Deterministic evolution law	Instantaneously invertible trellis	Right resolving labeled graph	Deterministic finite automaton
IV-4	—	Trellis that is invertible with delay	Right closing (Lossless of finite order) labeled graph	—
IV-5	—	—	Lossless labeled graph	Non-ambiguous finite automaton
IV-6	Externally induced (observable) state space system	Invertible realization	Conjugacy-inducing labeled graph	Local finite automaton
IV-7	Past-induced state space system	—	Right resolving, conjugacy inducing labeled graph	Local DFA
	(cont'd)	(cont'd)	(cont'd)	(cont'd)

Sect.	System Theory	Coding Theory	Symbolic Dynamics	Automata Theory
IV-8	Past-induced canonical realization	—	Right Krieger cover (Future cover) (Shannon cover)	Minimal DFA
IV-9	—	—	Right Fischer cover	—
IV-10	—	Tom cover	—	—

IV-1.

ST. Point controllable state behavior

CT. —

SD. Irreducible graph

AT. —

Defn: A state behavior (T, X, \mathcal{B}_x) is *point controllable* if for each $u, u' \in X$, there is an $x \in \mathcal{B}_x$ and $M \geq 0$ such that $x_0 = u$ and $x_M = u'$.

A graph G is *irreducible* if, for each pair of states I, I' in G , there is a path in G from I to I' .

Fact: A topological Markov shift, viewed as a state behavior, is point controllable iff its underlying graph is irreducible.

IV-2.

ST. Strongly point controllable state behavior

CT. —

SD. Primitive graph

AT. —

Defn: A state behavior (T, X, \mathcal{B}_x) is *strongly point controllable* if $\exists M$ such that

$\forall u, u' \in X$, there is an $x \in \mathcal{B}_x$ such that $x_0 = u$ and $x_M = u'$.

A graph G is *primitive* if there is a positive integer M such that for each pair of states I, I' , there is a path in G from I to I' of length M .

A topological Markov shift, viewed as a state behavior, is strongly point controllable iff its underlying graph is primitive.

In symbolic dynamics, primitive graphs are sometimes called *aperiodic* since a graph is primitive iff it is irreducible and the gcd of its cycle lengths is 1.

Fact: A sofic shift is irreducible (resp., mixing) iff it can be presented by an irreducible (resp., primitive) graph.

IV-3.

ST. Deterministic evolution law

CT. Instantaneously invertible trellis

SD. Right resolving labeled graph

AT. Deterministic finite automaton (DFA)

Defn: An evolution law is *deterministic* if the initial state and label determine the terminal state – in other words, whenever $(x, w, x'), (x, w, x'') \in \partial$, then $x' = x''$.

The term ‘unifilar’ is used for a similar way of presenting functions of Markov chains (in source coding theory).

A *deterministic finite automaton* is a finite automaton with a single initial state and whose labeling is deterministic i.e., at each state, distinct outgoing edges have distinct labels.

IV-4.

ST. —

CT. Trellis that is invertible with delay

SD. LFO labeled graph or Right closing labeled graph

AT. —

Defn: A labeled graph is *lossless of finite order* (LFO) if $\exists D$ (the ‘delay’) such that whenever two paths of length $D + 1$ present the same sequence and have the same initial state, then they have the same initial edge.

Equivalently, an LFO labeled graph is a finite-alphabet finite-state evolution law

such that $\exists D$ satisfying: whenever

$$(x_0, w_0, x_1)(x_1, w_1, x_2), \dots, (x_{D-1}, w_{D-1}, x_D) \in \partial,$$

$$(x'_0, w_0, x'_1), (x'_1, w_1, x'_2), \dots, (x'_{D-1}, w_{D-1}, x'_D) \in \partial,$$

and

$$x_0 = x'_0,$$

then

$$x_1 = x'_1.$$

This is a notion of ‘determinism with delay.’ ‘LFO’ is a term adopted by constrained coding, while ‘right closing’ is the term usually used in symbolic dynamics (see also V-7).

IV-5.

ST. —

CT. —

SD. Lossless labeled graph or Labeled graph with no diamonds

AT. Non-ambiguous finite automaton

Defn: A labeled graph is *lossless* if whenever two paths present the same label sequence and have the same initial and terminal states, then the paths coincide.

Equivalently, a lossless labeled graph is a finite-alphabet finite-state evolution law such that whenever

$$(x_0, w_0, x_1), (x_1, w_1, x_2), \dots, (x_{n-1}, w_{n-1}, x_n) \in \partial,$$

$$(x'_0, w_0, x'_1), (x'_1, w_1, x'_2), \dots, (x'_{n-1}, w_{n-1}, x'_n) \in \partial$$

and

$$x_0 = x'_0, x_n = x'_n,$$

then

$$x_i = x'_i$$

for $i = 0, \dots, n$.

In a labeled graph, a *diamond* is a pair of paths with the same initial state, terminal state and label sequence – hence, a lossless labeled graph is the same as a

labeled graph with no diamonds. The term ‘lossless’ has been adopted by constrained coding, while the phrase ‘no diamonds’ is usually used in symbolic dynamics.

Fact: If a labeled graph is lossless, then each bi-infinite sequence is the label sequence of only finitely many bi-infinite paths. The converse is true provided that the underlying graph is irreducible.

Fact: Deterministic \Rightarrow Lossless of Finite Order \Rightarrow Lossless

IV-6.

ST. Externally Induced state space system or Observable state space system

CT. Invertible realization

SD. Conjugacy-inducing labeled graph

AT. Local finite automaton

Defn: A state space system (T, W, X, \mathcal{B}_s) is *externally induced* if the external behavior determines the state sequence – in other words, whenever $(w, x), (w, x') \in \mathcal{B}_s$, then $x_0 = x'_0$ (equivalently, $x = x'$ because of time invariance).

See V-9 for more on the term ‘conjugacy.’

Fact: When the state space system is induced by a finite-state evolution law, (as in much of coding theory, symbolic dynamics and automata theory), ‘externally induced’ is equivalent to the condition that for some n , knowledge of $w|_{[-n,n]}$ is sufficient to determine the state x_0 at time 0.

Fact: A shift space is an SFT iff it can be presented by a conjugacy-inducing labeled graph.

IV-7.

ST. Past-Induced state space system

CT. —

SD. Right resolving conjugacy-inducing labeled graph

AT. Local DFA

Defn: A state space system (T, W, X, \mathcal{B}_s) is *past-induced* if whenever $(w, x), (w', x') \in \mathcal{B}_s$ and $w|_{(-\infty,0]} = w'|_{(-\infty,0]}$, then $x_0 = x'_0$.

Similarly, there is the notion of Future-induced state system \approx Left resolving conjugacy-inducing labeled graph \approx Co-deterministic local automaton

IV-8.

ST. Past-Induced canonical realization

CT. —

SD. Right Krieger cover or Future cover or Shannon cover

AT. Minimal DFA

Defn: For a system (T, W, \mathcal{B}) and $w \in \mathcal{B}$, the *follower set* of w is defined

$$\mathcal{F}(w) = \mathcal{F}(w|_{(-\infty, 0]}) = \{u \in \mathcal{B}|_{(0, \infty)} : w|_{(-\infty, 0]}u|_{(0, \infty)} \in \mathcal{B}\}$$

Say that $w, w' \in \mathcal{B}$ are *past equivalent* if $\mathcal{F}(w) = \mathcal{F}(w')$. Let X denote the set of equivalence classes $[w]$ of this equivalence relation. The *Past-Induced Canonical Realization* of (T, W, \mathcal{B}) is the state space system $\Sigma_s^\approx = (T, W, X, \mathcal{B}_s)$ where

$$\mathcal{B}_s = \{(w, x) \in W^T \times X^T : x_i = [\sigma_i(w)]\}.$$

In symbolic dynamics, for a sofic shift, the completion of the past-induced canonical realization is called the *Right Krieger cover* (sometimes, called the *future cover*). The *Shannon cover* is the same as the Right Krieger cover except that it is usually defined by follower sets of blocks rather than follower sets of left semi-infinite sequences.

The *minimal DFA* of a regular language \mathcal{L} is the smallest (in terms of number of states) deterministic finite automaton that presents \mathcal{L} .

Fact: The minimal DFA of a regular language is unique. When the regular language comes from a sofic shift, the right Krieger cover coincides with a particular subgraph of the minimal DFA.

Fact: When (T, W, \mathcal{B}) is an SFT, then Σ_s^\approx is automatically complete and is therefore the Right Krieger cover.

Similarly, there is the notion of Future-Induced Canonical Realization \approx Left Krieger Cover.

IV-9.

ST. —

CT. —

SD. Right Fischer cover

AT. —

Defn: For an irreducible sofic shift S , the *right Fischer cover* is the minimal (in terms of number of states) right resolving labeled graph which presents S .

Fact: The right Fischer cover is unique, and it is a particular subgraph of the right Krieger cover.

The uniqueness of the right Fischer cover is closely related to the uniqueness of the minimal DFA. But neither result implies the other.

IV-10.

ST. —

CT. Tom cover

SD. —

AT. —

Defn: An information theorist at Stanford.

V. Concepts of input/output

Sect.	System Theory	Coding Theory	Symbolic Dynamics	Automata Theory
V-1	Input/Output system	—	—	—
V-2	I/O system with output observable from input	—	—	—
V-3	Non-anticipating I/O system	Encoder	—	—
V-4	—	Feedbackfree encoder	Sliding block code (Continuous, shift-commuting map)	—
V-5	—	Causal encoder	—	—
V-6	—	—	Right resolving sliding block code	—
V-7	—	—	Right closing sliding block code	—
	(cont'd)	(cont'd)	(cont'd)	(cont'd)

Sect.	System Theory	Coding Theory	Symbolic Dynamics	Automata Theory
V-8	—	—	Finite-to-one sliding block code	—
V-9	—	—	(Topological) Conjugacy	—

V-1.

ST. Input/output system

CT. —

SD. —

AT. —

Defn: An *input/output system* (T, U, Y, \mathcal{B}) is a system $(T, U \times Y, \mathcal{B})$, with corresponding *input system* $(T, U, P_U \mathcal{B})$ and corresponding *output system* $(T, Y, P_Y \mathcal{B})$ satisfying the following:

1. The input system is memoryless

and

2. The input sequence and the past of the output sequence determine the future of the output sequence; more precisely, whenever $(u, y), (u, y') \in \mathcal{B}$ and $y|_{(-\infty, 0]} = y'|_{(-\infty, 0]}$, then $y = y'$.

V-2.

ST. Input/output system with output observable from input

CT. —

SD. —

AT. —

Defn: An input/output system (T, U, Y, \mathcal{B}) has *output observable from input* if for each $u \in P_U \mathcal{B}$, there is a unique y such that $(u, y) \in \mathcal{B}$.

V-3.

ST. Non-anticipating input/output system

CT. Encoder

SD. —

AT. —

Defn: A *non-anticipating input/output system* is an input/output system (T, U, Y, \mathcal{B}) such that whenever $(u, y) \in \mathcal{B}$, $u' \in P_U \mathcal{B}$, and $u'|_{(-\infty, 0]} = u|_{(-\infty, 0]}$, there is a (unique) y' such that $(u', y') \in \mathcal{B}$ and $y'|_{(-\infty, 0]} = y|_{(-\infty, 0]}$.

V-4.

ST. —

CT. Feedbackfree encoder

SD. Sliding block code or Continuous, shift-commuting map

AT. —

Defn: Let X and Y be shift spaces (recall that a shift space is a finite-alphabet complete time-invariant system). A *sliding block code* $\phi : X \rightarrow Y$ is a map which can be expressed as follows: there are integers m, a (*memory, anticipation*) and a map Φ from $(m + a + 1)$ -blocks of X to 1-blocks of Y such that $\phi(x)_i = \Phi(x|_{[i-m, i+a]})$. We write $\phi = \Phi_\infty$.

In coding theory, the term ‘feedbackfree encoder’ is used instead of the term ‘sliding block code.’

Fact: A map from one shift space to another is a sliding block code iff it is continuous and commutes with the shift map σ .

Note that for a labeled graph (G, \mathcal{L}) , the map \mathcal{L}_∞ from bi-infinite paths in G to bi-infinite label sequences is a sliding block code with $m = a = 0$.

V-5.

ST. —

CT. Causal encoder

SD. —

AT. —

Defn: A feedbackfree encoder is *causal* if the anticipation $a = 0$.

Of course, any feedbackfree encoder can be shifted so that it becomes causal. But usually feedbackfree encoders are assumed to be causal anyway.

V-6.
 ST. —
 CT. —
 SD. Right resolving sliding block code
 AT. —

Defn: A sliding block code $\phi : X \rightarrow Y$ is *right resolving* if $m = a = 0$ and $\Phi(x_0x_1) = \Phi(x_0x'_1)$ implies $x_1 = x'_1$.

V-7.
 ST. —
 CT. —
 SD. Right closing sliding block code
 AT. —

Defn: A sliding block code $\phi : X \rightarrow Y$ is *right closing* if whenever $x, x' \in X$, $x|_{(-\infty, 0]} = x'|_{(-\infty, 0]}$, and $\phi(x) = \phi(x')$, then $x = x'$.

V-8.
 ST. —
 CT. —
 SD. Finite-to-one sliding block code
 AT. —

Defn: A sliding block code is *finite-to-one* if each point in the image has only finitely many preimages.

The terms right resolving, right closing, and finite-to-one for sliding block codes correspond to the terms right resolving, right closing, and lossless for irreducible labeled graphs (i.e., a labeled graph (G, \mathcal{L}) is right resolving (resp., right closing, lossless) iff the sliding block code \mathcal{L}_∞ is right resolving (resp., right closing, finite-to-one).

Fact: For a sliding block code, $\phi : X \rightarrow Y$, the behavior $\{(x, \phi(x)) : x \in X\}$

defines an input/output system with output observable from input iff

- (1) The domain of ϕ is a full shift
- and
- (2) ϕ is right closing.

V-9.

ST. —

CT. —

SD. Conjugacy or Topological conjugacy

AT. —

Defn: A *conjugacy* is a sliding block code which is 1-1 and onto.

Fact: Up to conjugacy, the concepts Shift of finite type, Graph shift, and Topological Markov shift all coincide.

We also have the related notions:

Defn: A *factor map* or *factoring* is a sliding block code which is onto.

Defn: An *imbedding* is a sliding block code which is 1-1.

Fact: A sliding block code is an imbedding iff $\exists n$ such that $\phi(x)|_{[-n,n]} = \phi(x')|_{[-n,n]} \Rightarrow x_0 = x'_0$.

Note on minimal realizations: A realization of a system (T, W, \mathcal{B}) is a state space system (T, W, X, \mathcal{B}_s) whose external behavior is \mathcal{B} . A realization (T, W, X, \mathcal{B}_s) of \mathcal{B} is *minimal* if no other realization of \mathcal{B} can be obtained by ‘collapsing’ \mathcal{B}_s (via merging states in a way that can be made precise). Now, it is a fact that a system has a unique minimal realization iff its past-induced and future-induced canonical realizations coincide.

There is the following analogous result in symbolic dynamics. For an irreducible sofic shift Y , an irreducible SFT presentation is an SFT X and a factor map $\phi : X \rightarrow Y$. An irreducible SFT presentation of Y is minimal if no other irreducible SFT presentation of Y can be obtained by ‘collapsing’ ϕ (via sliding block codes in a way that can be made precise). It is a fact that an irreducible sofic shift has a unique minimal irreducible SFT presentation iff its Right and Left Fischer covers coincide. Such a sofic shift is called ‘almost of finite type.’

VI. Concepts of input/output machine

Sect.	System Theory	Coding Theory	Symbolic Dynamics	Automata Theory
VI-1	Input/State/ Output system	State space encoder	—	—
VI-2	I/S/O evolution law	Encoder trellis section	Finite-state code (encoder)	Transducer
VI-3	—	Encoder with finite-memory decoder	Sliding block decodable finite-state code	—
VI-4	—	Noncatastrophic encoder	Noncatastrophic finite-state code	—

VI-1.

ST. Input/State/Output system (abbr., I/S/O system)

CT. State space encoder

SD. —

AT. —

Defn: An *Input/State/Output System* is a state space system $(T, U \times Y, X, \mathcal{B})$ (with corresponding *input system* $(T, U, P_U \mathcal{B})$ and corresponding *output system* $(T, Y, P_Y \mathcal{B})$) such that

(1) the input system is memoryless

and

(2) the current state, together with the future of the input sequence, determine the future state sequence and future output sequence; more precisely, whenever $a \in P_X \mathcal{B}|_0$

and $b|_{[0,\infty)} \in P_U \mathcal{B}|_{[0,\infty)}$, there is a unique $(u, y, x)|_{[0,\infty)} \in \mathcal{B}|_{[0,\infty)}$ such that $x_0 = a$ and $u|_{[0,\infty)} = b|_{[0,\infty)}$.

This is equivalent to the condition that both systems $(T, U, Y \times X, \mathcal{B})$ and $(T, U, X, P_{U \times X} \mathcal{B})$ are nonanticipating input/output systems.

VI-2.

ST. Input/State/Output (I/S/O) evolution law

CT. Encoder trellis section

SD. Finite-state code or Finite-state encoder

AT. Transducer

Defn: An *Input/State/Output Evolution Law* is an evolution law $(T, U \times Y, X, \partial)$ for which there are functions $f : X \times U \rightarrow X$ (the ‘next state function’) and $g : X \times U \rightarrow Y$ (the ‘output function’) such that $(x, (u, y), x') \in \partial$ iff

$$x' = f(x, u) \tag{1}$$

and

$$y = g(x, u) \tag{2}$$

The corresponding *input evolution law* to an I/S/O evolution law is (T, U, X, ∂_U) with next state relation $(x, u, x') \in \partial_U$ iff $\exists y$ such that $(x, (u, y), x') \in \partial$. The corresponding *output evolution law* (T, Y, X, ∂_Y) is the evolution law with next state relation $(x, y, x') \in \partial_Y$ iff $\exists u$ such that relation $(x, (u, y), x') \in \partial$.

An *encoder trellis section* is the same as an I/S/O evolution law except that the state space is usually finite. An encoder trellis section can be viewed as a labeled graph where the label of an edge $(x, (u, y), x')$ is (u, y) ; the corresponding input (resp. output) evolution law may be regarded as an *input labeling* $\mathcal{I}(x, (u, y), x') = u$ (resp., an *output labeling* $\mathcal{O}(x, (u, y), x') = y$); according to (1) and (2) above, the input labeling is deterministic, and so given an initial state the future input sequence determines the future output sequence, symbol-by-symbol.

The term *finite-state code* (a term from constrained coding) is used to mean an encoder trellis section for which the output labeling is right closing. So, given an initial state, the input sequence can be recovered from the output sequence, with finite delay. Sometimes, the term *finite-state encoder* is used instead.

The term *transducer* is usually used to mean a labeled graph whose labels are pairs of symbols, sometimes with additional conditions imposed on the labelings.

Fact: An I/S/O system is complete iff it is represented by an I/S/O evolution law.

VI-3.

ST. —

CT. Encoder with finite-memory decoder

SD. Sliding-block-decodable finite-state code

AT. —

Defn: A finite-state code is *sliding block decodable* if the input can be recovered from the output via a sliding block code; more precisely, there is a sliding block code $\phi : Y^\infty \rightarrow U^\infty$ such that $\phi \circ \mathcal{O}_\infty = \mathcal{I}_\infty$ (recall that $\mathcal{I}_\infty, \mathcal{O}_\infty$ are the sliding block codes generated by the input, output labelings \mathcal{I}, \mathcal{O}).

VI-4.

ST. —

CT. Noncatastrophic encoder

SD. Noncatastrophic finite-state code

AT. —

Defn: A finite-state code is *noncatastrophic* if for any pair of output sequences that differ in finitely many places, any corresponding pair of input sequences must also differ in finitely many places; more precisely, whenever z, z' are bi-infinite paths in the underlying graph of the finite-state code and $\mathcal{O}_\infty(z)$ differs from $\mathcal{O}_\infty(z')$ in only finitely many places, then $\mathcal{I}_\infty(z)$ differs from $\mathcal{I}_\infty(z')$ in only finitely many places.

When the underlying graph is primitive, this definition agrees with the usual definition in coding theory. Otherwise, a slight modification is needed.

Fact: Sliding-block decodability \Rightarrow noncatastrophic, but the converse is false.

VII. Concepts of linearity

Sect.	System Theory	Coding Theory	Symbolic Dynamics	Automata Theory
VII-1	Linear System	Convolutional code	Linear shift space	—
VII-2	Linear state-space system	Linear realization	—	—
VII-3	Linear evolution law	Linear trellis section	—	—
VII-4	Linear I/S/O system	Convolutional encoder	Linear finite-state code	—
VII-5	—	Polynomial convolutional encoder	Linear sliding block code	—
VII-6	Group system	Group code	Group shift	—
VII-7	Group state-space system	Group realization	—	—
VII-8	Group evolution law	Group trellis section	—	—
VII-9	—	Orbit code, Geometrically uniform code,	Homogeneous shift	—

VII-1.

ST. Linear system

CT. Convolutional code

SD. Linear shift space

AT. —

Defn: A *linear system* is a linear subspace of $V^{\mathbb{Z}}$, where V is a vector space over a field F .

The term ‘linear system’ usually assumes F is the field of real numbers or the field of complex numbers.

The term *convolutional code* usually assumes F is a finite field.

Of course, the term *linear shift space* assumes ‘shift space.’

VII-2.

ST. Linear state space system

CT. Linear realization

SD. —

AT. —

Defn: A *linear state space system* (T, W, X, \mathcal{B}_s) is a state space system in which W and X are vector spaces and the behavior \mathcal{B}_s is a linear subspace of $W^T \times X^T$.

VII-3.

ST. Linear evolution law

CT. Linear trellis section

SD. —

AT. —

Defn: A *linear evolution law* is an evolution law in which the next state relation is a linear subspace.

Fact: The external behavior of a linear evolution law with finite-dimensional state space is always complete.

VII-4.

ST. Linear input/state/output system

CT. Convolutional encoder

SD. Linear finite-state code

AT. —

Defn: An I/S/O system $(T, U \times Y, X, \mathcal{B})$ is *linear* if U, Y and X are vector spaces and \mathcal{B} is a linear subspace of $(U \times Y \times X)^T$.

When U, Y , and X are vector spaces over a finite field, a linear I/S/O system is called a *convolutional encoder*. Often, this refers to the induced mapping which maps input to output.

Fact: For a finite-state convolutional encoder, Sliding block decodability \Leftrightarrow Non-catastrophic.

VII-5.

ST. —

CT. Polynomial convolutional encoder

SD. Linear sliding-block code

AT. —

Defn: Let F be a field, and k, n be positive integers. A *polynomial convolutional encoder* is a causal linear sliding block encoder from the full F^k -shift to the full F^n -shift (viewing the domain and range as vector spaces over F).

Fact: A shift space is the image of a linear sliding-block code iff it is a linear subspace and irreducible.

VII-6.

ST. Group system

CT. Group code

SD. Group shift

AT. —

Defn: Let \mathcal{A} be a group. Then $\mathcal{A}^{\mathbb{Z}}$ is a group with respect to the coordinatewise group structure. A *group code* (over \mathcal{A}) is a subgroup of $\mathcal{A}^{\mathbb{Z}}$. A group code which is also a shift space is called a *group shift*.

Fact: Every irreducible group shift is conjugate to a full shift.

VII-7.

ST. Group state space system

CT. Group realization

SD. —

AT. —

Defn: A *group state space system* (T, W, X, \mathcal{B}_s) is a state space system in which W and X are groups and the behavior \mathcal{B}_s is a subgroup of $W^T \times X^T$.

VII-8.

ST. Group evolution law

CT. Group trellis section

SD. —

AT. —

Defn: A *group evolution law* is an evolution law in which the next state relation is a subgroup.

Fact: The external behavior of a group evolution law whose state space satisfies the Descending Chain Condition is always complete.

VII-9.

ST. —

CT. Orbit code

SD. —

AT. —

Defn: Let \mathcal{A} be a group acting on a set S , and let $s \in S$. An *orbit code* is a code of the form

$$Y \equiv \{ \dots x_{-1}(s)x_0(s)x_1(s) \dots : x = \dots x_{-1}x_0x_1 \dots \in C \}$$

where C is a group code over \mathcal{A} .

When S is a metric space and \mathcal{A} is a group of isometries, an orbit code is called a *geometrically uniform code*.

A *homogeneous shift* is a shift space on which a group shift acts (coordinatewise) transitively.

Fact: Any mixing homogeneous shift is an orbit code.

Note on convolutional codes: the core meaning of “convolutional” is “linear time-invariant,” even though this meaning does not directly convey the notion of convolution. The alphabet is usually finite with algebraic structure: e.g., a finite field or a finite-dimensional vector space over a finite field. Coding theory has not traditionally been concerned with completeness; often a convolutional code is defined as the output of a linear time-invariant encoder where the input ranges over all semi-infinite input sequences (formal Laurent series), and such a code is not complete. More recent work has focused on polynomial encoders in which case inputs can range over all bi-infinite sequences, and the code is complete.

VIII. Other graph concepts

Sect.	System Theory	Coding Theory	Symbolic Dynamics	Automata Theory
VIII-1	—	—	Adjacency matrix	—
VIII-2	—	Parallel transitions	Multiple edges	—
VIII-3	—	—	Higher edge graph	—
VIII-4	—	Trellis section of length N	Higher power graph	—
VIII-5	—	Shift register graph	Higher edge graph of 1-state graph	DeBruijn graph

Recall the definitions of graph and labeled graph from III-4.

VIII-1.

ST. —

CT. —

SD. Adjacency matrix

AT. —

Defn: The *adjacency matrix* of a graph G is the square matrix A , indexed by the vertices of G , and defined by:

$$A_{IJ} \equiv \text{number of edges from } I \text{ to } J.$$

Concepts of irreducibility and primitivity of graphs are usually expressed in terms of adjacency matrices.

VIII-2.

ST. —

CT. Parallel transitions

SD. Multiple edges

AT. —

Defn: A set of *parallel transitions* is a set of edges with the same initial and terminal state.

VIII-3.

ST. —

CT. —

SD. Higher edge graph

AT. —

Defn: The *N-th Higher edge graph* of a graph G is the graph $G^{[N]}$ with vertices consisting of all paths of length $N - 1$ in G and a single edge from $e_1 \dots e_{N-1}$ to $f_1 \dots f_{N-1}$ iff $e_2 \dots e_{N-1} = f_1 \dots f_{N-2}$.

VIII-4.

ST. —

CT. Trellis section of length N

SD. Higher power graph

AT. —

Defn: The *N-th Higher power graph* of a graph G is the graph G^N with the same vertices as G , and an edge from I to J for each path of length N in G from I to J .

VIII-5.

ST. —

CT. Shift register graph

SD. Higher edge graph of 1-state graph

AT. DeBruijn graph

Defn: Higher edge graph of a 1-state graph

Fact: The underlying graph of a group code which is also a graph shift is a product of DeBruijn graphs.

IX. Entropy

IX-1.

ST. —

CT. —

SD. Entropy or Capacity

AT. —

Defn: The *entropy* of a shift space X (viewed as a system $(\mathcal{Z}, W, \mathcal{B})$) is

$$h(X) = \lim_{n \rightarrow \infty} \frac{\log(\#\mathcal{B}|_{[0,n]})}{n}.$$

Shannon effectively considered sofic shifts and called them discrete noiseless channels; he used the term *capacity* instead of entropy, since he reserved the latter for random processes. The constrained coding and magnetic recording communities have adopted the term capacity. But *entropy*, short for topological entropy, was adopted by symbolic dynamics.

Entropy (capacity) governs the maximal rate at which you can invertibly encode sequences from a full shift into sequences of a sofic shift.

INDEX OF TERMS

Adjacency matrix, 38
Capacity, 40
Causal encoder, 27
Closed, 6
Closure, 6
Code, 4
Code generated by a realization, 14
Code generated by a trellis, 16
Compact, 6
Complete, 5
Completion, 6
Conjugacy, 29
Conjugacy-inducing labeled graph, 22
Constrained system, 16
Continuous, shift-commuting map, 27
Controllable system, 12
Convolutional code, 34
Convolutional encoder, 34
DeBruijn Graph, 39
Deterministic evolution law, 20
Deterministic finite automaton (DFA), 20
Dynamical system, 4
Edge sequence shift, 16
Encoder trellis section, 31
Encoder with finite-memory decoder, 32
Encoder, 27
Entropy, 41
Evolution law, 14
External behavior, 14
External behavior of an evolution law, 16
Externally Induced state space system, 22
Factoring, 29
Factor map, 29
Feedbackfree encoder, 27
Finite automation, 15

Finite memory system, 10
 Finite-alphabet, 7
 Finite-constraint length code, 12
 Finite-state code, 31
 Finite-state, complete state behavior, 17
 Finite-state transition diagram (FSTD), 14
 Finite-to-one sliding block code, 28
 Fischer cover, 23
 Free system, 11
 Full shift, 11
 Future cover, 22
 Geometrically uniform code, 36
 Graph shift, 16
 Group code, 35
 Group evolution law, 36
 Group realization, 36
 Group shift, 35
 Group state space system, 36
 Group system, 35
 Group trellis section, 36
 Higher edge graph of 1-state graph, 39
 Higher edge graph, 39
 Higher power graph, 39
 Homogeneous shift, 36
 Imbedding, 29
 Input/output system, 26
 Input/output system with output observable from input, 26
 Input/State/Output system (abbr., I/S/O system), 30
 Input/State/Output (I/S/O) evolution law, 31
 Instantaneously invertible trellis, 20
 Invertible realization, 22
 Irreducible graph, 19
 Irreducible shift space, 12
 Krieger cover, 23
 L -complete, 9
 L -observable, 10
 L -step Shift of finite type (L -step SFT), 10

labeled finite directed graph, 14
Labeled graph with no diamonds, 21
Labeled graph, 14
Language, 4
LFO labeled graph, 20
Linear evolution law, 34
Linear finite-state code, 35
Linear input/state/output system, 34
Linear realization, 34
Linear shift space, 34
Linear sliding-block code, 35
Linear state space system, 34
Linear system, 34
Linear trellis section, 34
Local DFA, 22
Local finite automaton, 22
Lossless labeled graph, 21
Memoryless system, 11
Minimal DFA, 23
Mixing shift space, 12
Multiple edges, 39
Non catastrophic encoder, 32
Non catastrophic finite-state code, 32
Non-ambiguous finite automaton, 21
Non-anticipating input/output system, 27
Observable state space system, 22
Observable system, 10
Orbit code, 36
Parallel transition, 39
Past-Induced canonical realization, 23
Past-Induced state space system, 22
Point controllable state behavior, 19
Polynomial convolutional encoder, 35
Primitive graph, 19
Realization, 14
Regular language, 16
Right closing labeled graph, 20

Right closing sliding block code, 28
Right resolving conjugacy-inducing labeled graph, 22
Right resolving labeled graph, 20
Right resolving sliding block code, 28
Sequence space, 11
SFT, 10
Shannon cover, 22
Shift of Finite Type, 10
Shift of finite type, 10
Shift register graph, 39
Shift space, 7
Shift-invariant, 6
Sliding block code, 27
Sliding-block-decodable finite-state code, 32
Sofic shift, 16
Sofic system, 16
State behavior, 14
State code generated by a realization, 14
State sequence shift, 17
State space encoder, 30
State space system, 14
State space system of an evolution law, 15
Strongly complete, 10
Strongly controllable system, 12
Strongly observable system, 11
Strongly point controllable state behavior, 19
Time-invariant, 6
Tom cover, 24
Topological conjugacy, 29
Topological Markov shift, 17
Topologically transitive shift space, 12
Transducer, 31
Trellis, 15
Trellis diagram, 15
Trellis section, 14
Trellis that is invertible with delay, 20
Trim, 7