Constrained Systems With Unconstrained Positions

Jorge Campello de Souza, *Member, IEEE*, Brian H. Marcus, *Fellow, IEEE*, Richard New, and Bruce A. Wilson, *Member, IEEE*

***Abstract*—We develop methods for analyzing and constructing combined modulation/error-correctiong codes (ECC codes), in particular codes that employ some form of reversed concatenation and whose ECC decoding scheme requires easy access to soft information (e.g., turbo codes, low-density parity-check (LDPC) codes or parity codes). We expand on earlier work of Immink and Wijngaarden and also of Fan, in which certain bit positions are reserved for ECC parity, in the sense that the bit values in these positions can be changed without violating the constraint. Earlier work has focused more on block codes for specific modulation constraints. While our treatment is completely general, we focus on finite-state codes for maximum transition run (MTR) constraints. We 1) obtain some improved constructions for MTR codes based on short block lengths, 2) specify an asymptotic lower bound for MTR constraints, which is tight in very special cases, for the maximal code rate achievable for an MTR code with a given density of unconstrained positions, and 3) show how to compute the capacity of the set of sequences that satisfy a completely arbitrary constraint with a specified set of bit positions unconstrained.**

***Index Terms*—Finite-state encoders, modulation codes, maximum transition run (MTR) codes, reversed concatenation, run-length limited (RLL) codes.**

I. Introduction

In recording systems and communication systems, data is encoded via an error-correction code (ECC), which enables correction of a certain number of channel errors. In many such systems, data is also encoded into a constrained system of sequences via a modulation code, which helps to match the coded sequences to the channel and thereby reduce the likelihood of error.

Prominent examples of constrained systems are the maximum transition run (MTR(j)) systems [17] defined by the requirement that the maximum run of consecutive 1's is j (see Fig. 1 with $j = 4$) and the well-known run-length limited (RLL(d, k)) systems. While we will restrict ourselves in this paper mainly to binary systems, the results carry over easily to constrained systems over any finite alphabet of symbols.

In principle, the data can be encoded with one encoder that imparts both error-correction and modulation properties. But in practice the combined properties are achieved by cascading an error-correction encoder with a modulation encoder. In *standard concatenation*, shown in Fig. 2, data is first encoded by an error-correction encoder and then encoded by a modulation

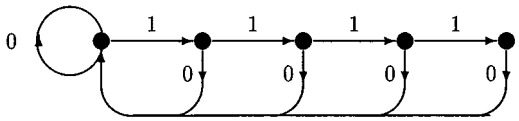

Fig. 1. MTR($j = 4$).

encoder. This is natural since otherwise the error-correction encoder might well destroy the modulation properties. However, this scheme has the disadvantage that the modulation decoder, which comes before the error-correction decoder, can propagate channel errors before they can be corrected. For this reason, a good deal of attention has recently focused on *reversed concatenation*, shown in Fig. 3, where the encoders are concatenated in the reversed order. Special arrangements must be made in order to ensure that the output of the error-correction encoder satisfies the modulation constraints. Typically, this is done by insisting that this encoder be systematic and then re-encoding the parity information using a second modulation encoder, whose corresponding decoder is designed to limit error propagation; the output of this second modulation encoder is then appended to the modulation-encoded data stream (typically, a few merging bits may need to be inserted in between the two streams in order to ensure that the entire stream satisfies the constraint). In order to distinguish between the two modulation encoders, we call the first one the *data modulation encoder* and the second *parity modulation encoder*; of course, we also have the corresponding decoders. In this scheme, after passing through the channel the modulation-encoded data stream is split from the modulation-encoded parity stream, and the latter is then decoded via the parity modulation decoder before being passed on to an ECC decoder. In this way, many channel errors can be corrected before passing through the data modulation decoder, thereby mitigating the problem of error propagation. Moreover, if the data modulation encoder has high rate, then the overall scheme will still have high rate because the parity stream is relatively small.

Now, suppose that for decoding the ECC scheme requires soft information, say on the bit level. For instance, the ECC could be a turbo code, low-density parity check (LDPC) code [15], or a simple parity code [6]. In reversed concatenation, after passing through the channel, the data modulation encoded stream is passed directly to the ECC decoder, and so soft information for each bit in this stream becomes readily available. However, this is not necessarily so for parity bits; the information will need to be filtered through the parity modulation decoder. Procedures for generating soft information for parity bits have been described in [3], [7], and [18]. However, these procedures can be complicated and introduce inaccuracies. On the other hand, this becomes much simpler and more accurate if the

Manuscript received March 28, 2001.

The authors are with the IBM Research Division, Almaden Research Center K65/802, San Jose, CA 95120 USA (e-mail: campello@almaden.ibm.com; marcus@almaden.ibm.com; new@almaden.ibm.com; bawilson@almaden.ibm.com).

Communicated by R. M. Roth, Associate Editor for Coding Theory.

Publisher Item Identifier S 0018-9448(02)01940-5.

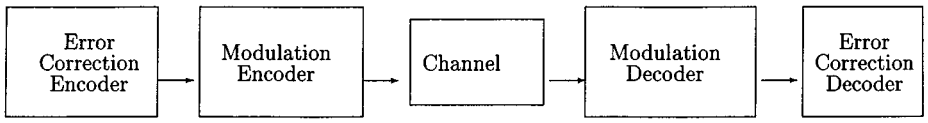

Fig. 2. Standard concatenation.

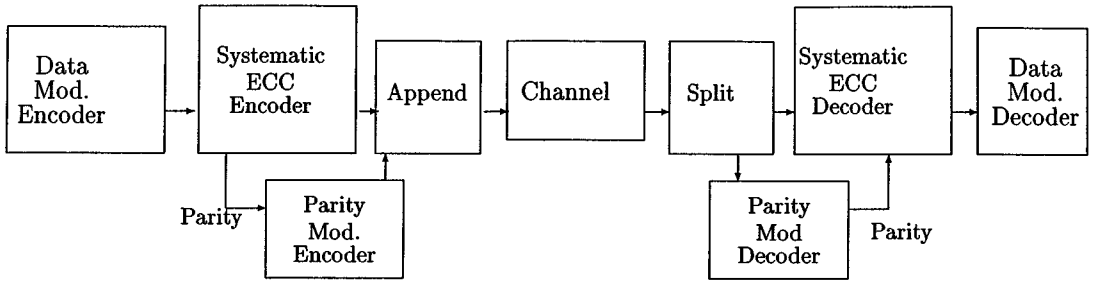

Fig. 3. Reversed concatenation.

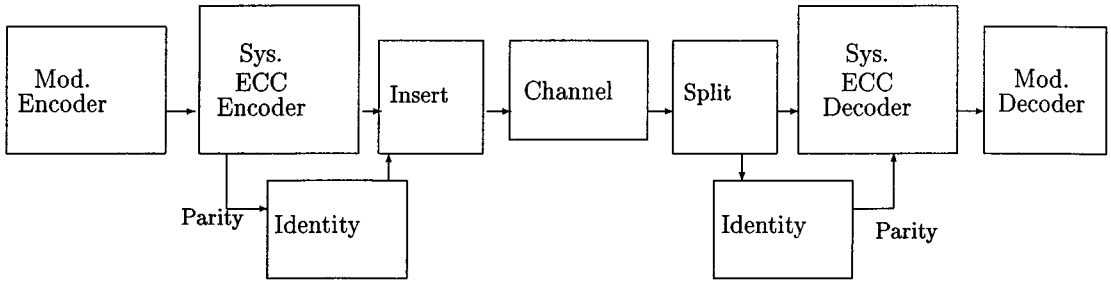

Fig. 4. Wijngaarden–Immink scheme.

parity modulation encoder is required to be systematic (this does incur an additional rate penalty, but applied only to the relatively small parity portion). For instance, for the MTR(j) constraint, one can use the rate $j : j + 1$ systematic parity modulation encoder which simply inserts a 0 after every j bits—see Fan and Cioffi [7], [8].

In a related approach [21] (see also [20] and [13, pp. 103–105]), Wijngaarden and Immink introduced a concatenation scheme, shown in Fig. 4, in which a modulation encoder produces constrained sequences such that certain bit positions are unconstrained in the sense that whenever the bit values in those positions are flipped (or not flipped) independently, the constraint is not violated. ECC parity information, obtained from a systematic ECC encoding of the data modulation encoded stream can, therefore, be inserted into the unconstrained positions without violating the constraint.

As a simple example, consider the length-5 block code [21] for the MTR $(j = 2)$ constraint

$$\{10101, 01101\}.$$

Note that all concatenations of these words satisfy the MTR($j = 2$) constraint; moreover, this still holds if the bit values in the third and fifth positions are flipped independently. We view this as a rate $1 : 5$ modulation code with two bit positions reserved for ECC parity.

Another way to look at this code is as follows. If we delete the third and fifth positions, then we obtain a length-3 block code

$$\{100, 010\}$$

all of whose concatenations satisfy the MTR($j = 1$) constraint. Inserting the two parity bits then weakens the constraint to MTR($j = 2$). So, the idea is that given a desired constraint S, we first construct a modulation code for a more restrictive constraint S' and then insert parity bits which result in sequences that satisfy S.

This Wijngaarden–Immink scheme is the subject of our paper. In fact, the title of our paper is a slight modification of a section title in their paper. Variants of this scheme have also been considered by [7], [8], as well as [2].

The focus in [21] was on block codes, in particular simple, combinatorial constructions of low complexity for very special constraints ($(0, k)$-RLL constraints and some $(0, G/I)$ constraints). In contrast, here we consider the more general class of finite-state codes for completely general constraints (although in the first few sections, we emphasize MTR constraints, which are equivalent to $(0, k)$-RLL constraints). Our intent is to combine the various approaches for easily providing soft information to an ECC decoder.

As mentioned in [21], this scheme can alternatively be used purely for modulation code construction (without regard to error

correction). For instance, the rate $1:5$ block modulation code with error-correction capability above can be viewed as a rate $3:5$ modulation code without error-correction capability. Such a code has the advantages that encoding is completely trivial on the unconstrained positions and channel errors on the unconstrained positions do not propagate at all.

Our paper is organized as follows. In Section II, we lay out the basic concepts regarding constrained systems with unconstrained positions. In Section III, we specialize to MTR constraints and show that for codes based on "short" block lengths, finite-state coding constructions can, in some situations, improve upon the performance of block coding constructions given in [21]. In Section IV, based on bit-stuffing we specify an asymptotic lower bound on code rate for MTR constraints with a given density of unconstrained positions. This bound is tight for $j = 1$ and $j = 2$, but is not tight for general j.

In Section V, we give a brief review of standard background on the notion of follower sets for constrained systems. In Section VI, given an arbitrary constrained system S and subset U of integers modulo some integer N, we use follower sets to construct a finite-state graph which presents the unique maximal subsystem of S such that any position $i \bmod N \in U$ is unconstrained. In principle, this enables us to compute the maximal possible rate of a code that satisfies a given constraint and is unconstrained in a specified set of positions. While the construction in this section is very general, it can be rather complicated. In Section VII, some simplifications are considered. Finally, in Section VIII, we show how to simplify the construction even further in the case of finite memory systems.

II. Background and Basic Definitions

A finite directed labeled graph G (or simply *graph*) is a finite collection of states and transitions, with each transition having an initial state, terminal state, and label; the notation $I \xrightarrow{a} J$ signifies a transition from state I to state J with label a. A collection of transitions is said to be *parallel* if they all have the same initial state and all have the same terminal state. A graph is *deterministic* if for any given state and any given symbol, there is at most one outgoing edge with the given symbol as label. A graph G has *finite memory* M if whenever any two paths in G of length M have the same label sequence, they end at the same state.

A binary *constrained system* S is a set of finite sequences, called *words*, with alphabet $\{0, 1\}$ defined as the set of label sequences obtained by traversing paths of a graph G. We say that the constrained system S is *presented* by G. Constrained systems that have a finite memory presentation are called *finite memory systems* or *systems of finite type*. Examples of such systems include the RLL and MTR constraints.

We will typically write a word as a sequence of symbols of length ℓ

$$x = x_0 \cdots x_{\ell-1}.$$

We use x^n to denote the concatenation of n copies of x. Note that the truncation of any word in S must necessarily belong to S. It is well known that any constrained system can be presented by a deterministic graph [14], [17].

Let S be a constrained system, N a positive integer, and $U \subseteq \{0, \ldots, N-1\}$ (the notation "U" is supposed to suggest "Unconstrained," and U will sometimes be called the *unconstrained set*). We say that a word x' is a *U-flip* of x if

$$x'_i = x_i \text{ whenever } i \bmod N \notin U.$$

In other words, x' is obtained from x by independently flipping (or not flipping) the bit values in positions

$$\{i\colon i \bmod N \in U\}.$$

The *(U, N)-unconstrained version of S*, denoted $S_{U,N}$, is the set of all sequences $x \in S$ such that

1) $x_i = 1$ for all $i \bmod N \in U$
2) all U-flips of x belong to S.

Note that in the unconstrained positions, the bit value is forced to be "1," but this was arbitrary: we could have just as well chosen "0" or made a random, but fixed, assignment of bit values in these positions. These positions are unconstrained in the sense that we can independently change the bit values without violating the constraint. In fact, if we augment $S_{U,N}$ by throwing in all possible such changes, we obtain the unique maximal subset of S which contains every U-flip of every element of S. The ratio $|U|/N$ is called the *parity insertion rate* (or simply *insertion rate*) because it represents the percentage of positions in which ECC parity information can be inserted without violating the constraint. Note that $S_{\emptyset,N} = S$ and $S_{\{0,\ldots,N-1\},N} = \emptyset$ unless S itself is unconstrained.

One more piece of notation: $\overline{S}_{U,N}$ denotes the set of sequences in $S_{U,N}$ of length exactly N.

If $0 \in U$, then the positions $0, N, 2N, \ldots$ are unconstrained. So, the truncation (say, by deleting the first symbol) of a word in $S_{U,N}$ need not belong to $S_{U,N}$, and so $S_{U,N}$ need not be a constrained system. But this is the only way in which it fails to be a constrained system. We could remedy this by throwing in truncations of words in $S_{U,N}$.

In Section VI, we will show that there is a graph $G_{U,N}$ with the following properties:

- $G_{U,N}$ is deterministic;
- $G_{U,N}$ has period N (i.e., the states are divided into N disjoint phases, $\{0, \ldots, N-1\}$, with transitions moving cyclically from one phase to the next);
- the transitions beginning in all phases of U have all outgoing edges labeled 1;
- $S_{U,N}$ is the set of words that can be generated starting in phase 0.

The special case of $G_{U,N}$ for MTR constraints is described in Section III.

The *capacity* of a constrained system is defined as

$$Cap(S) = \lim_{n\to\infty} (1/n) \log(N(n, S)) \tag{1}$$

where $N(n, S)$ is the number of words of length n in S (the log is $\log_2$). The capacity can be computed as the log of the

largest eigenvalue, $\lambda(A(G))$, of the adjacency matrix $A(G)$ of any deterministic presentation G of S [22]. For $S = \mathrm{MTR}(j)$, the adjacency matrix $A = A^{(j)}$ of the standard presentation of S is the $(j+1) \times (j+1)$ matrix whose superdiagonal and first column consist entirely of ones (with zeros elsewhere). For instance, for $j = 4$, we have

$$A = A^{(4)} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

This matrix is the adjacency matrix of the graph in Fig. 1.

It is well known that $Cap(S)$ is the maximal rate of an encoder from unconstrained sequences into sequences of S [19]. Of course, the encoded sequences are required to satisfy the constraint, not only within each codeword but across codeword boundaries; in particular, for a block code, the codewords must be freely concatenable without violating the constraint.

For finite-memory systems, decoding can be accomplished via a sliding-block decoder [1]. Even if the constraint S is of finite type, the system $S_{U,N}$ need not be of finite type—roughly because of the multiple phases. However, these systems $S_{U,N}$ do belong to a larger class of constrained systems for which sliding-block decodability is always achievable, but at the expense of considerable added complexity. Instead, our constructions of finite-state codes for these systems will be sliding-block decodable in a weaker sense: we will allow the decoding function to vary from phase to phase; that is, even if the same constrained sequence can be generated from more than one phase, we will allow the decoding to depend on the phase, implicitly assuming that the decoder has access to phase information (in most applications, this is a reasonable assumption). We remark that for finite-type S, the systems $S_{U,N}$ are natural examples of periodic finite type (PFT) systems of Moision and Siegel [23]

While $S_{U,N}$ is not literally a constrained system, we can define its capacity just as in (1), and this coincides with $\log(\lambda(A(G_{U,N})))$ (because if we throw in truncations of words in $S_{U,N}$, then we get an honest constrained system without changing the growth rate). Thus, capacity gives us the maximal possible rate of an encoder which produces sequences that satisfy the constraint S and allow the positions $i \bmod N \in U$ to be unconstrained.

When N is large, $G_{U,N}$ will necessarily have many states. However, an encoder for $S_{U,N}$ need not use all N phases. For instance, a rate $p : N$ encoder need use only one phase. In general, a rate $p : q$ encoder need use only

$$\frac{N}{\gcd(N, q)} \tag{2}$$

phases of $G_{U,N}$; each of these encoder phases can be viewed as a rate $p : q$ finite-state machine with initial states in one phase and terminal states in another phase [4]. As mentioned above, we allow the sliding-block decoding function to vary from phase to phase.

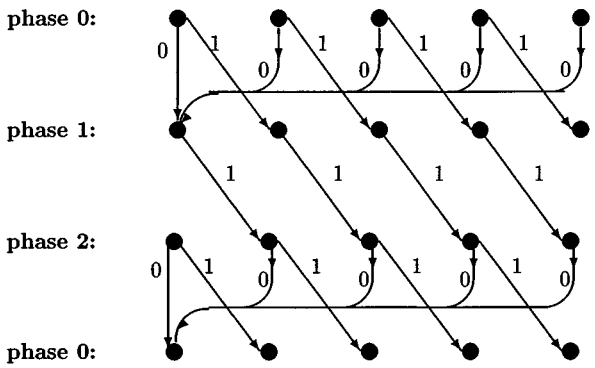

Fig. 5. $G_{U,N}$ for MTR($j = 4$), $N = 3$ and $U = \{1\}$.

III. Short Codes for MTR Constraints

For the MTR(j) constraint, $S_{U,N}$ is the set of sequences that can be generated from phase 0 in the graph $G_{U,N}$, described as follows.

- **States:** $j+1$ states in each of N phases
$$\{(\ell, i): 0 \le \ell \le N-1, 0 \le i \le j\}.$$
Here i represents the number of preceding consecutive 1's and ℓ represents the phase.
- **Transitions:** Beginning in any phase $\ell \notin U$, we have the transitions
$$(\ell, i) \xrightarrow{1} (\ell + 1 \bmod N, i+1), \qquad \text{if } i < j$$
$$(\ell, i) \xrightarrow{0} (\ell + 1 \bmod N, 0)$$
and beginning in any phase $\ell \in U$ we have the transitions
$$(\ell, i) \xrightarrow{1} (\ell + 1 \bmod N, i+1), \qquad \text{if } i < j.$$

For MTR($j = 4$), $N = 3$, and $U = \{1\}$, this is illustrated in Fig. 5 (with the phase 0 states repeated at both top and bottom).

In any phase $\ell \notin U$, the transitions mimic those of the standard presentation of MTR(j) (as shown in Fig. 1 for $j = 4$), but pass from phase ℓ to phase $\ell + 1$. In any phase $\ell \in U$, the next binary symbol is constrained to be a 1, but must allow for the possibility that the 1 can be flipped to a 0 without violating the constraint; thus, the transition outgoing from state (ℓ, i) must end at the "more severely constrained" of the two possibilities: $(\ell + 1, i + 1)$ or $(\ell + 1, 0)$, namely, $(\ell + 1, i + 1)$ (provided, of course, that $i < j$; otherwise, there is no transition).

Reference [21] gave many nice low-complexity constructions of block codes for RLL(0, j) constraints and therefore (by binary complementation) for MTR(j) constraints. But given the graph $G_{U,N}$, one can consider applying finite-state construction methods to yield finite-state codes [22]. It is then interesting to compare the resulting finite-state codes with the earlier block codes.

One way to do this is as follows. For a given codeword length q and a given number u, set $N = q$ and ask what is the maximum number m possible for a rate $m : q$ block code and finite-state code which satisfies the MTR(j) constraint and allows for u unconstrained positions in each block of length $N = q$.

For block codes, an application of a result of [11] shows that the maximum m is the log of the max of certain sums of entries

TABLE I
FOR MTR($j = 3$), MAXIMUM NUMBER m OF INFORMATION BITS FOR GIVEN LENGTH $q = N$ AND NUMBER u OF UNCONSTRAINED POSITIONS

q	u	m (*block*)	m (*finite state*)	*window*	*#states*
8	2	4	**5**	2	2
8	1	6	6	1	1
8	0	7	7	1	1
9	2	5	**6**	2	3
9	1	7	7	1	1
9	0	8	8	1	1
10	2	6	**7**	2	4
10	1	8	8	1	1
10	0	9	9	1	1
11	2	7	**8**	2	4
11	1	8	**9**	2	2
11	0	10	10	1	1
12	2	8	**9**	3	13
12	1	9	**10**	2	3
12	0	11	11	1	1
13	2	9	9	1	1
13	1	10	**11**	2	4
13	0	11	**12**	2	2

of powers of the adjacency matrix: for MTR($j = 3$), this turns out to be the log of the max of the four quantities

$$\hat{A}^q_{0,0}, \quad \hat{A}^q_{1,0} + \hat{A}^q_{1,1}, \quad \hat{A}^q_{2,0} + \hat{A}^q_{2,1} + \hat{A}^q_{2,2}$$
$$\hat{A}^q_{3,0} + \hat{A}^q_{3,1} + \hat{A}^q_{3,2} + \hat{A}^q_{3,3}$$

where $\hat{A}$ is the adjacency matrix of $G_{U,N}$ and the indices (i, i') of $\hat{A}^q$ refer to states $(0, i)$ and $(0, i')$ in $G_{U,N}$.

For finite-state codes, an application of [1] shows that the maximum m is $\lfloor \log(\lambda(\hat{A}^q)) \rfloor$.

Table I reveals that, for MTR($j = 3$) and some choices of $8 \le q \le 13$ and $0 \le u \le 2$, the number m can be increased if one allows a finite-state code rather than a block code (see the bold-faced entries in the table). The column labeled "window" shows the length, measured in number of q-blocks, of the decoder window for the finite-state code (of course, the decoder window for a block code is always 1). The column labeled "# states" shows the number of encoder states for the finite-state code (of course, a block code has only one state). The entries in these columns are determined by finding an approximate eigenvector and corresponding state splitting [22].

One can then evaluate tradeoffs between the block codes and finite-state codes. For instance, the rate $8 : 11$ block code with $u = 1$ can be compared against the rate $8 : 11$ finite-state code with $u = 2$. Here, the finite-state code has twice the error correction power of the block code, but the former has to cope with some error propagation. For bursty channels, the finite-state code would probably perform better.

One can also compare the rate $8 : 10$ block code against the rate $9 : 11$ finite-state code, each with $u = 1$. Here, the block code will have better error protection (one parity bit per 10 bits versus one parity bit per 11 bits) and will not have to cope with error propagation, but the finite-state code will have a higher rate. In a low-SNR regime, the block code may be superior, while in a high-SNR regime the finite-state code may be superior.

The entries in the table indicate that even the finite-state codes are not terribly complex. But in general they will not match the very low complexity of the Wjingaarden–Immink constructions [21].

TABLE II
RATE $8/9$ CODES FOR MTR($j = 4$)

scheme	*length*	*code rate*	*parity insertion rate*	*window (bits)*	*#states*
#1 (*block*)	18	8/9	$1/18 \approx .0556$	18	1
#2 (*block*)	45	8/9	$3/45 \approx .0667$	45	1
#3 (*finite state*)	9	8/9	$1/15 \approx .0667$	18	26
#4 (*finite state*)	9	8/9	$2/27 \approx .0741$	27	32

For the codes in Table I, the codeword length q coincides with the N in the definition of $S_{U,N}$, and so the parity insertion rate is always $u/N = u/q$. According to the discussion at the end of Section II, this has the advantage that only one of the N phases need be used in the construction of an encoder. On the other hand, in this case, we must have $m + u < q$, for otherwise we would have a rate $q : q$ encoder that satisfies the constraint. In particular, if $u \ge 1$, we can never have a rate $q - 1 : q$ code (indeed, this is consistent with the results in the table). On the other hand, if we allow the possibility of $N > q$ it is possible to construct codes with rate $q - 1 : q$ and $u \ge 1$.

For this purpose, we now consider the constraint $S =$ MTR($j = 4$) (a "reasonably well-constrained" system for recording applications), and we compare codes at rate $8/9$ (a "reasonably high" rate for combined modulation and ECC in a recording application), but allowing for the possibility of $N > 9$.

Table II presents a list of four rate $8/9$ codes for MTR($j = 4$). The first code is a block code with the shortest block length q that permits a nonzero parity insertion rate and code rate at least $8/9$. Here, $q = N = 18$ and the parity insertion rate is $1/18 \approx 0.0556$ (the encoder operates at rate $16 : 18$). It turns out that if one wants to strictly increase the insertion rate, but still keep the code rate at least $8/9$, then for a block code, the block length must increase to $q = N = 45$. For this code, one can arrange for $u = 3$ and so the insertion rate is $3/45 = 1/15 \approx 0.0667$. However, the large block length means that encoding is probably very complex. On the other hand, the same code rate $(8/9)$ and insertion rate $(1/15)$ can be achieved via a finite-state code (code #3) with block length only $q = 9$ (here, $N = 15$ and $u = 1$). Moreover, the decoder window has length $=$ two 9-bit blocks, and so is comparable (in number of bits) to code #1 (and much shorter than that of code #2). Also, according to (2) only five of the 15 phases need be used for an encoder, and it turns out that such an encoder can be constructed with roughly five states per phase. Finally, code #4 again has block length $q = 9$, code rate $= 8/9$, and further improved insertion rate $2/27 \approx 0.0741$. But this code probably requires a larger decoding window (three 9-bit blocks).

Actually, for MTR($j = 4$), one can compute $Cap(S_{\{0\},15}) \approx 0.901$. This suggests trying for a rate $9 : 10$ modulation code with parity insertion rate $= 1/15$—thereby improving the rate $(8/9)$ of code #3 above. Moreover, according to (2), the encoder need use only

$$\frac{15}{\gcd(15,\ 10)} = 3$$

of the 15 phases of $G_{\{0\},15}$; however, estimates using an approximate eigenvector [22] show that such an encoder will have an average of at least 37 states per phase, yielding a total of at least 100 encoder states (and possibly many more).

Of course, everything we have done in this section for MTR constraints applies equally well to $(0, k)$-RLL constraints via binary complementation.

IV. Long Codes for MTR Constraints

In Section III, we considered codes for MTR constraints based on relatively short block lengths. For instance, for MTR($j = 4$) and code rate $8/9$, we found a code with parity insertion rate approximately 0.0741. The encoder had block length $q = 9$ and the decoder had a window of at most 27 bits. We can improve upon the insertion rate, at the expense of increasing the block length, by the following construction, essentially due to [7], [9].

For $0 \le k \le j$, we say that a string is a *k-bit-stuffing-MTR(j) string* if it is obtained as follows: begin with a string s that satisfies the MTR($j-k$) constraint and subdivide s into intervals of size $j-k+1$; then, in between each of these intervals insert a string of k ones. The resulting string satisfies the MTR(j) constraint and has parity insertion rate $x = \frac{k}{j+1}$. The set of all such strings of a fixed length N can be viewed as a block code, and the asymptotic optimal code rate of such codes, as $N \to \infty$, is $\frac{j-k+1}{j+1} \log \lambda_{j-k}$ (where λ_m denotes the largest eigenvalue of the standard adjacency matrix $A^{(m)}$ for the MTR(m) constraint). Of course, if we want to ensure that free concatenations of codewords also satisfy the constraint, then we must add a "0" bit at the end of the entire string.

If the desired insertion rate x is not a multiple of $\frac{1}{j+1}$, we can construct a code via a weighted average of two bit-stuffing schemes: if $\frac{k}{j+1} < x < \frac{k+1}{j+1}$, consider a weighted average of k-bit-stuffing-MTR(j) and $(k+1)$-bit-stuffing-MTR(j): subdivide an interval of some large length N into two subintervals and do k-bit-stuffing in the first subinterval and $(k+1)$-bit-stuffing in the second subinterval; the subintervals should have lengths $(1-w)N$ and wN, where

$$w = \frac{x - \frac{k}{j+1}}{\frac{1}{j+1}}. \tag{3}$$

Note that the asymptotic optimal code rate of such block codes is

$$(1-w) * \frac{j-k+1}{j+1} \log \lambda_{j-k} + (w) * \frac{j-k}{j+1} \log \lambda_{j-k-1}. \tag{4}$$

Again, in order to ensure that the resulting strings satisfy the MTR(j) constraint, we need to insert a "0" bit in between the two subintervals.

For MTR($j = 4$), the parity insertion rate $x \approx 0.0741$ lies in between $\frac{0}{j+1} = 0$ and $\frac{1}{j+1} = 1/5$. Thus, this insertion rate can be realized via a weighted average of 0-bit-stuffing and 1-bit-stuffing with weight $w \approx 5 * 0.0741 \approx 0.3705$ (according to (3) with $k = 0$). This yields an asymptotic code rate, as in (4), of approximately

$$(1 - 0.3705) * 1 * 0.9752 + 0.3705 * (4/5) * 0.9468 \approx 0.8945 > 8/9.$$

Thus, the code rate $8/9$ can be achieved with insertion rate x strictly larger than 0.0741. Indeed, setting (4) equal to $8/9$, solving for w, and then solving for x in (3), we get $x \approx 0.0793$; so we can achieve code rate $8/9$ with insertion rate as high as 0.0793.

For $S = \text{MTR}(j)$ and parity insertion rate x, let $g_S(x)$ denote the code rate obtained by the weighted average of k-bit-stuffing and $(k+1)$-bit-stuffing, described above. According to (4), the graph of $g_S(x)$ is the piecewise-linear curve that connects the $j+1$ points

- $(0, \log \lambda_j)$
- $\left(\frac{1}{j+1}, \frac{j}{j+1} \log \lambda_{j-1}\right)$
- $\left(\frac{2}{j+1}, \frac{j-1}{j+1} \log \lambda_{j-2}\right)$
- $\cdots$
- $\left(\frac{j-1}{j+1}, \frac{2}{j+1} \log \lambda_1\right)$
- $\left(\frac{j}{j+1}, \frac{1}{j+1} \log \lambda_0\right) = \left(\frac{j}{j+1}, 0\right)$.

This is illustrated in Fig. 6 for $j = 4$. Each point where the slope changes is indicated by an "$*$." The point plotted as "o" is $(0.0793, 0.8889)$, indicating that a weighted average of two bit-stuffing schemes can achieve code rate $8/9$ with insertion rate approximately 0.0793 (as mentioned above).

One might imagine achieving still higher rates by using a weighted average of more than two of the bit-stuffing schemes mentioned above. However, Fig. 6 suggests that $g_S(x)$ is concave, and so this would not yield any improvement. Indeed, this is the case.

Proposition 1: For all positive integers j, the function $g_S(x)$ is concave on the domain $[0, \frac{j}{j+1}]$ (in fact, strictly concave on the domain of points $\{0, \frac{1}{j+1}, \ldots, \frac{j}{j+1}\}$). Thus, for parity insertion rate x with $\frac{k}{j+1} < x < \frac{k+1}{j+1}$, the weighted average of k-bit-stuffing-MTR(j) and $(k+1)$-bit-stuffing-MTR(j) yields a strictly higher code rate than any other weighted average of bit stuffings.

Proof: Since concavity is not affected by an affine change of the independent variable, the proposition is equivalent to the following lemma, which we prove in the Appendix. □

Lemma 2: The function

$$k \mapsto k \log(\lambda_{k-1}) \tag{5}$$

is strictly concave on the domain of positive integers.

At this point, it is natural to ask if weighted averages of these bit-stuffing schemes are optimal, i.e., if $g_S(x)$ is the asymptotic optimal rate of codes that satisfy the constraint $S = \text{MTR}(j)$ for a given insertion rate x. This turns out to be true for very special cases (see Theorem 3 later). However, it is false in general. For example, for $S = \text{MTR}(j = 4)$, $N = 14$ and $U = \{2, 3, 7, 8, 11, 12, 13\}$ (and so $x = 0.5$), it turns out that $Cap(S_{U,N}) \approx 0.4031$, yet $g_S(0.5) \approx 0.4026$. We leave, as an open problem, the question of whether or not there is a larger class of simple bit-stuffing schemes that completely describe the optimal coding schemes (for all j).

We pause to put this in a more formal setting. Recall that $\overline{S}_{U,N}$ denotes the set of sequences in $S_{U,N}$ of length exactly N.

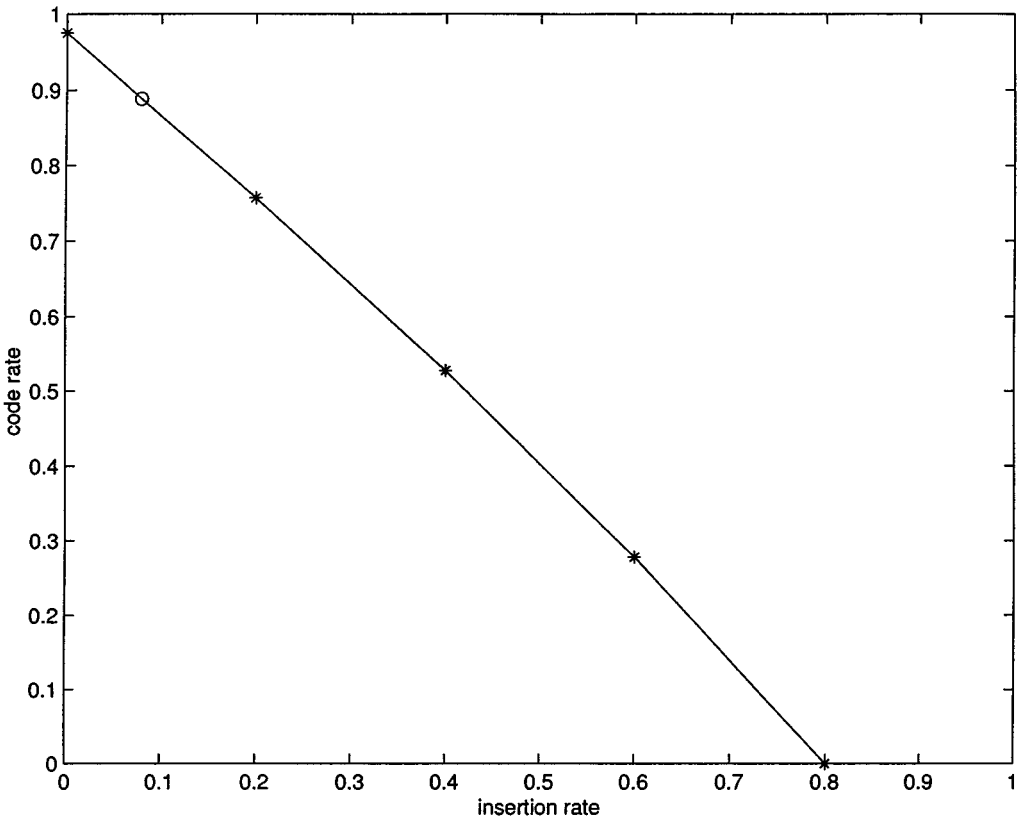

Fig. 6. $g_S(x)$ for MTR($j = 4$).

Given a constrained system S and $x \in [0, 1]$, let $f_S(x)$ denote the limiting, as $N \to \infty$, optimal code rate with parity insertion rate x. Precisely

$$f_S(x) = \limsup_{N\to\infty} \max_{\{U\subseteq\{0,\ldots,N-1\}:|U|/N=x\}} \frac{\log|\overline{S}_{U,N}|}{N}. \quad (6)$$

Of course, this makes sense only for rational x. Note that since

$$Cap(S_{U,N}) \le \frac{\log|\overline{S}_{kU,kN}|}{kN}$$

it follows that $f_S(x)$ dominates the rate of any finite-state code into the constraint $S_{U,N}$ with $|U|/N = x$. Moreover, it also dominates the rate achievable by any scheme based on reversed concatenation (Fig. 3) with the parity modulation encoder required to be systematic (such as that studied by Fan [7], [9]) or the Wijngaarden–Immink scheme [21] based on insertion of parity bits discussed in the Introduction.

For MTR constraints, it is not possible to achieve insertion rates above $\frac{j}{j+1}$: for if $x > \frac{j}{j+1}$ and N is sufficiently large, then a subdivision of any string in $\overline{S}_{U,N}$ into consecutive nonoverlapping intervals of length $j+1$ will contain at least one interval consisting of $j+1$ ones, and so would violate the MTR(j) constraint. Thus, for $S = \text{MTR}(j)$ and $x > \frac{j}{j+1}$, we have $f_S(x) = -\infty$. Clearly, for $0 \le x \le \frac{j}{j+1}$, we have $f_S(x) \ge g_S(x)$ and both $f_S(x)$ and $g_S(x)$ are concave. However, as we said above, equality does not hold in general, although it does hold in very special cases.

Theorem 3: For $j = 1$ and $j = 2$ and $0 \le x \le \frac{j}{j+1}$

$$f_S(x) = g_S(x).$$

We give a complete proof of this for $j = 2$ as follows (the proof for $j = 1$ is considerably easier). Recall that $A^{(j)}$ denotes the standard adjacency matrix for the MTR(j) constraint; in particular

$$A = A^{(2)} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}. \quad (7)$$

Let $B = B^{(j)}$ denote the matrix obtained from $A^{(j)}$ by replacing all entries of the first column by zeros; in particular

$$B = B^{(2)} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \quad (8)$$

Given a length N, a parity insertion rate $x \in [0, 1]$ and a specification of unconstrained positions $U \subseteq \{0, \ldots, N-1\}$ with $|U|/N = x$, define the matrix

$$M = M_0 M_1 \cdots M_{N-1},$$
$$\text{where } M_i = B \text{ if } i \in U \text{ and } M_i = A \text{ if } i \notin U. \quad (9)$$

Each entry of M represents the number of sequences in $\overline{S}_{U,N}$ that begin with a restricted set of prefixes and suffixes; specifically, $M_{u,v}$ is the number of sequences in $\overline{S}_{U,N}$ that begin with at most $3-u$ ones and end with exactly $v-1$ ones. It then follows that the sum of the entries in the first row of M is exactly the total number of sequences of length N that satisfy the MTR(j) constraint with the positions U unconstrained

$$\left|\overline{S}_{U,N}\right| = \left(M\overline{1}\right)_1 = \sum_{v=1}^{3} M_{1,v}$$

(here, $\overline{1}$ denotes the column vector consisting entirely of ones). The crux of our proof of Theorem 3 is the following lemma, which is proved in the Appendix.

Lemma 4: There is a fixed constant K_0 satisfying the following. Given a length N, a parity insertion rate $x \in [0, 1]$ and a specification of unconstrained positions $U \subseteq \{0, \ldots, N-1\}$ with $|U|/N = x$, let M be the matrix defined in (9). Then there is a matrix of the form

$$M' = A^{3n_1}(A^2B)^{n_2}(AB^2)^{n_3} \tag{10}$$

such that

1) $M\overline{1} \leq M'\overline{1}$,
2) the number of occurrences of B in (10) is within K_0 of xN, and
3) the number of occurrences of A in (10) is within K_0 of $(1-x)N$.

Proof of Theorem 3 for $j = 2$: It follows from Lemma 4 (part 1) and Perron–Frobenius Theory [14, Ch. 4] that, up to a fixed multiplicative constant, $M\overline{1}$ is dominated by

$$\mu_2^{n_1}\mu_1^{n_2}\mu_0^{n_3}$$

where μ_0, μ_1, and μ_2 are the largest eigenvalues of AB^2, A^2B, and A^3, respectively. We claim that

$$\mu_0 = \lambda_0, \quad \mu_1 = \lambda_1^2, \quad \mu_2 = \lambda_2^3$$

(recall that λ_m denotes the largest eigenvalue of the adjacency matrix $A^{(m)}$). This follows by straightforward computation (in fact, more generally, for arbitrary j and $0 \leq k \leq j$, one can use the recurrence relation $f_m = f_{m-1} + \cdots + f_{m-j-1}$ to show that $(A^{(j)})^{k+1}(B^{(j)})^{j-k}$ is a 2×2 block triangular matrix with diagonal blocks 0 and $(A^{(k)})^{k+1}$, and so indeed $\mu_k = \lambda_k^{k+1}$). So, up to a fixed multiplicative constant, $M\overline{1}$ is dominated by

$$\lambda_2^{3n_1}\lambda_1^{2n_2}\lambda_0^{n_3}.$$

Let

$$N' = 3n_1 + 3n_2 + 3n_3.$$

Then $\frac{\log|\overline{S}_{U,N}|}{N'}$ is dominated by a weighted average of the numbers

$$\left\{\log(\lambda_2),\ \tfrac{2}{3}\log(\lambda_1),\ \tfrac{1}{3}\log(\lambda_0)\right\}$$

with weights

$$\left\{\frac{3n_1}{N'}, \frac{3n_2}{N'}, \frac{3n_3}{N'}\right\}.$$

These same weights applied to the numbers $\{0/3, 1/3, 2/3\}$ yield

$$\frac{3n_1}{N'} * \frac{0}{3} + \frac{3n_2}{N'} * \frac{1}{3} + \frac{3n_3}{N'} * \frac{2}{3} = \frac{n_2 + 2n_3}{N'}. \tag{11}$$

According to Lemma 4 (parts 2 and 3), the numerator (respectively, denominator) of the right-hand side of (11) differs from xN (respectively, N) by at most K_0 (respectively, $2K_0$). Thus, as $N \to \infty$, the right-hand side of (11) tends to x. Since $g_S(x)$ is continuous and concave (Proposition 1), $f_S(x)$ is dominated by (and hence equal to) $g_S(x)$. □

V. Follower Sets of Constrained Systems

In this section, we briefly summarize some background on follower sets for constrained systems. For a more thorough treatment, the reader may consult [14] or [22].

Given a set S of finite sequences and a finite sequence u, the *follower set* of u is defined as follows:

$$\mathcal{F}(u) = \mathcal{F}_S(u) = \{\text{finite sequences } v\text{: } uv \in S\}.$$

We allow u to be the empty word ϵ, in which case the follower set is all of S. Note that if u does not occur in S, then $\mathcal{F}(u)$ is empty. Any constrained system has only finitely many follower sets [14], [22].

Since a constrained system typically has infinitely many words, many follower sets must coincide with one another. For example, for the constrained system MTR(j) the follower set of a word depends only on its suffix of length j; in fact, this system has only $j+1$ follower sets: $\mathcal{F}(0)$, $\mathcal{F}(01)$, $\mathcal{F}(011)$, $\ldots$, $\mathcal{F}(1^j)$. The follower sets can be used to manufacture a special presentation G, called the *follower set graph*, of a constrained system S. Namely, the states of G are follower sets and the transitions are

$$\mathcal{F}(u) \xrightarrow{a} \mathcal{F}(ua), \qquad a \in \{0, 1\} \tag{12}$$

provided that ua occurs in S. Note in particular that the follower set graph is deterministic. Note also that whenever a word x is the label of a path in the follower set graph ending at state $\mathcal{F}(u)$, we must have $\mathcal{F}(u) \subseteq \mathcal{F}(x)$.

Some follower sets are helpful for proofs but not so much for code construction. Clearly, the follower set of the empty word is an example—more generally, so is any follower set that has no incoming edges or no outgoing edges.

An *irreducible graph* is a graph such that for any given ordered pair of states I, J there is a path in the graph from I to J. Any graph can be decomposed into irreducible subgraphs (called *irreducible components*) together with transient connections from one component to another. An *irreducible constrained system* is a constrained system such that for any given ordered pair of words u and v in S, there is a word w such that uwv is also in S. It turns out that any irreducible constrained system can be presented by an irreducible component of its follower set graph; this component is sometimes called the *irreducible follower set graph* [14].

Most constrained systems of interest are irreducible. For instance, the irreducible follower set graphs of some MTR and RLL constraints are given in Fig. 7 and 8. These agree with the standard presentations that are usually given for these constraints (in particular, Figs. 1 and 7 agree).

Sometimes the irreducible follower set graph agrees with the follower set graph itself (for instance, for MTR constraints), and sometimes the irreducible follower set graph is obtained by merely deleting the follower set of the empty word (for instance, for RLL constraints). But quite often more follower sets need to be deleted.

As another example, Fig. 9 shows the irreducible follower set graph for the constrained system, defined by requiring that runlengths of zeros be congruent to either 0 or 1 modulo 3.

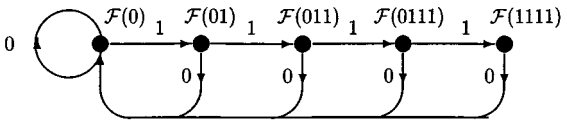

Fig. 7. Irreducible follower set graph for MTR($j = 4$).

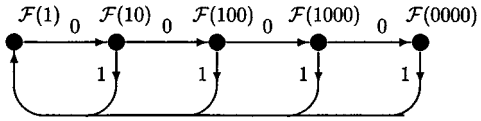

Fig. 8. Irreducible follower set graph for $(1, 4)$-RLL.

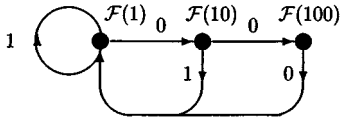

Fig. 9. Irreducible follower set graph for system S with zero runlengths $= 0$ or $1 \bmod 3$.

VI. Presentation of General Constrained Systems With Unconstrained Positions

In this section, we show that $S_{U,N}$ is the set of all sequences that can be generated from a particular phase of a graph $G_{U,N}$ with period N. This graph is defined as follows.

- **States:** All pairs of the form $(\ell, \mathcal{F})$ where
$$\ell \in \{0, \ldots, N-1\}$$
and $\mathcal{F}$ is an intersection (possibly empty) of one or more follower sets of S
- **Transitions:** For $\ell \notin U$, we have the transitions
$$(\ell, \mathcal{F}(u_1) \cap \cdots \cap \mathcal{F}(u_k)) \xrightarrow{a} (\ell + 1 \bmod N, \mathcal{F}(u_1 a) \cap \cdots \cap \mathcal{F}(u_k a))$$
provided that a belongs to $\mathcal{F}(u_1) \cap \cdots \cap \mathcal{F}(u_k)$. For $\ell \in U$, we have the transitions
$$(\ell, \mathcal{F}(u_1) \cap \cdots \cap \mathcal{F}(u_k)) \xrightarrow{1} (\ell + 1 \bmod N, \mathcal{F}(u_1 0) \cap \cdots \cap \mathcal{F}(u_k 0) \cap \mathcal{F}(u_1 1) \cap \cdots \cap \mathcal{F}(u_k 1))$$
provided that both 0 and 1 belong to $\mathcal{F}(u_1) \cap \cdots \cap \mathcal{F}(u_k)$.

For a particular ℓ, the states of the form $(\ell, \mathcal{F})$ constitute the *ℓth phase* of $G_{U,N}$.

Note that for each state $(\ell, \mathcal{F})$, $\mathcal{F}$ is an actual intersection of follower sets; so, if two distinct collections of follower sets have the same intersection, then they define the same state in each phase of $G_{U,N}$.

For given S, U and N, the following result gives a graphical description of $S_{U,N}$ and, in principle, allows us to compute the maximum possible code rate for a modulation code which encodes into S and allows for the positions $i \bmod N \in U$ to be unconstrained.

Theorem 5:

1) The transitions of $G_{U,N}$ are well-defined.
2) $S_{U,N}$ is the set of all sequences that can be generated from phase 0 in $G_{U,N}$.

Proof:

Part 1: To show that the transitions are well-defined we must verify that whenever $a \in \{0, 1\}$ and
$$\mathcal{F}(u_1) \cap \cdots \cap \mathcal{F}(u_k) = \mathcal{F}(u'_1) \cap \cdots \cap \mathcal{F}(u'_m) \tag{13}$$
we have
$$\mathcal{F}(u_1 a) \cap \cdots \cap \mathcal{F}(u_k a) = \mathcal{F}(u'_1 a) \cap \cdots \cap \mathcal{F}(u'_m a). \tag{14}$$
To see this, observe that if v belongs to the left-hand side of (14), then av belongs to the left-hand side of (13) and so av belongs to the right-hand side of (13) and so v belongs to the right-hand side of (14).

Part 2: Suppose that $x = x_0 \cdots x_{\ell-1}$ is the label of a sequence of transitions in $G_{U,N}$ beginning in phase 0. Then, $x \in S$ because x is the label of a sequence of transitions in the follower set graph of S. Now, since the transition at phases $i \in U$ are all labeled 1, it follows that $x_i = 1$ for each $i \bmod N \in U$. According to the definition of $S_{U,N}$, it remains only to show that any U-flip of x belongs to S. For this, consider the graph formed from $G_{U,N}$ by adding a parallel transition labeled 0 to each transition beginning in each phase in U. By construction, every sequence presented by this augmented graph is the label of a sequence of transitions in the follower set graph of S. It follows that any U-flip of x belongs to S, as desired.

For the converse, we show that any $x = x_0 \cdots x_{\ell-1} \in S_{U,N}$ can be presented by a sequence of transitions in $G_{U,N}$ beginning in phase 0. In fact, we claim that x is the label of such a path ending at state $(\ell \bmod N, \cap_{\{U\text{-flips } y \text{ of } x\}} \mathcal{F}(y))$. We prove this by induction on the length ℓ of x. For the base case $\ell = 1$, this follows from the fact that the empty word is allowed as a follower set.

So, assume this is true for $\ell - 1$, and write $x' = x_0 \cdots x_{\ell-2}$. Since $x \in S_{U,N}$, for any U-flip y of x, we have $y \in S$. If $\ell - 1 \bmod N \notin U$, then the set of such U-flips is the set of all words of the form $y = y' x_{\ell-1} \in S$ where y' is a U-flip of x'. Thus, we have the transition
$$\left(\ell - 1 \bmod N, \cap_{\{U\text{-flips } y' \text{ of } x'\}} \mathcal{F}(y')\right) \xrightarrow{x_{\ell-1}} \left(\ell \bmod N, \cap_{\{U\text{-flips } y \text{ of } x\}} \mathcal{F}(y)\right)$$
in $G_{U,N}$. If $\ell - 1 \bmod N \in U$, then $x_{\ell-1} = 1$ and the set of U-flips of x is the set of all words of the form $y = y' a \in S$ where y' is a U-flip of x' and a is either 0 or 1. Thus, we have the transition
$$\left(\ell - 1 \bmod N, \cap_{\{U\text{-flips } y' \text{ of } x'\}} \mathcal{F}(y')\right) \xrightarrow{1} \left(\ell \bmod N, \cap_{\{U\text{-flips } y \text{ of } x\}} \mathcal{F}(y)\right).$$
Thus, $x = x_0 \cdots x_{\ell-1}$ can be generated by a sequence of transitions in $G_{U,N}$ beginning in phase 0. □

Sometimes there are ordering relationships between follower sets that make for far fewer intersections of follower sets than might be expected. For instance, consider the case of a *linearly ordered* constrained system, i.e., a constrained system such that for any pair of follower sets, one is contained in the other. Then the intersection of any collection of follower sets is a follower set itself. Prominent examples of such systems are RLL(0, k) and MTR(j) constraints. For MTR($j = 4$), a comparison of the irreducible follower set graph in Fig. 7 with the graph $G_{U,N}$ in Fig. 5 shows that indeed in each phase there is one state for each follower set.

Finally, recall that $\overline{S}_{U,N}$ denotes the set of all sequences of paths of length N in $S_{U,N}$, and so $\overline{S}_{U,N}$ is the set of all label sequences of paths in $G_{U,N}$ that begin at $(0, \mathcal{F}(\epsilon))$. It follows that the size $|\overline{S}_{U,N}|$ of this set can, in principle, be computed from the adjacency matrix of $G_{U,N}$. Note that $r = \lfloor \log(|\overline{S}_{U,N}|) \rfloor / N$ is the maximal rate at which we can encode into sequences of length N such that

1) all codewords obey the constraint S (although concatenations of codewords need not obey the constraint);
2) for any codeword the bit value in any position $u \in U$ is 1 but can be freely switched to 0 without violating the constraint S.

VII. Simplifications

The graph $G_{U,N}$ may be enormous relative to G, even for small N. However, the number of states can be reduced in the following steps.

Step 1: Not all intersections of follower sets are needed at all phases. Rather, we need only a collection of intersections that is closed under the following operations:

- for $\ell \in U$
$$(\ell, \mathcal{F}(u_1) \cap \cdots \cap \mathcal{F}(u_k)) \mapsto (\ell + 1 \bmod N, \mathcal{F}(u_1 0) \cap \cdots \cap \mathcal{F}(u_k 0) \cap \mathcal{F}(u_1 1) \cap \cdots \cap \mathcal{F}(u_k 1))$$
- for $\ell \notin U$
$$(\ell, \mathcal{F}(u_1) \cap \cdots \cap \mathcal{F}(u_k)) \mapsto (\ell + 1 \bmod N, \mathcal{F}(u_1 0) \cap \cdots \cap \mathcal{F}(u_k 0))$$
and
$$(\ell, \mathcal{F}(u_1) \cap \cdots \cap \mathcal{F}(u_k)) \mapsto (\ell + 1 \bmod N, \mathcal{F}(u_1 1) \cap \cdots \cap \mathcal{F}(u_k 1))$$
(of course, we need only those above that define valid transitions in $G_{U,N}$).

So, starting only with follower sets (but not intersections of follower sets) in phase 0, we can accumulate, cyclically from phase to phase, only those states obtained from applying these operations, until no new states occur; of course, we may have to traverse each phase several times until the process stops. This generally results in far fewer states.

Step 2: Even after Step 1, we still may be left with *inessential* states, i.e., states which are either not the terminal state or initial state of arbitrarily long words (in particular, we can delete states of the form $(\ell, \mathcal{F})$ where $\mathcal{F}$ is either empty or the follower set of the empty word). For example, in the graph of Fig. 5, the inessential states are the second and fifth states in phase 0, the third and fifth states in phase 1, and the first and fourth states in phase 2.

As another example, consider the system S shown in Fig. 9, with $N = 5$ and $U = \{0\}$. In phase 0, all outgoing edges must be labeled "1." But since "1" cannot follow "100," for any $\mathcal{F} \supset \mathcal{F}(100)$, the state $(0, \mathcal{F})$ has no outgoing edges and is therefore inessential. But this forces other states in phase 0 to be inessential. For example, we can easily check that the only path outgoing from the state $(0, \mathcal{F}(1) \cap \mathcal{F}(10))$ is as shown in the expression at the bottom of the page. Since the terminal state of this path is inessential, each state in this path, in particular $(0, \mathcal{F}(1) \cap \mathcal{F}(10))$, must be inessential. In fact, it turns out that all that remains after deleting inessential states is shown in Fig. 10.

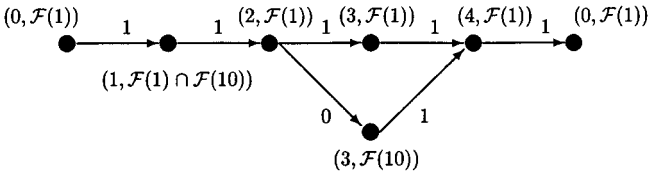

Fig. 10. Presentation of $S_{\{0\},5}$ for system S with zero runlengths $= 0, 1 \bmod 3$.

Step 3: Third, even if S is irreducible, it still can happen that the graph resulting from Steps 1 and 2 is reducible. However, there is always at least one irreducible component of maximal capacity. For coding purposes, we can delete all but one such component.

VIII. Finite-Memory Systems

Recall that a presentation G of a constrained system S has *finite memory* M if whenever any two paths in G of length M have the same label sequence, they end at the same state; and constrained systems that have a finite memory presentation are called finite-memory systems. For such a system, the follower set graph always has finite memory; in fact, the follower set of any word of length $\geq M$ equals the follower set of its suffix of length M. Prominent examples of finite-memory systems are RLL, MTR systems, and their NRZ precoded versions. The system in Fig. 9 and the well-known charge-constrained systems do not have finite memory. The following result shows how the graph $G_{U,N}$ can be simplified for finite-memory systems.

Theorem 6: Let S be an irreducible constrained system of finite memory M and an N and U satisfying the

Gap Condition: the gaps (modulo N) between elements of U are all of size at least M.

Then the Procedure given in Step 1 of Section VII results in states of the form $(\ell, \mathcal{F})$ where $\mathcal{F}$ is either a *singleton*, i.e., a single follower set or a *doubleton*, i.e., an intersection of exactly two follower sets. Moreover, the only phases ℓ for which a state $(\ell, \mathcal{F})$ can be a doubleton are those where $\ell = i + k \bmod N$, where $i \in U$ and $1 \leq k \leq M - 1$.

Proof: Without loss of generality, we can assume that $0 \in U$. First observe that, applying the procedure in Step 1, we begin

$$(0, \mathcal{F}(1) \cap \mathcal{F}(10)) \xrightarrow{1} (1, \mathcal{F}(1) \cap \mathcal{F}(10) \cap \mathcal{F}(100)) \xrightarrow{0} (2, \mathcal{F}(1) \cap \mathcal{F}(10) \cap \mathcal{F}(100)) \xrightarrow{0}$$
$$(3, \mathcal{F}(1) \cap \mathcal{F}(10) \cap \mathcal{F}(100)) \xrightarrow{0} (4, \mathcal{F}(1) \cap \mathcal{F}(10) \cap \mathcal{F}(100)) \xrightarrow{0} (0, \mathcal{F}(1) \cap \mathcal{F}(10) \cap \mathcal{F}(100)).$$

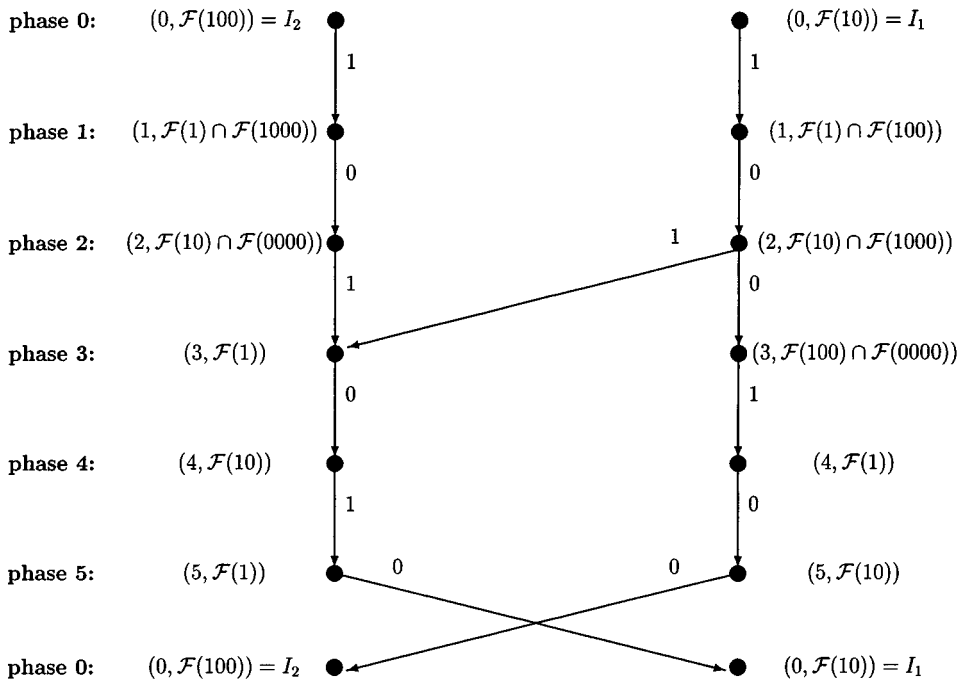

Fig. 11. Presentation of $S_{\{0\},6}$ for $S = (1, 4)$-RLL.

with singletons in phase 0, and because of the Gap Condition, we accumulate only (at worst) doubletons in phases $1, \ldots, M$. In particular, any state accumulated in phase M will be of the form $(M, \mathcal{F}(v0u) \cap \mathcal{F}(v1u))$ for some word u of length M. But by definition of finite memory, $\mathcal{F}(v0u) = \mathcal{F}(u) = \mathcal{F}(v1u)$, and so in phase M, these are really singletons. They will remain singletons in phases $M+1, M+2, \ldots$ until the next phase $\ell \in U \bmod N$ is encountered; from then on, we will see only (at worst) doubletons for at most $M-1$ more phases, again because of the Gap Condition. But in the next phase, the finite memory condition will force singletons, etc. □

As an example, consider S, the $(1, 4)$-RLL system, with $N = 6$ and $U = \{0\}$. Beginning with the irreducible follower set graph in Fig. 8, we apply the constructions of Theorems 5 and 6. We claim that after eliminating inessential states, we are left with the graph, shown in Fig. 11, which presents $S_{U,N}$. To see this, first observe that for any word u, the state $(0, \mathcal{F}(u))$ will have an outgoing edge in $G_{U,N}$ only if both $u0$ and $u1$ belong to S. This eliminates the states $(0, \mathcal{F}(1))$ and $(0, \mathcal{F}(0000))$. For state $(0, \mathcal{F}(1000))$, there is an outgoing transition, labeled 1 to state $(1, \mathcal{F}(1) \cap \mathcal{F}(0000))$, but neither 0 nor 1 can be generated from this state. So, the only surviving states in phase 0 are

$$I_1 = (0, \mathcal{F}(10)) \quad \text{and} \quad I_2 = (0, \mathcal{F}(100)).$$

By starting from these states, and applying the Procedure in Step 1, one can check that all that remains of $G_{U,N}$ is that shown in Fig. 11.

Now, note that in this graph there are exactly three paths of length 6 from phase 0 to itself, one from each of I_1 to I_1, I_1 to I_2, and I_2 to I_1. It follows that $Cap(S_{U,N}) = \log(\lambda)/6$ where λ is the largest eigenvalue of the matrix

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

It is well-known that $\lambda = (1+\sqrt{5})/2$ and $\log(\lambda) \approx 0.6942$. So, $Cap(S_{U,N}) \approx 0.6942/6 \approx 0.116$.

Further simplifications are possible if we are willing to combine some phases together. This will be useful for simplifying the capacity computation and for constructing rate $p : q$ codes in the case where q is a multiple of N. We need the following formal constructions to do this.

1) Let G be a graph and n a positive integer. The *higher power graph* G^n is the graph with the same state set as G, and an edge labeled by a sequence of length n for each path in G of length n (with the label inherited from the path).

2) For a graph G and an integer M, let $G^{(M)}$ denote the graph with states in G and the following transitions: whenever there are two paths from state I to state J, one with label $x_0x_1 \cdots x_{M-1}$ and the other with label $\overline{x}_0x_1 \cdots x_{M-1}$, endow $G^{(M)}$ with a transition from I to J and label $1x_1 \cdots x_{M-1}$.

3) For graphs $G_0, G_1, \ldots, G_{k-1}$, each with the same set $\mathcal{V}$ of states, let $G_0 \uplus G_1 \uplus \cdots \uplus G_{k-1}$ denote the graph ("trellis construction") defined by the following.

   - **States:** The union of k disjoint copies, $\mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_{k-1}$, of $\mathcal{V}$.
   - **Transitions:** For each $i = 0, 1, \ldots, k-1$, mimic each transition in G_i, with a transition from $\mathcal{V}_i$ to $\mathcal{V}_{i+1}$ (where the subscripts are read modulo k).

Now, let S be an irreducible constrained system with memory M presented by its irreducible follower set graph G. Assume the Gap Condition and that $0 \in U$. Write the unconstrained set $U = \{0, u_1, \ldots, u_k\}$ and write $G' = G^{(M)}$. From Theorem 6, we see that $S_{U,N}$ is the set of sequences generated by

$$\overline{G} = G' \uplus G^{u_1-M} \uplus G' \uplus G^{u_2-u_1-M} \cdots \uplus G' \uplus G^{N-u_k-M}$$

beginning in $\mathcal{V}_0$. The adjacency matrix of the higher power graph $\overline{G}^N$ restricted to phase 0 is

$$\overline{A} = A' A^{u_1 - M} A' A^{u_2 - u_1 - M} \cdots A' A^{N - u_k - M}$$

where A is the adjacency matrix of G, and A' is the adjacency matrix of G'. Thus, the capacity of $S_{U,N}$ can then be computed as $(1/N)\log(\lambda)$ where λ is the largest eigenvalue of $\overline{A}$. Note that in the special case $U = \{0\}$, this reduces to

$$\overline{A} = A' A^{N-M}.$$

As an example, consider again $S_{6,\{0\}}$ for $S = (1, 4)$-RLL. Here

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The graph has memory $M = 4$ and one can check that the graph $G^{(M)}$ has only three edges: $\mathcal{F}(10) \stackrel{1001}{\rightarrow} \mathcal{F}(1)$, $\mathcal{F}(10) \stackrel{1010}{\rightarrow} \mathcal{F}(10)$, and $\mathcal{F}(100) \stackrel{1010}{\rightarrow} \mathcal{F}(10)$, and so

$$A' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

We compute

$$\overline{A} = A' A^2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

the essential part of which is simply the 2×2 submatrix determined by the second and third rows and columns

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

consistent with the computation earlier in this section.

Appendix
Proofs of Lemmas 2 and 4

Proof of Lemma 2: It is well known [13, p. 61] that λ_k is the unique solution $z > 1$ to the equation

$$z^{k+2} - 2z^{k+1} + 1 = 0.$$

Let a be a real variable. We claim that for each value of $a > 0$, the equation

$$z^{a+2} - 2z^{a+1} + 1 = 0 \tag{15}$$

has a unique solution $z = z(a) \in (1, 2)$. This is a consequence of the following facts regarding the left-hand side of (15):

- it is 0 at $z = 1$,
- it is 1 at $z = 2$,
- it has negative derivative (as a function of z) at $z = 1$ (namely, the derivative is $-a$),
- it has derivative 0 at only one point $z > 0$ (namely, at $z = \frac{2(a+1)}{a+2}$).

In particular, note that

$$z(a) > \frac{2(a+1)}{a+2}. \tag{16}$$

We claim that $z'(a) > 0$ and so $z(a)$ is monotonically increasing with a. To see this, first rewrite (15) as $(2-z)z^{a+1} = 1$, take natural logs and differentiate z with respect to a to obtain

$$z'(a)\left(\frac{1}{2-z} - \frac{a+1}{z}\right) = \ln(z)$$

equivalently

$$z'(a) = \frac{(2-z)z\ln(z)}{(a+2)z - 2(a+1)}$$

which is positive by (16).

For $a > 1$, let

$$u(a) = a \log z(a-1).$$

Note that $u'(a) > 0$ and so $u(a)$ is increasing with a. It suffices to show that $u''(a) < 0$ on the domain $a > 1$. We find it easier instead to show that $u''(a) < 0$ merely for all $a \geq 4$ and then use an auxiliary argument to complete the proof of concavity of the function (5) on the domain of positive integers.

For this, first note that $z(a-1) = 2^{u(a)/a}$. It follows that $u(a)$ satisfies the equation

$$2^{u(a)(1+1/a)} - 2^{u(a)+1} + 1 = 0$$

which we can rewrite as

$$2^{u(a)/a} = 2 - 2^{-u(a)}. \tag{17}$$

In what follows, we will write $u(a)$, $u'(a)$, $u''(a)$ as simply u, u', u''. Differentiating (17) with respect to a, we obtain

$$2^{u/a}[u'/a - u/a^2] = 2^{-u}u'.$$

Solving for u', we obtain

$$u' = \frac{2^{u/a}u}{a2^{u/a} - a^2 2^{-u}}.$$

Now, substituting for $2^{u/a}$ via (17), we obtain

$$u' = \frac{(2 - 2^{-u})u}{2a - 2^{-u}a - a^2 2^{-u}}.$$

Multiplying numerator and denominator by 2^u, we obtain

$$u' = \frac{(2^{u+1} - 1)u}{2^{u+1}a - a - a^2}. \tag{18}$$

Differentiating this equation with respect to a, we see that

$$u'' = \eta/\delta$$

where δ is positive

$$\eta = (2^{u+1}a - a - a^2)((2^{u+1} - 1)u' + 2^{u+1}u'uC) - (2^{u+1} - 1)(2^{u+1}au'C + 2^{u+1} - 1 - 2a)u$$

and $C = \ln 2$.

We will show that $\eta < 0$ for all $a \geq 4$. Now, using (18), we substitute for u' *only* in the second factor of the first term of η and obtain

$$\eta = (2^{u+1} - 1)((2^{u+1} - 1) + 2^{u+1}uC)u - (2^{u+1} - 1)(2^{u+1}au'C + 2^{u+1} - 1 - 2a)u.$$

But this expression simplifies to

$$\eta = (2^{u+1}-1)(2^{u+1}uC + 2a - 2^{u+1}au'C)u.$$

Thus, $\eta < 0$ if and only if

$$u' > \frac{2^{u+1}uC + 2a}{2^{u+1}aC} = u/a + 2^{-u}/C. \tag{19}$$

Now, we can rewrite (18) as

$$u' = (u/a)\left(1 + \frac{a}{2^{u+1}-a-1}\right).$$

So, the inequality (19) is equivalent to

$$\frac{u}{2^{u+1}-a-1} > 2^{-u}/C.$$

Recalling that $u' > 0$, it follows from (18) that $2^{u+1}-a-1 > 0$, and so the preceding inequality is equivalent to

$$u > 2/C - (a+1)/(2^uC). \tag{20}$$

For $a = 4$, one computes that

$$u = 4\log\lambda_3 \approx (4)(0.9468) \approx 3.78 > 2/C.$$

Now, since u is increasing with a, we have $u > 2/C$ for all $a \geq 4$; thus, (20) holds for $a \geq 4$. Thus, the function (5) is concave on the domain of integers $k \geq 4$. Now, one can verify, via explicit computation, that this function is concave on the domain of integers $\{1, 2, 3, 4, 5\}$. Since the domain of positive integers is the union of these two domains, which intersect in two consecutive integers (namely, $\{4, 5\}$), it follows that the function (5) is concave on the entire domain of positive integers, as desired (use the characterization of a concave function as a function with decreasing slopes). □

Proof of Lemma 4: Let A and B be as in (7) and (8). We will make use of the following matrix relations:

R1: $B^3 = 0$;

R2: $AB^2A \leq A$;

R3: $B(AB)^n = BAB$ for all $n \geq 1$;

R4: $A(A^2B)^nABA^2 \leq (A^2B)^{n+1}A^2$ for all $n \geq 1$;

R5: $A(A^2B)^nA^mA^2B \leq A^{m+1}(A^2B)^{n+1}$ for all $n, m \geq 1$.

The first three of these relations can be verified by straightforward computation. The fourth and fifth, which we verify below, are a bit more subtle.

We will use these relations to gradually transform M into the desired form M'.

It follows from R1 that we may assume in (9) that we never see three consecutive B's.

Stage 1: From R2, we see that we can delete each appearance of AB^2 without decreasing any entry of M (we can assume in (9) that M ends with an A). But in the course of doing so, we change the length N and insertion rate x. We will rectify this in a moment; but for now, simply let m_0 denote the total number of occurrences of AB^2 that we have deleted in this stage.

Stage 2: At this point, we may assume that M contains only isolated copies of B. It follows from R3 that we can replace any occurrence of $A^2B(AB)^nA^2$ with A^2BABA^2 and not change any entry of M (again, this changes the length N and insertion rate x). Let m_1 denote the total number of occurrences of AB that we have deleted in Stage 2. Then, tack on to the right end of M the matrix

$$(A^2B)^{\lceil m_1/3\rceil}(AB^2)^{\lceil m_1/3\rceil + m_0}.$$

This will not decrease $M\overline{1}$. Note that if m_1 were divisible by 3, then this would completely rectify the length and insertion rate. Otherwise, it changes, in (9), the number of occurrences of B and the number of occurrences of A by bounded amounts.

Stage 3: Now M is a product of several intermingled powers of A, A^2B, and isolated copies of AB, followed by a single power of A^2B and a single power of AB^2. Using R4, we can combine each isolated copy of AB with an A to form another copy of A^2B. Then, using R5, we can combine all powers of A together and all powers of A^2B together, yielding a matrix of the form

$$M' = A^m(A^2B)^{m_2}(AB^2)^{m_3}.$$

Then we can delete some initial copies of A to make m divisible by 3 at the expense of changing the number of occurrences of A by a bounded amount. This completes the proof of Lemma 4, except for the verification of inequalities R4 and R5.

Verification of R4: Consider the sequence of integers generated by the recurrence

$$f_n = f_{n-1} + f_{n-2}$$

with initial conditions: $f_0 = f_1 = 1$; this is the well-known sequence of Fibonacci numbers. Now, by a simple induction, one can show that

$$(A^2B)^n = \begin{bmatrix} 0 & f_{2n} & f_{2n-1} \\ 0 & f_{2n} & f_{2n-1} \\ 0 & f_{2n-1} & f_{2n-2} \end{bmatrix}.$$

From this, one computes

$$A(A^2B)^nABA^2 = \begin{bmatrix} 4f_{2n+1} & 2f_{2n+1} & 0 \\ 2f_{2n+2} & f_{2n+2} & 0 \\ 2f_{2n+1} & f_{2n+1} & 0 \end{bmatrix}$$

and

$$(A^2B)^{n+1}A^2 = \begin{bmatrix} f_{2n+4} & f_{2n+3} & 0 \\ f_{2n+4} & f_{2n+3} & 0 \\ f_{2n+3} & f_{2n+2} & 0 \end{bmatrix}.$$

Comparing these two matrices, we see that it suffices to show

$$4f_n \leq f_{n+3} \quad \text{and} \quad 2f_n \leq f_{n+2}.$$

For the latter, observe that

$$2f_n < 2f_n + f_{n-1} = f_{n+1} + f_n = f_{n+2}.$$

For the former, observe that

$$\begin{aligned} f_{n+3} &= f_{n+2} + f_{n+1} = 2f_{n+1} + f_n \\ &= 3f_n + 2f_{n-1} > 3f_n + f_{n-1} + f_{n-2} = 4f_n. \end{aligned}$$

Verification of R5: For each $0 \leq u \leq 2$ and $1 \leq v \leq 2$, let $S_{u,v}$ denote the set of sequences that satisfy MTR($j = 2$), begin with exactly u ones and end with exactly v ones, and are of the form

$$a_0a_1a_2 1 a_3a_4 1 \cdots a_{2n-1}a_{2n}1 b_0 \cdots b_{m-1}c_0c_1 1. \tag{21}$$

Let $T_{u,v}$ be the set of sequences that satisfy MTR($j = 2$), begin with exactly u ones and end with exactly v ones, and are of the form

$$d_0 \cdots d_m e_0 e_1 1 e_2 e_3 1 \cdots e_{2n}e_{2n+1}1.$$

We will show the following.

a) For all combinations of (u, v), except $u = 2$, $v = 1$, there is a one-to-one (but not necessarily onto) mapping: $\phi_{u,v}$: $S_{u,v} \to T_{u,v}$.

b) There is a partition of $S_{2,1}$

$$S_{2,1} = S' \cup S''$$

and one-to-one (but not necessarily onto) mappings $\phi_{2,1}$: $S' \to T_{2,1}$ and ψ: $S'' \to T_{0,1} \setminus \phi_{0,1}(S_{0,1})$

Since the $(r+1, v+1)$ entry of the left-hand side of R5 is $\sum_{u \geq r} |S_{u,v}|$ and the same entry of the right-hand side of R5 is $\sum_{u \geq r} |T_{u,v}|$, inequality R5 will then follow.

The mappings are all constructed by shifting some ones and reversing the order of most of each sequence of the form (21). Specifically, for a), the mappings are

$(u = 0, v = 1)$:

$$\phi_{0,1}(0a_1a_2 1a_3a_4 1 \cdots a_{2n-1}a_{2n}1b_0 \cdots b_{m-1}c_0 01) = 0c_0b_{m-1} \cdots b_0 a_{2n} 1 a_{2n-1} \cdots 1a_3a_2 1a_1 01$$

$(u = 0, v = 2)$:

$$\phi_{0,2}(0a_1a_2 1a_3a_4 1 \cdots a_{2n-1}a_{2n}1b_0 \cdots b_{m-1}011) = 0b_{m-1} \cdots b_0 a_{2n}a_{2n-1} 1 \cdots 1a_4a_3 1a_2a_1 1011$$

$(u = 1, v = 1)$:

$$\phi_{1,1}(10a_2 1a_3a_4 1 \cdots a_{2n-1}a_{2n}1b_0 \cdots b_{m-1}c_0 01) = 10c_0b_{m-1} \cdots b_0 1 a_{2n}a_{2n-1} 1 \cdots 1a_4a_3 1a_2 01$$

$(u = 1, v = 2)$:

$$\phi_{1,2}(10a_2 1a_3a_4 1 \cdots a_{2n-1}a_{2n}1b_0 \cdots b_{m-1}011) = 10b_{m-1} \cdots b_0 a_{2n} 1 a_{2n-1}a_{2n-2} 1 \cdots a_4 1 a_3a_2 1011$$

$(u = 2, v = 2)$:

$$\phi_{2,2}(1101a_3a_4 1 \cdots a_{2n-1}a_{2n}1b_0 \cdots b_{m-1}011) = 110b_{m-1} \cdots b_0 1 a_{2n}a_{2n-1} \cdots 1a_4a_3 1011.$$

For b), let S'' denote the subset of $S_{2,1}$ defined by $b_1 = 1$ and $b_2 = 1$ (with the notation in (21)), and let $S' = S_{2,1} \setminus S''$. Define $\phi_{2,1}$ by

$$\phi_{2,1}(1101a_3a_4 1 \cdots a_{2n-1}a_{2n}1b_0 \cdots b_{m-1}c_0 01) = 110c_0b_{m-1} \cdots b_1 1 b_0 a_{2n} 1 a_{2n-1}a_{2n-2} 1 \cdots 1a_5a_4 1a_3 01$$

and ψ by

$$\psi(1101a_3a_4 1 \cdots a_{2n-1}a_{2n}1b_0 \cdots b_{m-1}c_0 01) = 00c_0b_{m-1} \cdots b_0 1 a_{2n}a_{2n-1} 1 a_{2n-2} \cdots 1a_4a_3 1001.$$

We must show that $\psi(S'') \subseteq T_{0,1} \setminus \phi_{0,1}(S_{0,1})$, equivalently, that the images of ψ and $\phi_{0,1}$ are disjoint. This follows from the following.

1) By definition of S'', for any sequence in the image of ψ, there are ones in both positions $m+1$ and $m+2$ (counting from the left with the first position viewed as position 1).

2) For any sequence in the image of $\phi_{0,1}$, there cannot be ones in both positions $m+1$ and $m+2$ (with the same counting convention as in 1)) because otherwise the corresponding domain sequence in $S_{0,1}$ would have $b_0 = b_1 = 1$ and thus violate the MTR($j = 2$) constraint. □

REFERENCES

[1] R. L. Adler, D. Coppersmith, and M. Hassner, "Algorithms for sliding block codes—An application of symbolic dynamics to information theory," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 5–22, Jan. 1983.

[2] K. Anim-Appiah and S. McLaughlin, "Constrained-input turbo codes for $(0, k)$ RLL channels," in *Proc. CISS Conf.*, Baltimore, MD, 1999.

[3] ——, "Toward soft output ASPP decoding for nonsystematic nonlinear block codes," unpublished paper, 2000.

[4] J. Ashley and B. Marcus, "Time-varying encoders for constrained systems: An approach to limiting error propagation," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1038–1043, May 2000.

[5] W. G. Bliss, "Circuitry for performing error correction calculations on baseband encoded data to eliminate error propagation," *IBM Tech. Discl. Bull.*, vol. 23, pp. 4633–4634, 1981.

[6] T. Conway, "A new target response with parity coding for high density magnetic recording," *IEEE Trans. Magn.*, vol. 34, pp. 2382–2386, 1998.

[7] J. Fan, "Constrained coding and soft iterative decoding for storage," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1999.

[8] J. Fan and R. Calderbank, "A modified concatenated coding scheme, with applications to magnetic data storage," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1565–1574, July 1998.

[9] J. Fan and J. Cioffi, "Constrained coding techniques for soft iterative decoders," in *Proc. IEEE GLOBECOM*, 1999.

[10] J. Fan, B. Marcus, and R. Roth, "Lossles sliding-block compression of constrained systems," *IEEE Trans. Inform. Theory*, vol. 46, pp. 624–633, Mar. 2000.

[11] C. Frieman and A. Wyner, "Optimum block codes for noiseless input restricted channels," *Inform. Contr.*, vol. 7, pp. 398–415, 1964.

[12] K. A. S. Immink, "A practical method for approaching the channel capacity of constrained channels," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1389–1399, Sept. 1997.

[13] ——, *Codes for Mass Data Storage Systems*. Eindhoven, The Netherlands: Shannon Foundation Publishers, 1999.

[14] D. Lind and B. Marcus, *An Introduction to Symbolic Dynamics and Coding*. New York: Cambridge Univ. Press, 1995.

[15] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar. 1999.

[16] M. Mansuripur, "Enumerative modulation coding with arbitrary constraints and post-modulation error correction coding and data storage systems," *Proc. SPIE*, vol. 1499, pp. 72–86, 1991.

[17] J. Moon and B. Brickner, "Maximum transition run codes for data storage," *IEEE Trans. Magn.*, vol. 32, pp. 3992–3994, Sept. 1996.

[18] L. Reggiani and G. Tartara, "On reverse concatenation and soft decoding algorithms for PRML magnetic recording channels," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 612–618, Apr. 2001.

[19] C. E. Shannon, "The mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 1948.

[20] A. Wijngaarden and K. Immink, "Efficient error control schemes for modulation and synchronization codes," in *Proc. Int. Symp. Information Theory*, Cambridge, MA, Aug. 1998, p. 74.

[21] ——, "Maximum run-length limited codes with error control properties," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 602–611, Apr. 2001.

[22] B. H. Marcus, R. M. Roth, and P. H. Siegel, "Constrained systems and coding for recording channels," in *Handbook of Coding Theory*, V. S. Pless and W. C. Huffman, Eds. Amsterdam, The Netherlands: Elsevier, 1998.

[23] B. Moision and P. H. Siegel, "Periodic-finite-type shift spaces," in *Proc. IEEE Int. Symp. Information Theory*, Washington, DC, June 24–29, 2001, p. 65.