1995 Plenary Lecture

Symbolic Dynamics and Coding Applications

Brian Marcus, IBM Almaden Research Center
marcus@almaden.ibm.com

This article is a summary of the plenary lecture that I gave at the 1995 IEEE International Symposium on Information Theory. The lecture was intended to give a concrete and elementary introduction to the subject of symbolic dynamics. From my point of view, symbolic dynamics is the subject of "coding" from one "constrained set" of sequences to another. Of course, there are restrictions on what I mean by "coding" and "constrained set," and these are spelled out in section 4 below.

I will begin with a very concrete unsolved graph-theoretic problem, called the Road Problem, which first arose in symbolic dynamics. Then I will motivate symbolic dynamics from two very different points of view: (1) modeling of dynamical systems, which goes back to Hadamard almost 100 years ago, and (2) coding for data recording channels, which is of course much more recent. Finally, I will introduce some of the basic concepts and fundamental results in the subject and briefly indicate how they pertain to data recording.

For introductory reading on symbolic dynamics and its applications, requiring only a minimum of mathematical background, I unapologetically recommend the recent textbook, *An Introduction to Symbolic Dynamics and* Coding [9], by Doug Lind and myself. In addition there are some other very good expositions such as the monograph [4] and the article [14] (section IV). For expositions which focus more exclusively on data recording applications, the reader may consult [10] or [11]. All of these references contain extensive bibliographies.

1. The Road Problem

Suppose you are driving to Cleveland, but have lost your way. You call the auto club and ask for directions. The auto club operator gives you directions, and you now happily find yourself in Cleveland. But you never told the operator where you were calling from (Memphis, Los Angeles, Peoria?). How can this be? Well, this would be possible if the interstate highway system were a network of roads labeled in such a way that one particular sequence of instructions would direct drivers starting at any city to all arrive at a

particular city (Cleveland) at the same time. The Road Problem below is a formalization of this.

Let G be a *finite directed graph* (or simply *graph*) with *out-degree 2*; in other words, G is a finite network of states and directed edges connecting the states such that each state has exactly 2 outgoing edges (in the problem above, the states represent cities and the edges represent roads). We emphasize that a graph, by itself, has only states and edges, but no a priori labeling. A *road-coloring* of G is a labeling of edges such that at each state one outgoing edge is labeled 0 and the other is labeled 1. Figure 1 gives some examples of road-colorings.

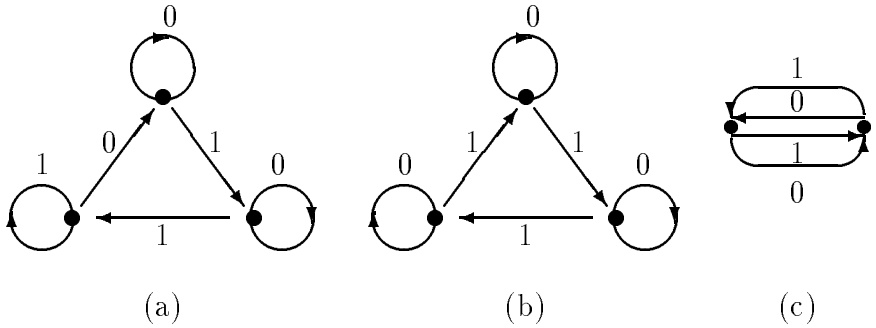

Figure 1: Some road-colorings

For a road-coloring, any binary word may be viewed as a sequence of instructions given to drivers starting at each of the states. A *synchronizing word* is a word that drives everybody to the same state. For instance, in Figure 1(a), the word 11 drives everybody to the state in the lower-left corner. But Figure 1(b) does not have a synchronizing word because whenever a driver takes a '0' road he stays where he is and whenever a driver takes a '1' road he rotates by 120 degrees. The road-coloring in Figure 1(c) is essentially the only road-coloring of its underlying graph, and there is no synchronizing word because drivers must always oscillate between the two states.

The Road Problem, which has been open since 1970, asserts that any graph with out-degree 2 and satisfying certain other assumptions has a road-coloring with a synchronizing word. Specifically:

Road Problem: If G is a graph with out-degree 2, is *strongly connected* (i.e., you can get from any state to any other state), and is *aperiodic* (i.e., the

greatest common divisor of cycle lengths is 1), then there is a road-coloring of G which has a synchronizing word.

Clearly some kind of connectivity assumption is required, and that is the role of strong connectivity. In the presence of that assumption, aperiodicity is necessary since otherwise there would be a "phase" introduced in the graph that would never allow a word to synchronize (as in Figure 1(c)). Note that the underlying graph in Figure 1(a) satisfies the assumptions of the Road Problem, and as we saw in that figure it does indeed have a road-coloring with a synchronizing word (although Figure 1(b) shows another road-coloring of the same graph that does not have a synchronizing word).

The Road Problem has been solved in several special cases, such as when the number of states is prime [7] or when the graph contains a non-self-intersecting cycle of prime length [13]. But the general problem remains unsolved. For more on the Road Problem, see [9](pp. 137-8 and p. 169).

In symbolic dynamics, the sequences considered are usually *bi-infinite*, i.e., extend from time $-\infty$ to time $+\infty$. Observe that any edge-labeling of a graph G defines a mapping ϕ from bi-infinite sequences of edges on G to bi-infinite sequences of labels (just by reading off the labels). As we will indicate in section 4, if the labeling happens to be a road-coloring with a synchronizing word, then ϕ is "almost one-to-one," and this is the significance of the Road Problem for symbolic dynamics.

2. Origins of symbolic dynamics: modeling of dynamical systems

Symbolic dynamics began as an effort to model dynamical systems using sequences of symbols. A *dynamical system* is a pair (X, T) where X is a set and T is a transformation from X to itself. For definiteness, we assume that T is invertible, although this is not a necessary restriction. Since T maps X to itself, we can iterate T: $T^2 = T \circ T$, $T^3 = T \circ T \circ T$, etc. The *orbit* of a point $x \in X$ is the sequence of points: $\ldots, T^{-2}(x), T^{-1}(x), x, T(x), T^2(x), \ldots$. In the theory of dynamical systems, one asks questions about orbits such as the following: are there periodic orbits (i.e., x such that $T^n(x) = x$ for some $n > 0$)? are there dense orbits (the orbit of x is dense if for any point y in X, $T^n(x)$ is "close" to y for some n)? how does the behavior of an orbit vary with x? how can we describe the collection of all orbits of the dynamical system? when is the dynamical system "chaotic"? See [6].

The subject of dynamical systems has its roots in Classical Mechanics; there, the set X is the set of all possible states of a system (e.g., the positions, momenta of all particles in a physical system), and the transformation T is the time evolution map, which maps the state of the system at one time to the state of the system one second later.

Symbolic dynamics provides a model for the orbits of a dynamical system (X, T) via a space of sequences. This is done by "quantizing" X into cells, associating symbols to the cells and representing points as bi-infinite sequences of symbols. For instance in Figure 2, X is a square, and T is some transformation of the square (the exact definition of T is unimportant). We have drawn a portion of the orbit of a point $x \in X$ for the dynamical system (X, T). We have also quantized X into two cells: the left half, called '0', and the right half, called '1'. Then the point x is represented by the bi-infinite sequence $s(x) = \ldots s_{-2}s_{-1}.s_0s_1s_2\ldots$ where s_n is the label of the cell to which $T^n(x)$ belongs (here, we use the decimal point to separate coordinates $s_i, i < 0$ from $s_i, i \geq 0$). So, for x as given in Figure 2, we see that

$$s(x) = \ldots 11.001\ldots$$

for instance $s_0 = 0$ because x belongs to the left half of the square, and $s_2 = 1$ because $T^2(x)$ belongs to the right half of the square. Now, if x is represented by the sequence $s(x)$, which sequence is $T(x)$ represented by? Well, since $T(x)$ has the same orbit as x, except that it is shifted one unit in time, we see that $s(T(x))$ is the shift (to the left) of $s(x)$:

$$s(T(x)) = \ldots 110.01\ldots$$

In other words, T is represented 'symbolically' as the shift transformation.

By representing all points of X as bi-infinite sequences, we obtain a *symbolic dynamical system* (Y, σ) where Y is a set of sequences (representing X) and σ is the shift transformation (representing T). The "symbolic" refers to the symbols, and the "dynamical" refers to the action of the shift transformation.

For this representation to be faithful, distinct points should be represented by distinct sequences, and this imposes extra conditions on how X is quantized into cells. Also, Y is typically a set of sequences constrained by certain rules, such as a certain symbol may only be followed by certain other symbols. For a specific example, see [9](sec. 6.5).

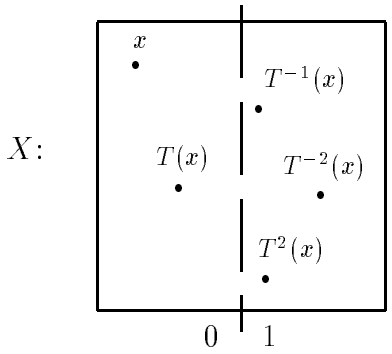

Figure 2: Representing points symbolically

In this way, one can use symbolic dynamics to study dynamical systems. Properties of orbits of the original dynamical system are reflected in properties of the resulting sequences. For instance, a point whose orbit is periodic becomes a periodic sequence, and a point whose orbit is dense becomes a "well-distributed" or "random" sequence. Beginning with Hadamard in 1898 and followed by Hedlund, Morse and others in the 1920's and 1930's, this method was used to prove the existence of periodic and other interesting motions in certain classical dynamical systems: this was done by finding interesting sequences satisfying the constraints defined by the corresponding symbolic dynamical system (see [6], [9](sec. 6.5), [12]).

In more recent years, symbolic dynamics has been used as a tool in classification problems for dynamical systems. Here, the problem of determining when one dynamical system is 'equivalent' to another becomes, via symbolic dynamics, a coding problem. Roughly speaking, two dynamical systems, (X_1, T_1) and (X_2, T_2), are *equivalent* if there is an invertible mapping from X_1 to X_2 which makes T_1 "look like" T_2. If the corresponding symbolic dynamical systems are denoted (Y_1, σ) and (Y_2, σ), then an equivalence between (X_1, T_1) and (X_2, T_2) becomes a time-invariant, invertible encoding from Y_1 to Y_2 (time-invariant because the shift transformation represents the dynamics). Thus, the classification problem in dynamical systems leads to a coding problem between constrained sets of sequences.

3. Coding for data recording channels

Another subject in which constrained coding problems arise is that of

data recording. In magnetic recording, within any given clock cell, a '1' is represented as a change in magnetic polarity, while a '0' is represented as an absence of such a change. Two successive 1's (separated by some number, $m \geq 0$, of 0's) are read as a voltage peak followed by a voltage trough (or vice versa). If the peak and trough occur too close together, we get *intersymbol interference*: the amplitudes of the peak and trough are degraded, and the positions at which they occur are distorted. In order to control intersymbol interference, it is desirable that 1's not be too close together, or equivalently that runs of 0's not be too short. On the other hand, for timing control, it is desirable that runs of 0's not be too long; this is because only 1's are observed: the length of a run of 0's is inferred by connection to a clock via a feedback loop, and a long run of 0's could cause the clock to drift more than one clock cell. This gives rise to *run-length-limited constraints,* $RLL(d,k)$, where runs of 0's are constrained to be bounded below by some number d and bounded above by some number k. Figure 3 shows how to represent $RLL(1,3)$ via a labeled graph.

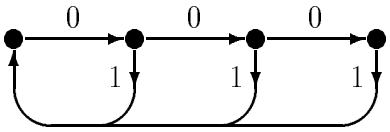

Figure 3: RLL(1,3)

Now in order to record completely arbitrary information, we need to build an encoder which encodes arbitrary binary sequences into sequences that satisfy a given constraint (such as RLL(1,3)). The *encoder* is a synchronous finite-state machine, as shown in Figure 4. It maps arbitrary data sequences, grouped into blocks of length p (called p-blocks), into constrained sequences, grouped into blocks of length q (called q-blocks). The encoded q-block is a function of the p-block as well as an internal state. When concatenated together, the sequence of encoded q-blocks must satisfy the given constraint. The ratio p/q is called the *rate* of the code.

The decoder is a *sliding block code*, which means that the decoded p-block depends only upon the local context of the received q-block – that is, decoding

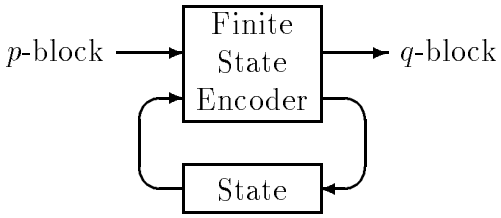

Figure 4: Encoder

is accomplished by applying a time-invariant function to a window consisting of a bounded amount of memory and/or anticipation, but otherwise is state-independent (see Figure 5, which depicts the situation where the memory is 1 and the anticipation is 2). Note that whenever the window of the decoder passes beyond a raw channel error, that error cannot possibly affect future decoding; thus, sliding block decoders control error propagation.

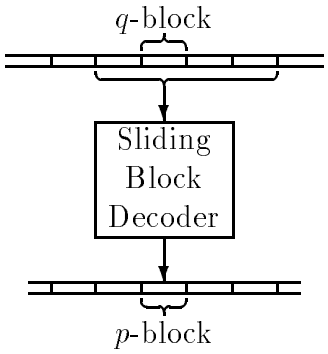

Figure 5: Sliding block decoder

Symbolic dynamics has played an important role in providing a framework, called the state-splitting algorithm, for constructing such codes as well as for establishing bounds on various figures of merit for such codes. We say a bit more about this in section 5 below. Detailed expositions of the algorithm are given in [9](chapter 5), [10], and [11].

4. Basic concepts and problems of symbolic dynamics

Symbolic dynamics focuses on sequence spaces and code mappings that

can be described in a very concrete way. The sequence spaces are typically of the following type: a *sofic shift* is a set of bi–infinite sequences obtained from a (finite) labeled graph (although more general sequence spaces are often considered). We usually assume that the underlying graphs are strongly connected and aperiodic. The term "sofic" comes from the Hebrew "sofi" which means finite, and the term "shift" reflects the origins of the subject as modeling dynamical systems. Sofic shifts are sometimes called *constrained systems*, but this term is usually reserved for the set of finite sequences obtained from a labeled graph.

Sofic shifts arise naturally in information and coding theory. In Figure 3, we presented the run-length-limited system, RLL $(1,3)$, as a sofic shift. In most textbooks, convolutional codes are presented as sofic shifts, although there the edges are labeled by n-tuples of binary symbols rather than simply by binary symbols.

The code mappings in symbolic dynamics are the sliding block codes, as described in section 3 and depicted in Figure 5. There we discussed sliding block codes in the context of decoders, i.e., as inverses to finite-state encoders. But a sliding block code is simply a time-invariant code mapping which depends only on a bounded amount of memory and anticipation; it need not have anything to do with a finite-state encoder.

A sliding block code which is invertible is called a *conjugacy*, and two sofic shifts are said to be *conjugate* if there is a conjugacy from one to the other (it turns out that the inverse of a conjugacy is again a sliding block code and hence also a conjugacy). Conjugacies are important because they can be used as equivalences between dynamical systems (in the sense of the discussion at the end of section 2) and also as encoders (in the sense of section 3).

A central problem in symbolic dynamics is: how can you tell when two sofic shifts are conjugate? This problem has remained unsolved for over 25 years. Nevertheless in the early '70's, R.F. Williams [16] showed that any conjugacy can be decomposed into special conjugacies obtained by simple operations on graphs, known as state-splitting and state-amalgamation. This result did not solve the conjugacy problem because it gives no bound on the number of splittings/amalgamations required to pass from one sofic shift to another via a conjugacy. However, it did give a good method of constructing conjugacies (namely, via state-splitting) as well as a strong and delicate necessary condition for conjugacy.

It is natural to wonder if the capacity, in the sense of Shannon, has any bearing on the conjugacy problem; here, the capacity of a sofic shift, S, is defined:

$$C(S) = \lim_{m \to \infty} \frac{\log_2 (\text{ number of allowed } m\text{-blocks in } S)}{m}.$$

It is not hard to see that whenever two sofic shifts are conjugate, they have the same capacity, but the converse is far from true. However, capacity does turn out to completely capture a weaker notion of conjugacy, described below.

An *almost 1-1 sliding block code* is a sliding block code that is invertible on "typical" sequences. Here, typicality can be defined in any one of several equivalent ways, such as a set of probability one with respect to any "decent" probability measure. One particular example of a typical set is the set of all bi-infinite binary sequences x that contain a particular word w infinitely often to the left (i.e., w occurs infinitely often in $\ldots x_{-3}x_{-2}x_{-1}$). Recall that any labeling $\mathcal{L}$ of a graph G, presenting a sofic shift S, defines a mapping ϕ from bi-infinite sequences of edges on G to sequences in S. The mapping ϕ can be regarded as a sliding block code (with no memory and no anticipation). Now, if G is a graph for which the Road Problem is solved (i.e., G has a road-coloring $\mathcal{L}$ with a synchronizing word w), then the corresponding mapping ϕ is an almost 1-1 sliding block code because every bi-infinite binary sequence which contains w infinitely often to the left is the label of exactly one bi-infinite sequence of edges (to see this, use the synchronizing and road-coloring properties).

We say that sofic shifts S_1 and S_2 are *almost conjugate* if there is a third sofic shift S_3 and almost 1-1 sliding block codes $\phi_1 : S_3 \to S_1$ and $\phi_2 : S_3 \to S_2$. As with conjugacies, almost conjugacies can be used to define equivalences between dynamical systems and also as encoders (for the latter, the system S_3 together with the sliding block codes, ϕ_1 and ϕ_2, can be used to encode sequences in S_1 to those in S_2).

In the late '70's, R. Adler and I showed that two sofic shifts are almost conjugate if and only if they have the same capacity [2]. The proof is actually quite constructive; it involves a combination of state-splitting together with a solution to a variant of the Road Problem. For a detailed discussion of the conjugacy and almost conjugacy problems, we refer the reader to [9](chapters 7,8,9).

While we have focused on the conjugacy and almost conjugacy problems, there are other important coding problems in symbolic dynamics. One of these yields a striking application to linear algebra: namely, the characterization of the sets of non-zero eigenvalues of nonnegative matrices (see [5] or [9](section 11.2)).

5. Applications of Symbolic Dynamics to Coding

In the early '80's, Adler, Coppersmith and Hassner (ACH) [1] developed the state-splitting algorithm for encoding data into sofic shifts. They gave an explicit construction, using state-splittings and almost conjugacies, to show that whenever $C(S) \geq p/q$, there is a rate p/q finite-state encoder for S with sliding block decoder (as defined in section 3), provided that S has finite memory. Here, a sofic shift has *finite memory* if it can be presented by a labeled graph such that any sufficiently long label sequence is synchronizing; in other words, for some integer m all paths of length m that have the same label sequence must end at the same state (for example, it is not hard to show that run-length-limited systems have finite memory).

The ACH paper was the beginning of a more rigorous theory of constrained coding. Their result has been extended to infinite memory sofic shifts, and bounds have been established on such figures of merit as number of encoder states as well as *decoding window* (= memory + anticipation + 1) of the sliding block decoder (see [10], [11] and the references therein). While the state-splitting algorithm does construct codes with "relatively small" decoding windows, it is not yet understood how to construct codes with the smallest such windows. This is of substantial engineering interest since the smaller the decoding window, the smaller the error propagation. In this regard, we mention some promising recent work of J. Ashley [3] as well as work of H. Hollmann and K. Immink which combines state-splitting together with state-amalgamation as well as earlier look-ahead encoding ideas of P. Franaszek (for instance, see [8]).

Finally, we mention that symbolic dynamics provides a useful framework for studying codes with algebraic structure, such as convolutional codes and D. Forney's geometrically uniform codes; see [14], [15].

Acknowledgment: I would like to thank the following people for reading an earlier version of this article: Jonathan Ashley, Elza Erkip, Garud Iyengar, Erik Ordentlich, Ronny Roth, Brett Schein, and Mitchell Trott.

References

[1] R.L. Adler, D. Coppersmith and M. Hassner, "Algorithms for sliding block codes — an application of symbolic dynamics to information theory," *IEEE Transactions on Information Theory, IT-29 (1983), 5–22.*

[2] R. Adler and B. Marcus, "Topological entropy and equivalence of dynamical systems," AMS Memoirs, 219 (1979).

[3] J. Ashley, "A linear bound for sliding block decoder window size II," *IEEE Transactions on Information Theory, to appear.*

[4] M.-P. Béal, Codage Symbolique, Masson, Paris, 1993.

[5] M. Boyle and D. Handelman, "The spectra of nonnegative matrices via symbolic dynamics," *Annals of Mathematics, 133 (1991), 249–316.*

[6] R. Devaney, An Introduction to Chaotic Dynamical Systems, Addison-Wesley, 1987.

[7] J. Friedman, "On the road coloring problem," *Proc. AMS, 110 (1990), 1133–1135.*

[8] H. D. L. Hollmann, "On the construction of bounded-delay encodable codes for constrained systems," IEEE Transactions on Information Theory, IT-41 (1995), 1354–1378.

[9] D. Lind and B. Marcus, *An Introduction to Symbolic Dynamics and Coding, Cambridge University Press, 1995.*

[10] B. Marcus, R. Roth and P. Siegel, "Constrained systems and coding for recording channels," Handbook of Coding Theory (ed. R. Brualdi, C. Huffman, V. Pless), Elsevier Press, to appear.

[11] B. H. Marcus, P. H. Siegel and J. K. Wolf, "Finite-state modulation codes for data storage," *IEEE Journal on Selected Areas in Communications, 10 (1992), 5–37.*

[12] M. Morse and G. A. Hedlund, "Symbolic dynamics," American J. Math., 60 (1938), 815–866.

[13] G. L. O'Brien, "The road coloring problem," *Israel J. Math., 39(1981), 145–154.*

[14] E. J. Rossin, N.T. Sindhushayana, and C.D. Heegard, "Trellis group codes for the Gaussian channel," IEEE Transactions on Information Theory, IT-41 (1995), 1217–1245.

[15] N.T. Sindhushayana, B. Marcus and M. Trott, "Homogeneous shifts," *IMA J. Math. Control and Information, to appear.*

[16] R. F. Williams, "Classification of subshifts of finite type," Annals of Math. 98 (1973), 120–153; erratum, Annals of Math. 99 (1974), 380–381.

BIOGRAPHY:

Brian Marcus attended Claremont Mens College and received his B.A. from Pomona College in 1971. He received his M.A. and Ph.D. in mathematics from the University of California at Berkeley in 1972 and 1975, respectively. From 1975-1984 he was Assistant Professor and then Associate Professor of Mathematics at the University of North Carolina-Chapel Hill. Since 1984, he has been a Research Staff Member at the IBM Almaden Research Center. His current research interests include symbolic dynamics and coding for storage devices.

Dr. Marcus held the IBM Watson Postdoctoral Fellowship in 1976-7 and shared the Leonard G. Abraham Prize Paper Award of the IEEE Communications Society in 1993. He has held visiting positions at Stanford University, UC-Berkeley, UC-Santa Cruz and the University of Maryland. He is the author of several papers in ergodic theory and information theory as well as a textbook, *An introduction to Symbolic Dynamics* (Cambridge University Press, 1995).