Chapter 4

Finite-State Encoders

As described in Section 1.4, an encoder takes the form of a finite-state-machine (see Figure 1.6). In this chapter, we make the definition of finite-state encoders more precise and then discuss special types of finite-state encoders and various levels of decoding capabilities.

4.1 Definition of finite-state encoders

Let S be a constrained system and n be a positive integer. An *(S, n)-encoder* is a labeled graph $\mathcal{E}$ such that —

- each state of $\mathcal{E}$ has out-degree n;
- $S(\mathcal{E}) \subseteq S$;
- $\mathcal{E}$ is lossless.

A *tagged (S, n)-encoder* is an (S, n)-encoder $\mathcal{E}$ where the outgoing edges from each state in $\mathcal{E}$ are assigned distinct *input tags* from an alphabet of size n. The notation $u \stackrel{s/a}{\to} v$ stands for an edge in $\mathcal{E}$ from state u to state v which is labeled a and tagged by s. The tag s is supposed to represent an input and the label a is supposed to represent an output. We will sometimes use the same symbol $\mathcal{E}$ to denote both a tagged (S, n)-encoder and the underlying (S, n)-encoder.

A *rate $p : q$ finite-state encoder for S* is a tagged $(S^q, 2^p)$-encoder, where we assume that the input tags are the binary p-blocks.

A tagged (S, n)-encoder (or a rate $p : q$ finite-state encoder for S) is deterministic or has finite anticipation or is definite according to whether the (S, n)-encoder (or $(S^q, 2^p)$-encoder) satisfies these conditions. In particular, only the (output) labels (and not the input tags) play a role in these properties.

As mentioned in Section 1.4, we sometimes use the term *rate* to mean the ratio p/q, instead of the pair of block sizes $p : q$. It will be clear from the context which version of the term rate we mean.

Example 4.1 Figure 4.1 depicts a rate $1 : 2$ two-state encoder for the $(1,3)$-RLL constrained system. The input tag assigned to each edge is written before the slash, followed by

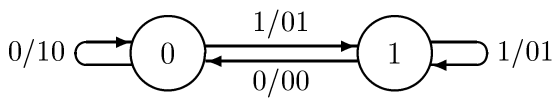

Figure 4.1: Rate $1 : 2$ two-state encoder for $(1,3)$-RLL constrained system.

the label of the edge. The encoder in the figure is known as the Modified Frequency Modulation (MFM) code and is due to Miller [Mill63]. This encoder is actually deterministic. □

Given a rate $p : q$ finite-state encoder $\mathcal{E}$ for S, encoding is accomplished as follows.

1. Select an arbitrary initial state u_0 in $\mathcal{E}$.
2. If the current state is u and the input data is the p-block $\mathbf{s}$, find the outgoing edge e from state u in $\mathcal{E}$ with input tag $\mathbf{s}$. The *codeword* generated is the q-block that labels the edge e. The next encoder state is $\tau_{\mathcal{E}}(e)$.
3. Repeat Step 2 as long as input is provided.

With only the losslessness assumption, decoding can be implemented, but it is terribly impractical because one cannot decode any symbols at all until an entire codeword sequence (i.e., sequence of q-blocks) is received; also, one can decode only those codeword sequences that are labels of paths that start at u_0 and terminate in a particular state. However, if an encoder $\mathcal{E}$ has finite anticipation $\mathcal{A} = \mathcal{A}(\mathcal{E})$, then we can decode in a state-dependent manner as follows (this kind of decoding was discussed briefly in Section 1.4).

1. Use the initial state u_0 of $\mathcal{E}$ as the initial state of the decoder.
2. If the current state is u, then the current codeword to be decoded, together with the $\mathcal{A}$ upcoming codewords, constitute a word of length $\mathcal{A}+1$ (measured in q-blocks) that is generated by a path that starts at u; by definition of anticipation, the initial edge e of such a path is uniquely determined; the reconstructed (decoded) data is the input tag of e; the next decoder state is $\tau_{\mathcal{E}}(e)$.

3. Repeat Step 2 as long as codewords are provided.

The anticipation of an encoder $\mathcal{E}$ measures the *decoding delay* of the corresponding state-dependent decoder. Note that the decoder will recover all but the last $\mathcal{A}$ encoded q-blocks.

The encoder in Example 4.1 is deterministic (equivalently, finite anticipation with $\mathcal{A} = 0$), and so can be decoded by a state-dependent decoder with no decoding delay at all. To illustrate non-trivial decoding delay, we revisit Example 1.1 of Section 1.4.

Example 4.2 Figure 4.2, which is the same as Figure 1.7, depicts a rate $2:3$ two-state encoder for the $(0,1)$-RLL constrained system [Sie85a]. This encoder is not deterministic,

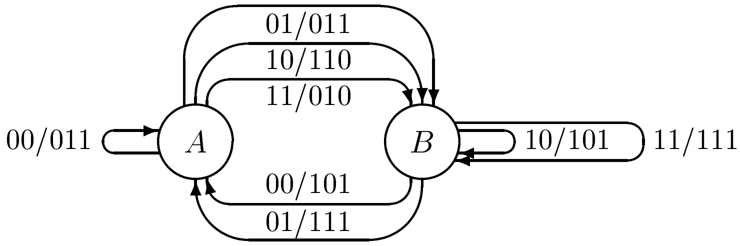

Figure 4.2: Rate $2:3$ two-state encoder for $(0,1)$-RLL constrained system.

but has anticipation 1: for any path, the initial state and the first two 3-bit output labels uniquely determine the initial edge. So, this encoder can be decoded with delay equal to one block. □

Perhaps the simplest class of encoders is that of block encoders. A *block* (S,n)*-encoder* $\mathcal{E}$ is an (S,n)-encoder with only one state. Such an encoder can be viewed as simply a *dictionary* $\mathcal{L}$, consisting of n symbols. Denote by $\mathcal{L}^*$ the set of words that are obtained by concatenation of words in $\mathcal{L}$. Clearly, $\mathcal{L}^* = S(\mathcal{E}) \subseteq S$, which means that the codewords of the dictionary $\mathcal{L}$ are *freely-concatenable*: every concatenation of the codewords in $\mathcal{L}$ produces a word in S.

As with the more general class of finite-state encoders, we can speak of a *tagged block encoder* and therefore also a *rate* $p:q$ *block encoder*. Such an encoder amounts to a one-to-one mapping from the set of all 2^p binary words of length p to the q-blocks in the dictionary. Note that the inverse of this mapping serves as a decoder.

Shannon [Sha48] showed that whenever there is a rate $kp:kq$ block encoder, for some k, we must have $p/q \le \mathsf{cap}(S)$ (we will give a stronger version of this in Theorem 4.2 below). Conversely, he proved (nonconstructively) that whenever $p/q < \mathsf{cap}(S)$, there is an integer k and a rate $kp:kq$ block encoder (see Theorem 4.3 below). The following result improves upon this.

Theorem 4.1 (Finite-state coding theorem) *Let S be a constrained system. If $p/q \leq \mathsf{cap}(S)$, then there exists a rate $p : q$ finite-state encoder for S with finite anticipation.*

Theorem 4.1 is proved in Chapter 5; it guarantees encoders which are state-dependent decodable at any rate up to capacity. It can be derived as a weaker version of the main theorem of [ACH83]. It improves upon the earlier coding results, in particular Shannon's result mentioned above, in three important ways:

- It is fairly constructive: it effectively provides encoders whose number of states is close to the smallest possible (see also Section 7.2).
- It proves the existence of finite-state encoders that achieve rate *equal* to the capacity $\mathsf{cap}(S)$, when $\mathsf{cap}(S)$ is rational.
- For any positive integers p and q satisfying the inequality $p/q \leq \mathsf{cap}(S)$, there is a rate $p : q$ finite-state encoder for S that operates at rate $p : q$. In particular, choosing p and q relatively prime, one can design an encoder/decoder using the smallest possible codeword length (namely, q) compatible with the chosen rate p/q.

For the purposes of constructing (S, n)-encoders, we could restrict our attention to irreducible components of a labeled graph presenting S. We do not pay any price in terms of achievable rate in doing so, because we can always choose an irreducible component with a maximum largest eigenvalue (see Theorem 3.7).

Moreover, when G is irreducible, but G^q decomposes into irreducible components, then, as in Theorem 3.10, the components are isolated and all have the same largest eigenvalue. So, for encoding purposes, we are free to use any such component, although some components may result in simpler encoders than others (see [How89], [WW91]).

Example 4.3 Let G be the Shannon cover in Figure 2.6 of the 2-charge constrained system and consider the two irreducible components in Figure 2.16, G_0 and G_1, of G^2. The edges of the component G_1 can be tagged so as to give a rate $1 : 2$ block encoder for the 2-charge constrained system. □

The following is the converse to Theorem 4.1.

Theorem 4.2 (Finite-state converse-to-coding theorem) *Let S be a constrained system. If there exists a rate $p : q$ finite-state encoder for S, then $p/q \leq \mathsf{cap}(S)$.*

Proof. Let $\mathcal{E}$ be a rate $p : q$ finite-state encoder for S. We may assume that $\mathcal{E}$ is irreducible; for otherwise, replace $\mathcal{E}$ by one of its irreducible sinks (recall the notion of sink from Section 2.5.1). By Theorem 3.4, since $\mathcal{E}$ is lossless,

$$\log \lambda(A_{\mathcal{E}}) = \mathsf{cap}(S(\mathcal{E})). \tag{4.1}$$

But since each state of $\mathcal{E}$ has exactly 2^p outgoing edges, we have

$$A_{\mathcal{E}}\mathbf{1} = 2^p\mathbf{1} ,$$

where $\mathbf{1}$ is the column vector of all 1's. Thus, by the Perron-Frobenius Theorem (Theorem 3.11), we have,

$$\lambda(A_{\mathcal{E}}) = 2^p .$$

Putting this together with (4.1), and observing that $S(\mathcal{E}) \subseteq S^q$, we obtain

$$p = \mathsf{cap}(S(\mathcal{E})) \leq \mathsf{cap}(S^q) ,$$

and so $p/q \leq \mathsf{cap}(S)$. $\square$

In Example 4.1, we exhibited a rate $1 : 2$ finite-state encoder for the $(1, 3)$-RLL constrained system. So, the capacity of this system must be at least $1/2$. Indeed, from Table 3.1 we see that the capacity is approximately .5515. Similarly, from Example 4.2 it follows that the capacity of the $(0, 1)$-RLL constrained system must be at least $2/3$, and we know already that this capacity equals $\log\left((1+\sqrt{5})/2\right) \approx .6942$.

Recall from our discussion in Section 1.4 that if encoded data is corrupted by noise, then a state-dependent decoder may lose track of the correct state information and therefore propagate errors indefinitely without recovering. Since state-dependence is not involved in decoding a block encoder, such encoders will limit error propagation (in fact, error propagation will be confined to a single block). For this reason, we will first spend some time discussing block encoders before we pass to the more general framework of encoders which limit error propagation (in Section 4.3).

4.2 Block encoders

Let S be a constrained system presented by a deterministic graph G and let q be a positive integer. We can obtain a rate $p : q$ block encoder for S as follows. We pick an arbitrary state u in G and find the largest integer p such that $(A_G^q)_{u,u} \geq 2^p$. Then, we construct the encoder dictionary by taking 2^p words of length q that are generated by cycles in G that start and terminate in u. In fact, if we choose these words in consecutive lexicographic order, then a codeword can be reconstructed *efficiently* from its index in the dictionary. This technique is known as *enumerative coding* (see [Cov73], [Imm91, p. 117], [TB70], and Problem 4.2). The question is whether the attainable rate p/q can be made large enough so that we can approach capacity. The answer turns out to be positive, as summarized in the following theorem.

Theorem 4.3 (Block coding theorem [Sha48]) *Let S be a constrained system. There exists a sequence of rate $p_m : q_m$ block encoders for S such that $\lim_{m\to\infty} p_m/q_m = \mathsf{cap}(S)$.*

Proof. Let G be an irreducible deterministic graph such that $\mathsf{cap}(S(G)) = \mathsf{cap}(S)$; by Theorem 3.7, such a graph indeed exists. We will further assume that G is in fact primitive; the general irreducible case can be deduced by appealing to Theorem 3.10.

By Theorem 3.17, it follows that for every state u in G we have

$$\lim_{\ell \to \infty} \frac{1}{\ell} \log(A_G^\ell)_{u,u} = \log \lambda(A_G) = \mathsf{cap}(S) \ .$$

Hence, for every state u in G and $\epsilon > 0$, there exist integers q and $p = \lfloor q(\mathsf{cap}(S) - \epsilon) \rfloor$ such that

$$(A_G^q)_{u,u} \geq 2^{q(\mathsf{cap}(S)-\epsilon)} \geq 2^p \ .$$

Hence, our block encoder construction approaches capacity when $q \to \infty$. □

The following result provides a characterization of the existence of block encoders for a given constrained system.

Proposition 4.4 *Let S be a constrained system with a deterministic presentation G and let n be a positive integer. Then there exists a block (S, n)-encoder if and only if there exists a subgraph H of G and a dictionary $\mathcal{L}$ with n symbols of $\Sigma(S)$, such that $\mathcal{L}$ is the set of labels of the outgoing edges in H from each state in H.*

Proof. If there is such a subgraph H and a dictionary $\mathcal{L}$, then any concatenation of words of $\mathcal{L}$ yields a word in S. Hence, $\mathcal{L}$ is a dictionary of a block encoder.

Conversely, suppose there exists a block (S, n)-encoder $\mathcal{E}$ with a dictionary $\mathcal{L}$. By Lemma 2.9, there is an irreducible component G' of G such that $\mathcal{L}^* = S(\mathcal{E}) \subseteq S(G')$. Hence, by Lemma 2.13, there is a state u in G' such that $\mathcal{L}^* \subseteq \mathcal{F}_{G'}(u)$. In particular, there must be an edge $u \xrightarrow{\mathbf{w}} u_{\mathbf{w}}$ in G' for every $\mathbf{w} \in \mathcal{L}$. Continuing from the terminal states of these edges, there must be edges $u_{\mathbf{w}} \xrightarrow{\mathbf{z}} u_{\mathbf{wz}}$ in G' for every $\mathbf{w}, \mathbf{z} \in \mathcal{L}$. Iterating this process, we end up traversing a subgraph of G', where each state in the subgraph has $|\mathcal{L}| = n$ outgoing edges labeled by $\mathcal{L}$. □

For a given constrained system S, and a given choice of p and q, Proposition 4.4 gives a decision procedure for determining if there is rate $p : q$ block encoder for S (and if so how to construct one) as follows. Let G be a deterministic presentation of S and, for two states u and v in G, let $\mathcal{F}_G^q(u, v)$ denote the set of all words of length q that can generated in G by paths that start at u and terminate in v. Searching for the states of the subgraph H of G^q in Proposition 4.4, we look for a set P of states in G for which

$$\Big| \bigcap_{u \in P} \Big(\bigcup_{v \in P} \mathcal{F}_G^q(u, v) \Big) \Big| \geq 2^p \ .$$

We call such a set P of states a (p, q)-*block set*, and the set on the left-hand side of this inequality the *corresponding dictionary*.

While there are exponentially many (as a function of the number of states of G) subsets to search over, the following result of Freiman and Wyner [FW64, Lemma 2] (see also [MSW92, Section V and Appendix B]) simplifies the procedure in many cases.

Proposition 4.5 *Let S be a constrained system with a finite memory essential presentation G (in particular, G is deterministic). Let p and q be positive integers, and assume that q is at least as large as the memory of G. If there is a (p,q)-block set of vertices in G, then there is a (p,q)-block set P with the following property: whenever u is in P and $\mathcal{F}_G(u) \subseteq \mathcal{F}_G(u')$, then u' is also in P.*

A set of states that satisfies the property, stated in the conclusion of Proposition 4.5, is called a *complete* set of states.

Proof of Proposition 4.5. Let u and u' be states in G such that $\mathcal{F}_G(u) \subseteq \mathcal{F}_G(u')$. Since q is at least as large as the memory of G, it follows that any word of length q generated by a path from state u to some state v can also be generated by a path from state u' to v. Thus, if P is a (p,q)-block set, $u \in P$, and $u' \in V_G$, then the set $P' = P \cup \{u'\}$ is also a (p,q)-block set. So, any (p,q)-block set can be iteratively enlarged until it is complete. □

We show how this method works in the next example.

Example 4.4 Let S be the (0,2)-RLL constrained system, presented by the deterministic graph G in Figure 4.3. Note G has memory $\mathcal{M} = 2$. From Table 3.1, we see that $\mathsf{cap}(S) \approx .8791$, and so it makes sense to ask if there is a rate $p : q$ block encoder for S with $p = 4$ and $q = 5$. For this, first observe that the follower sets of states in G satisfy

$$\mathcal{F}_G(2) \subset \mathcal{F}_G(1) \subset \mathcal{F}_G(0) ,$$

and so the only complete sets are $P_0 = \{0\}$, $P_1 = \{0,1\}$, and $P_2 = \{0,1,2\}$. Now, the adjacency matrix for G and its fifth power are

$$A_G = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

and

$$A_G^5 = \begin{pmatrix} 13 & 7 & 4 \\ 11 & 6 & 3 \\ 7 & 4 & 2 \end{pmatrix} .$$

For P_0, the corresponding dictionary is the set of labels of the self-loops in G^5 at state 0. The number of such self-loops is $(A_G^5)_{1,1} = 13 < 16 = 2^4$; so, P_0 is not a (p,q)-block set.

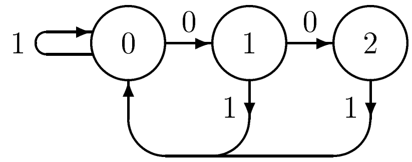

Figure 4.3: Presentation G of $(0, 2)$-RLL constrained system.

For P_1, since $\mathcal{F}(1) \subset \mathcal{F}(0)$ and $q = 5 > 2 = \mathcal{M}$, the corresponding dictionary is the set of labels of paths that start at state 1 and end at either state 0 or 1. The size of this dictionary is the sum of the $(2, 1)$ and $(2, 2)$ entries of A_G^5, and this is $17 > 16 = 2^4$. Thus, P_1 is a $(4, 5)$-block set and defines a rate $4 : 5$ block encoder for S by deleting one word from the dictionary and then assigning 4-bit tags to the remaining codewords. A particular assignment for such a code, that was actually used in practice, is shown in [MSW92, Table III].

For P_2 the corresponding dictionary is the set of labels of paths that start at state 2 and end at any of the states 0, 1, or 2. Thus, the size of the dictionary is the sum of the entries in the third row of A_G^5. Since this is only 13, P_2 fails to be a $(4, 5)$-block set. So, of the three complete sets, only one of them, P_1, yields a rate 4:5 block encoder. □

The same method can also be applied to give a rate $8 : 9$ block encoder for the $(0, 3)$-RLL constrained system. with capacity approximately .9468; see [MSW92, p. 25]. Yet, in both of these examples the rates of the codes ($4/5 = .8$ for $(0, 2)$-RLL and $8/9 \approx .8889$ for $(0, 3)$-RLL) are quite far from capacity (.8791 for $(0, 2)$-RLL and .9468 for $(0, 3)$-RLL). In order to get code rates much closer to capacity, it typically requires much longer block lengths.

The problem with longer block lengths is twofold. First, one must deal with the problem of designing a practically implementable assignment of a very large number of user words. Secondly, while errors can not propagate beyond a block boundary, if the block length is large, then a large number of bits may be corrupted by a single channel error.

For these reasons, finite-state encoders based on relatively short block lengths can offer an attractive alternative to block encoders based on long block lengths—provided that error propagation can still be controlled satisfactorily. Indeed, this is the role of sliding-block decodability: a sliding-block decoder with small window size will control error propagation. For instance, in Example 4.2, we gave a rather simple rate $2 : 3$ two-state encoder for the $(0, 1)$-RLL constrained system; this rate is relatively close to capacity ($\approx .6942$), and we saw in Section 1.4 that there is a corresponding sliding-block decoder with a very small window. In contrast, to obtain a rate $p : q$ block encoder for this system with $p/q = 2/3 \approx .6667$, the procedure outlined above reveals that it requires block lengths of size $p = 12$ and $q = 18$. As another example, consider the $(1, 7)$-RLL constrained system, whose capacity is approximately .6793. In Section 4.3, we will exhibit a relatively simple rate $2 : 3$ six-state encoder with a sliding-block decoder with small window. In contrast, using a result by

Lee [Lee88] (see also [LW89]), one can show that the smallest block lengths for a rate 2/3 block encoder for this system are $p = 42$ and $q = 63$.

4.3 Sliding-block decodable encoders

Let S be a constrained system over an alphabet Σ and let m and a be integers such that $\mathsf{m}+\mathsf{a} \geq 0$. A tagged (S, n)-encoder is (m, a)-*sliding-block decodable*, if the following holds: for any two paths $e_{-\mathsf{m}}e_{-\mathsf{m}+1} \ldots e_0 \ldots e_{\mathsf{a}}$ and $e'_{-\mathsf{m}}e'_{-\mathsf{m}+1} \cdots e'_0 \ldots e'_{\mathsf{a}}$ that generate the same word, the edges e_0 and e'_0 have the same (input) tag. We will use the shorter term sliding-block decodable encoder to denote a tagged encoder which is (m, a)-*sliding-block decodable* for some m and a.

A *sliding-block decoder* for a tagged (S, n)-encoder $\mathcal{E}$ is a mapping $\mathcal{D}$ from the set $S(\mathcal{E}) \cap \Sigma^{\mathsf{m}+\mathsf{a}+1}$ to the set of input tags, such that, if $\mathbf{w} = w_0w_1w_2 \cdots$ is any symbol sequence generated by the encoder from the input tag sequence $\mathsf{s} = s_0s_1s_2 \cdots$, then, for $i \geq \mathsf{m}$,

$$s_i = \mathcal{D}(w_{i-\mathsf{m}}, \ldots, w_i, \ldots, w_{i+\mathsf{a}}) .$$

We call a the *look-ahead* of $\mathcal{D}$ and m the *look-behind* of $\mathcal{D}$. The sum $\mathsf{m}+\mathsf{a}+1$ is called the *decoding window length* of $\mathcal{D}$. It is easy to verify that a tagged (S, n)-encoder has a sliding-block decoder if and only if it is sliding-block decodable.

A tagged encoder is called *block decodable* if it is $(0, 0)$-sliding-block decodable, equivalently if whenever two edges have the same (output) label, they must also have the same (input) tag.

The following proposition is straightforward.

Proposition 4.6 *If a tagged (S, n)-encoder is (m, a)-definite, then it is (m, a)-sliding-block decodable for any tagging of the edges.*

Notions of sliding-block decodability naturally extend to rate $p : q$ finite-state encoders as follows. Let S be a constrained system over an alphabet Σ. A *sliding-block decoder* for a rate $p : q$ finite-state encoder $\mathcal{E}$ for S is a mapping

$$\mathcal{D} : S(\mathcal{E}) \cap (\Sigma^q)^{\mathsf{m}+\mathsf{a}+1} \longrightarrow \{0, 1\}^p$$

such that, if $\mathbf{w} = w_0w_1w_2 \cdots$ is any sequence of q-blocks (codewords) generated by the encoder from the input tag sequence of p-blocks $\mathsf{s} = s_0s_1s_2 \cdots$, then, for $i \geq \mathsf{m}$,

$$s_i = \mathcal{D}(w_{i-\mathsf{m}}, \ldots, w_i, \ldots, w_{i+\mathsf{a}}) .$$

Figure 1.9 shows a schematic diagram of a sliding-block decoder.

Recall that a single error at the input to a sliding-block decoder can only affect the decoding of q-blocks that fall in a "window" of length at most $\mathsf{m}+\mathsf{a}+1$, measured in q-blocks. Thus, error propagation is controlled by sliding-block decoders.

Example 4.5 The encoder in Figure 4.1 is $(1,0)$-definite. Hence, it is $(1,0)$-sliding-block decodable for any tagging of the edges. Moreover, for the specific tagging shown in Figure 4.1, the encoder is actually block decodable: the second bit in each label equals the input tag. □

Example 4.6 The tagged encoder given in Example 1.2 is not sliding-block decodable. To see this, observe that an arbitrarily long sequence of a's can be generated by the self-loops at each state, and yet the self-loops have different input tags.

Example 4.7 Let $\mathcal{E}$ be the encoder in Figure 4.2. This encoder is $(0,1)$-definite and therefore $(0,1)$-sliding-block decodable. Table 4.1, which we showed earlier in Section 1.4, defines a sliding-block decoder

$$\mathcal{D} : S(\mathcal{E}) \cap \left(\{0,1\}^3\right)^2 \longrightarrow \{00, 01, 10, 11\} .$$

Entries marked by "—" in the table do not affect the value of the decoded input tag s_i. □

w_i (current codeword)	w_{i+1} (next codeword)	$s_i = \mathcal{D}(w_i, w_{i+1})$ (decoded input tag)
010	—	11
011	101 or 111	01
011	010, 011, or 110	00
101	101 or 111	10
101	010, 011, or 110	00
110	—	10
111	101 or 111	11
111	010, 011, or 110	01

Table 4.1: Sliding-block decoder for encoder in Figure 4.2.

The following result shows that for encoders, sliding-block decodability implies finite anticipation. So, sliding-block decodability is indeed a stronger property than state-dependent decodability.

Proposition 4.7 *If an essential tagged (S,n)-encoder is (m,a)-sliding-block decodable, then it has anticipation at most a.*

w_i	w_{i+1}	w_{i+2}	w_{i+3}	s_i
00	00	00	—	0
00	00	01	—	0
00	00	10	00	1
00	00	10	01	0
00	01	—	—	0
00	10	—	—	1
01	00	00	—	0
01	00	01	—	1
01	00	10	00	1
01	00	10	01	0
10	00	00	—	0
10	00	01	—	1
10	00	10	00	1
10	00	10	01	0
10	01	—	—	0

Table 4.2: Decoding function of encoder in Figure 4.4.

Proof. The proof is similar to that given for *Definite* $\Rightarrow$ *Finite anticipation* in Proposition 2.3. □

Example 4.8 The capacity of the $(2,7)$-RLL constrained system is approximately .5174 (see Table 3.1). Figure 4.4 presents a rate $1:2$ six-state encoder for this constrained system. The encoder is $(0,3)$-sliding-block decodable and its decoder, $s_i = \mathcal{D}(w_i, w_{i+1}, w_{i+2}, w_{i+3})$, is

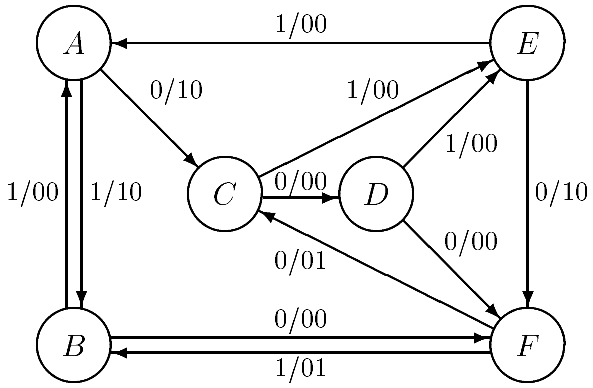

Figure 4.4: Rate $1:2$ six-state encoder for $(2,7)$-RLL constrained system.

presented in Table 4.2. By Proposition 4.7, the anticipation of this encoder is at most 3 and, in fact, it is exactly 3 (see Problem 2.3: the graph in Figure 2.20 is an untagged version of

Figure 4.4, with the labels a, b, and c standing for 00, 01, and 10, respectively). The encoder in Figure 4.4 is due to Franaszek [Fra72] and has been used in several commercial products.

Figure 4.5 shows another rate 1 : 2 (untagged) encoder for the $(2,7)$-RLL constrained system. This encoder, which is due to Howell [How89], has only five states. Yet, its anticipation is 4 (see Problem 2.4). □

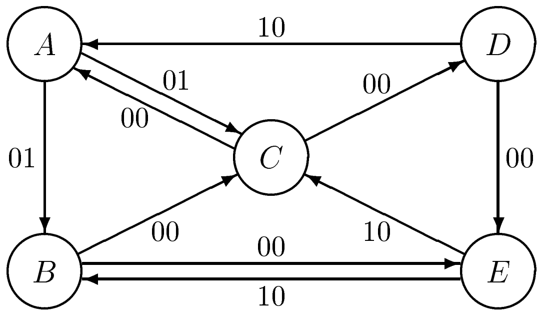

Figure 4.5: Rate 1 : 2 five-state encoder for $(2,7)$-RLL constrained system.

Example 4.9 The capacity of the $(1,7)$-RLL constrained system is approximately .6793. Figure 4.6 presents a rate 2 : 3 four-state encoder for this constrained system. This encoder is due to Weathers and Wolf [WW91] and is $(0,2)$-sliding-block decodable (a sliding-block decoder can be found in [WW91]). □

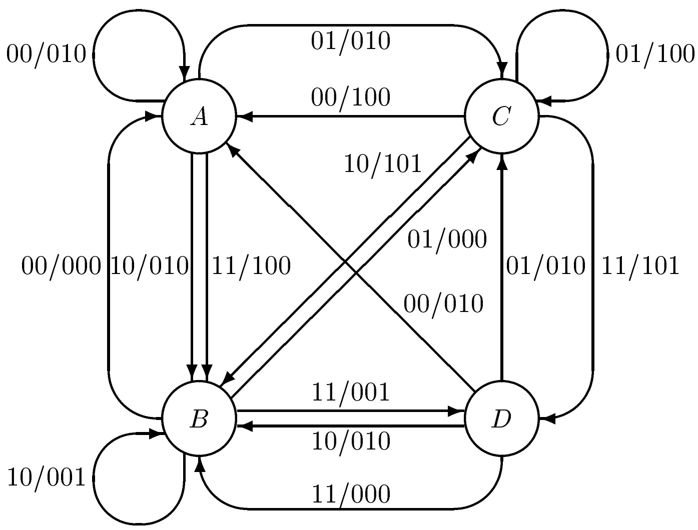

Figure 4.6: Rate 2 : 3 two-state encoder for $(1,7)$-RLL constrained system.

For finite-type constrained systems, Theorem 4.1 can be improved.

Theorem 4.8 (Adler, Coppersmith, and Hassner [ACH83]) *Let S be a finite-type constrained system. If $p/q \le \mathsf{cap}(S)$, then there exists a rate $p : q$ finite-state encoder for S with a sliding-block decoder.*

This result is proved in Section 5.4.

We remark that except in trivial cases it is impossible to have $\mathsf{a} < 0$. On the other hand, it is quite possible to have $\mathsf{m} < 0$ (but still having $\mathsf{m}+\mathsf{a} \ge 0$). A value $\mathsf{m} < 0$ corresponds to the case where the sliding-block decoder reconstructs a tag which was input way back in the past—i.e., m time slots *earlier* than the 'oldest' symbol in the examined window. See Section 6.6, where we briefly discuss how this is used to reduce the decoding window length, $\mathsf{m}+\mathsf{a}+1$, of the decoder and therefore also the error propagation.

Next, we point out that there is an algorithm for testing whether a given tagged (S, n)-encoder is (m, a)-sliding-block decodable. When m is nonnegative, this is a simple modification of that described in Section 2.7.4 (see Problem 4.7); when $\mathsf{m} < 0$, this is given in Proposition 4.10 below. In contrast, however, we have the following.

Theorem 4.9 (Siegel [Sie85b], [AKS96]) *Given an untagged (S, n)-encoder $\mathcal{E}$, the problem of deciding whether there is a tag assignment to the edges of $\mathcal{E}$ such that $\mathcal{E}$ is block decodable (namely, $(0, 0)$-sliding-block decodable) is NP-complete.*

In fact, the proof of this result shows that the input tag assignment problem in Theorem 4.9 is NP-complete even for fixed $n \ge 3$. But the problem becomes polynomial if we fix the size of the alphabet of S.

Finally, we outline here the algorithm to test whether a tagged (S, n)-encoder is (m, a)-sliding-block decodable when $\mathsf{m} < 0$. For a tagged (S, n)-encoder $\mathcal{E}$, let $R_{\mathcal{E}*\mathcal{E}}$ be the $|V_{\mathcal{E}}|^2 \times |V_{\mathcal{E}}|^2$ matrix whose rows and columns are indexed by the states of $\mathcal{E} * \mathcal{E}$, and for every $u, u', v, v' \in V_{\mathcal{E}}$, the entry $(R_{\mathcal{E}*\mathcal{E}})_{\langle u,u'\rangle,\langle v,v'\rangle}$ equals the number of pairs of edges $u \overset{s/a}{\to} v$ and $u' \overset{s'/a'}{\to} v'$ in $\mathcal{E}$, with distinct input tags $s \ne s'$ (but not necessarily with the same label). Also, denote by $A_{\mathcal{E}} \otimes A_{\mathcal{E}}$ the Kronecker product of $A_{\mathcal{E}}$ with itself—i.e., $(A_{\mathcal{E}} \otimes A_{\mathcal{E}})_{\langle u,u'\rangle,\langle v,v'\rangle} = (A_{\mathcal{E}})_{u,v}(A_{\mathcal{E}})_{u',v'}$ for every $u, u', v, v' \in V_{\mathcal{E}}$.

Proposition 4.10 *A tagged (S, n)-encoder $\mathcal{E}$ is (m, a)-sliding-block decodable with $\mathsf{m} < 0$ if and only if $R_{\mathcal{E}*\mathcal{E}}\,(A_{\mathcal{E}} \otimes A_{\mathcal{E}})^{-\mathsf{m}-1}\, A_{\mathcal{E}*\mathcal{E}}^{\mathsf{m}+\mathsf{a}+1} = 0$.*

The full proof of Proposition 4.10 is left as an exercise (Problem 4.10).

4.4 Block decodable encoders

Recall that a tagged encoder is block decodable if it is $(0,0)$-sliding-block decodable. While a block decodable encoder is state dependent, it can be decoded just like a block encoder, and thus block decodable encoders limit error propagation to the same extent as block encoders. The following example shows why block decodable encoders might provide an advantage over block encoders.

Example 4.10 Consider the constrained system S over the alphabet $\{a,b,c,d\}$ which is presented by the labeled graph G of Figure 4.7.

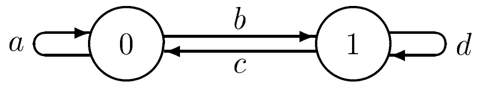

Figure 4.7: Graph presentation for Example 4.10.

Now, suppose we would like to construct a rate $p:q$ block encoder for S. Since the follower sets of the two states in G are disjoint, then, by Proposition 4.4, the codewords of the dictionary must all be generated by cycles that start and terminate in the very same state of G. However, for each of the two states u in G, there are $(A_G^q)_{u,u} = 2^{q-1}$ cycles of length q that start and terminate in u. Hence, the best we can achieve is a rate $(q-1):q$ block encoder for S, which, evidently, does not achieve the capacity $\mathsf{cap}(S) = 1$.

On the other hand, any tagging of the edges of G yields a rate 1:1 block decodable encoder and so achieves capacity. $\Box$

As another example, consider the $(1,3)$-RLL constrained system S. The rate 1:2 two-state encoder for S in Figure 4.1 is block decodable, but is not a block encoder. In fact, we leave it to the reader to verify that there is no rate 1:2 block encoder for this system (to see this, use Proposition 4.4 and the procedure described immediately afterwards).

Block-decodable encoders form a class somewhere intermediate between block encoders and sliding-block decodable encoders. The following result gives a characterization, similar to Proposition 4.4, for the existence of block decodable encoders.

Proposition 4.11 *Let S be a constrained system with a deterministic presentation G and let n be a positive integer. Then there exists a block decodable (S,n)-encoder if and only if there exists such an encoder which is a subgraph of G.*

Proof. The sufficiency of the condition is obvious. The proof of necessity is similar to that of Proposition 4.4. First, we may assume that there exists an irreducible block decodable (S, n)-encoder $\mathcal{E}$. By Lemma 2.13, for each state v in $\mathcal{E}$, there is a state u in G such that $\mathcal{F}_{\mathcal{E}}(v) \subseteq \mathcal{F}_G(u)$; while there may be many such states u, just pick one, and call it $u(v)$. Now, for every symbol a which appears as a label of an outgoing edge in $\mathcal{E}$ from v, there is a unique edge in G outgoing from $u(v)$ with label a; call this edge $e(v, a)$. Then the set of edges $\{e(v, a)\}$ obtained in this way defines a subgraph of G. These edges inherit tags from the corresponding edges in $\mathcal{E}$, and it is evident that this tagged subgraph defines a block decodable (S, n)-encoder. □

Example 4.11 Let S be the $(0, 1)$-RLL constrained system and let G be the Shannon cover of S, as shown in Figure 2.2. We have presented a rate $2 : 3$ two-state encoder for S in Example 4.2. Note that this encoder is not block decodable. We claim that there is no block decodable, rate $2 : 3$ finite-state encoder for S. For if there were such an encoder, then by Proposition 4.11 there would be a subgraph H of G^3, with each state of H having outdegree $= 4$. But there is no such subgraph: G^3 is shown in Figure 2.10; it has outdegree 5 at state 0 and outdegree 3 at state 1, and the deletion of state 1 would leave state 0 with outdegree only 3. □

It follows from Proposition 4.11 that if G is the Shannon cover of a constrained system S and there is a block decodable rate $p : q$ finite-state encoder for S , then there exists a set P of states in G such that

$$\Big| \bigcup_{v \in P} \mathcal{F}_G^q(u, v) \Big| = \sum_{v \in P} (A_G^q)_{u,v} \geq 2^p \quad \text{for every } u \in P \; .$$

A set of states P which satisfies this inequality is referred to in [Fra68] as a *set of principal states*. Note that any (p, q)-block set, as defined in Section 4.2, is necessarily a set of principal states. The existence of a set of principal states is necessary for the existence of a block decodable encoder, in particular for a block encoder. In fact, it is necessary and sufficient for the existence of a deterministic encoder, but in general it is not sufficient for the existence of a block decodable encoder. However, it turns out (see [Fra68] and [Fra70]) that for a special class of irreducible constrained systems, including powers of (d, k)-RLL constrained systems, the existence of a rate $p : q$ deterministic encoder is equivalent to the existence of a rate $p : q$ block decodable encoder. So, for these systems, one can obtain block decodable encoders by searching for a set of principal states. We will see in Section 5.2.2 that a set of principal states, if any exists, can be found efficiently.

An explicit description of block decodable codes for (d, k)-RLL constrained systems is given by Gu and Fuja in [GuF94], and also by Tjalkens in [Tja94]. Their constructions are optimal in the sense that for any given (d, k)-RLL constrained system, and given q, they achieve the highest possible p for a rate $p : q$ block decodable encoder. The Gu–Fuja construction is a generalization of a coding scheme due to Beenker and Immink [BI83].

We now describe the Beenker–Immink construction (see also [Imm91, pp. 116–117]). Let $\mathcal{L}(q; d, k; r)$ denote the set of all q-blocks in the (d, k)-RLL constrained system, with at least d leading zeroes and at most r trailing zeroes. We assume that $q > k \geq 2d$ and that $d \geq 1$, and set $r = k-d$. Encoding is carried out by a one-to-one mapping of the $p = \lfloor \log |\mathcal{L}(q; d, k; k-d)| \rfloor$ input bits (tags) into q-blocks, or codewords, in $\mathcal{L}(q; d, k; k-d)$. Such a mapping can be implemented either by a look-up table (of size 2^p) or by enumerative coding. However, since the codewords in $\mathcal{L}(q; d, k; k-d)$ are not freely-concatenable, the encoded codeword needs to be adjusted: when the concatenation of the previous codeword with the current codeword causes a violation of the (d, k)-RLL constraint, we invert one of the first d zeroes in the latter codeword. The condition $q > k \geq 2d$ guarantees that such inversion can always resolve the constraint violation. The first d bits in each codeword (which are initially zero) are referred to as *merging bits*. Since encoding of a current codeword depends on the previous codeword, the Beenker–Immink encoder is not a block encoder; however, it is block decodable.

We can use Example 4.1 to illustrate how this scheme works (even though the condition $q > k$ is not met). In the example, the set $\mathcal{L}(2; 1, 3; 2)$ consists of the two codewords 00 and 01. When the codeword 00 is to be followed by another 00, we resolve the constraint violation by changing the latter codeword into 10.

A well-known application of the Beenker–Immink method is that of the 3-EFM(16) code that was described in Section 1.7.2. The codewords of this code are taken from the set $\mathcal{L}(16; 2, 10; 8)$, which is of size 257, thus yielding a rate $8 : 16$ block decodable encoder for the $(2, 10)$-RLL constrained system.

In their paper [GuF94], Gu and Fuja show that, for any (d, k)-RLL constrained system S with $k > d \geq 1$, and for any $q \geq d$, a block decodable (S^q, n)-encoder exists if and only if $n \leq |\mathcal{L}(q; d, k; k-1)|$ (Tjalkens presents a similar result in [Tja94] for the range $q \geq k \geq 2d > 1$). Hence, the Beenker–Immink construction is optimal for $d = 1$ and sub-optimal for $d > 1$. The construction presented in [GuF94] that attains the equality $n = |\mathcal{L}(q; d, k; k-1)|$ requires more than just inverting a merging bit; still, as shown in [GuF94], it can be efficiently implemented. See also [Tja94].

4.5 Non-catastrophic encoders

A tagged (S, n)-encoder is a *non-catastrophic encoder* if it has finite anticipation and whenever the sequences of output labels of two right-infinite paths differ in only finitely many places, then the sequences of input tags also differ in only finitely many places. A rate $p : q$ finite-state tagged encoder for S is non-catastrophic if the corresponding tagged $(S^q, 2^p)$-encoder is non-catastrophic.

Observe that non-catastrophic encoders restrict error propagation in the sense that they limit the *number* of decoded data errors spawned by an isolated channel error. In general, however, such encoders do not necessarily limit the *time span* in which these errors occur.

On the other hand, tagged encoders which are sliding-block decodable do limit the time span as well and therefore are preferable.

The following result shows that, with the standard capacity assumption, we can always find non-catastrophic encoders and that whenever there is excess in capacity or whenever the constraint is almost-finite-type (such as the charge-constrained systems), the decoder can be made sliding-block, ensuring that decoder error propagation is limited in both number and time span.

Theorem 4.12 *Let S be a constrained system. If $p/q \le \mathsf{cap}(S)$, then there exists a non-catastrophic rate $p : q$ finite-state encoder for S. Moreover, if, in addition, either $p/q < \mathsf{cap}(S)$ or S is almost-finite-type, then the encoder can be chosen to be sliding-block decodable.*

So, for general constrained systems, the error propagation is guaranteed to be limited only in number. Indeed, in [KarM88] (see Section 5.4), an example is given of a constrained system with rational capacity, for which there is *no* sliding-block decodable encoder with rate equaling capacity (of course, by Theorem 4.12, such a constrained system cannot be almost-finite-type). In the non-catastrophic encoders constructed in Theorem 4.12, the decoding errors generated by an isolated channel error are confined to two bounded bursts, although these bursts may appear arbitrarily far apart (see Section 5.4).

The notion of non-catastrophic encoder is a standard concept in the theory of convolutional codes. In that setting, it coincides with sliding-block decodability [LinCo83, Ch. 10].

The proof of Theorem 4.12 is fairly complicated. We give an outline in Section 5.4. Although it does not exactly provide a practical encoder synthesis algorithm, the proof makes use of some very powerful techniques that can be brought to bear in particular applications. Several of the ideas in the generalization to almost-finite-type systems have also played a role in the design of coded-modulation schemes based upon spectral-null constraints. See, for example, [KS91a].

The quest for a sliding-block decodable encoder with rate equaling capacity for a particular example provided the original motivation for Theorem 4.12. The example is as follows.

Let S be the 6–$(1,3)$-CRLL constrained system (see Section 1.5.5 and Problem 2.16). It turns out that $\mathsf{cap}(S) = 1/2$ (see Problem 3.23). In fact, the only non-trivial B–(d,k)-CRLL constrained systems with rational binary capacity are the 2–$(0,1)$-CRLL and the 6–$(1,3)$-CRLL systems, both with capacity 1/2 [AS87][AHPS93]. For this constraint, Patel [Patel75] constructed a particular rate $1 : 2$ finite-state $(S,2)$-encoder. So, the rate of this encoder is as high as possible. Unfortunately, this encoder does not have finite anticipation. However, Patel was able to modify the encoder to have finite anticipation and even a sliding-block decoder with very small decoding window length, at the cost of only a small sacrifice in rate (although there is an additional cost in complexity). This modified encoder, known as the Zero-Modulation (ZM) code, was used in an IBM tape drive [Patel75].

Recall that charge-constrained systems are almost-finite-type and that runlength-limited systems are finite-type and therefore almost-finite-type. Now, the intersection of two almost-finite-type constrained systems is again almost-finite-type (see Problem 2.8); so, the 6–$(1, 3)$-CRLL constrained system S is almost-finite-type. It then follows from Theorem 4.12 that there actually is a rate $1 : 2$ tagged finite-state $(S, 2)$-encoder which is sliding-block decodable. However, the encoding–decoding complexity of such a code appears to be enormous. On the other hand, there is a rate $4 : 8$ tagged finite-state $(S, 2)$-encoder which is sliding-block decodable, with only moderate encoding–decoding complexity (see [KS91b] and Problem 4.3).

We remark that Ashley [Ash93] has proved a far-reaching generalization of Theorem 4.12.

4.6 Relationships among decodability properties

Finally, in the following result, which we leave as an exercise for the reader, we summarize the relationships among the decodability properties that we have considered in this chapter.

Proposition 4.13 *Let $\mathcal{E}$ be an essential tagged encoder. Then*

$$\left.\begin{array}{ccccc}
\textit{Definite} & \Rightarrow & \begin{array}{c}\textit{Sliding-block}\\ \textit{decodable}\end{array} & \Rightarrow & \textit{Noncatastropic} \\
\Uparrow & & \Uparrow & & \\
\begin{array}{c}\textit{Block}\\ \textit{encoder}\end{array} & \Rightarrow & \begin{array}{c}\textit{Block}\\ \textit{decodable}\end{array} & \Rightarrow & \textit{Deterministic}
\end{array}\right\} \Rightarrow \begin{array}{c}\textit{Finite}\\ \textit{anticipation}\end{array} \Rightarrow \textit{Encoder}\,.$$

4.7 Markov chains on encoders

In Section 3.4, we introduced Markov chains on graphs. Clearly, the definitions apply to (S, n)-encoders as special cases of graphs.

In particular, given a tagged (S, n)-encoder $\mathcal{E} = (V, E, L)$, we can obtain a Markov chain on $\mathcal{E}$ by assuming that the input tags within an input sequence are statistically independent and uniformly distributed. This model is commonly used in analyzing encoders, and it can be approximated rather well in reality by 'scrambling' the sequence of input tags (e.g., assuming that the input tags take values on $\Upsilon = \{0, 1, 2, \ldots, n-1\}$, the ith input tag in the sequence is added modulo n to the ith symbol in some fixed pseudo-random sequence over Υ). Equivalently, in this Markov chain, the conditional probability, q_e, of each edge $e \in E$ is equal to $1/n$.

Example 4.12 We analyze here the 3-EFM(16) code that was described in Section 1.7.2. Recall that this encoder has three states, 0, 1, and 2–8, with out-degree $n = 256$ at each state. The edges are labeled by 16-bit codewords from the $(2, 10)$-RLL constraint.

The adjacency matrix of $\mathcal{E}$ is given by

$$A_{\mathcal{E}} = \begin{pmatrix} 83 & 57 & 116 \\ 83 & 57 & 116 \\ 83 & 57 & 116 \end{pmatrix} .$$

Indeed, the number of edges from state u to state v in $\mathcal{E}$ is independent of u; in fact, the dependency on u of the labels of those edges is restricted only to the merging bits, which are the first two bits of those labels.

A uniform distribution over the input tag sequences induces a Markov chain on $\mathcal{E}$ whose transition matrix is given by

$$Q_{\mathcal{E}} = \tfrac{1}{256} \cdot A_{\mathcal{E}}$$

(see Section 3.4). By Proposition 3.20 it follows that as the path length ℓ increases, the probability of ending at state u in $\mathcal{E}$ converges to the component π_u in the following vector

$$\boldsymbol{\pi}^\top = (\pi_0 \ \pi_1 \ \pi_{2\text{–}8}) = (\tfrac{83}{256} \ \ \tfrac{57}{256} \ \ \tfrac{116}{256}) \approx (\, .324 \ \ .223 \ \ .453\,) .$$

In fact, in our case the rows of $Q_{\mathcal{E}}$ are all equal, so Proposition 3.20 holds not only in the limit, but rather for every particular path length $\ell > 0$.

As pointed out in Section 1.7.2, there are 113 outgoing edges from state 1 in $\mathcal{E}$ whose labels can be altered in their second bit (i.e., in the second merging bit) without violating the constraint. Therefore, the probability of allowing such a bit inversion is

$$\tfrac{113}{256} \cdot \pi_1 \approx .098 .$$

It follows that approximately once in every 10 codewords, on the average, a merging bit can be inverted. The law of large numbers (Theorem 3.21) can now guarantee an arbitrarily small deviation from this average with probability approaching one as the path length ℓ goes to infinity. $\square$

4.8 Spectral analysis of encoders

One important application of Markov chains on encoders is the spectral analysis of the output sequences that are generated by an encoder. Let S be a constrained system whose alphabet is a subset of the real field $\mathbb{R}$, and let $\mathcal{E} = (V, E, L)$ be an (S^q, n)-encoder. Each path of length ℓ in $\mathcal{E}$ generates a sequence of length ℓ over $\mathbb{R}^q$; yet, for the purpose of spectral analysis, we will regard those sequences as words of length $q\ell$ over $\mathbb{R}$. That is, we will be interested in the constrained system S' that is generated by a graph $\mathcal{E}'$ whose set of states is given by

$$V \cup \{u_{e,i}\}_{e \in E,\, 1 \le i < q} ,$$

and each edge e in $\mathcal{E}$ with a label $L(e) = w_1 w_2 \ldots w_q$ becomes a path γ_e in $\mathcal{E}'$ that takes the form

$$\sigma_{\mathcal{E}}(e) \stackrel{w_1}{\to} u_{e,1} \stackrel{w_2}{\to} u_{e,2} \stackrel{w_3}{\to} \ldots \stackrel{w_{q-1}}{\to} u_{e,q-1} \stackrel{w_q}{\to} \tau_{\mathcal{E}}(e) .$$

Every Markov chain $\mathcal{P}$ on $\mathcal{E}$ can be easily transformed into a Markov chain $\mathcal{P}'$ on $\mathcal{E}'$ with

$$\mathcal{P}'(\gamma_e) = \mathcal{P}(e) ;$$

indeed, define $\mathcal{P}'(e') = \mathcal{P}(e)$ for the first edge e' in γ_e, and let all the other edges along γ_e have conditional probability 1. Note that $\mathcal{P}'$ is an irreducible Markov chain on $\mathcal{E}'$ if and only if $\mathcal{P}$ is irreducible on $\mathcal{E}$. Also, the period of an irreducible $\mathcal{P}'$ is q times the period of $\mathcal{P}$.

Next we recall the definition of power spectral density from Problem 3.35 and apply it to $\mathcal{E}'$ and $\mathcal{P}'$, assuming that $\mathcal{P}'$ is an irreducible Markov chain on $\mathcal{E}'$. For simplicity, we will assume here that for each equivalence class C of the congruence relation on the states of $\mathcal{E}'$ we have

$$\mathsf{E}_{\mathcal{P}'} \left\{ L'(e') \,|\, \sigma_{\mathcal{E}'}(e') \in C \right\} = 0 .$$

Let

$$\mathbf{X} = X_{-\ell+1} X_{-\ell+1} \ldots X_0 X_1 \ldots X_\ell$$

denote a random word of length $2\ell{+}1$ taking values on S' with a probability distribution as induced by $\mathcal{P}'$. The autocorrelation of $\mathbf{X}$ is given by

$$R_{\mathbf{X}}(t) = \mathsf{E}_{\mathcal{P}'} \left\{ X_i X_{i+t} \right\} , \quad t = 0, \pm 1, \pm 2, \ldots, \pm \ell ;$$

by stationarity, $R_{\mathbf{X}}(t)$ does not depend on i (provided that $-\ell \le i, i{+}t \le \ell$). The power spectral density $f \mapsto \Psi_{\mathbf{X}}(f)$ of $\mathbf{X}$ is the two-sided Fourier transform of $R_{\mathbf{X}}(t)$; namely,

$$\Psi_{\mathbf{X}}(f) = \sum_{t=-\infty}^{\infty} R_{\mathbf{X}}(t) e^{-\jmath 2\pi t f} ,$$

where $\jmath = \sqrt{-1}$. The power spectral density can also be expressed as

$$\Psi_{\mathbf{X}}(f) = \lim_{\ell \to \infty} \frac{1}{2\ell{+}1} \mathsf{E}_{\mathcal{P}'} \left\{ |\Phi_{\mathbf{X}}(f)|^2 \right\} , \tag{4.2}$$

where $f \mapsto \Phi_{\mathbf{X}}(f)$ is the (two-sided) Fourier transform of $\mathbf{X}$, i.e.,

$$\Phi_{\mathbf{X}}(f) = \sum_{i=-\ell}^{\ell} X_i e^{-\jmath 2\pi i f}$$

(see Problem 3.35).

The value $\Psi_{\mathbf{X}}(0)$ is commonly referred to as the *dc component* of the power spectral density. It follows from (4.2) that

$$\Psi_{\mathbf{X}}(0) = \lim_{\ell \to \infty} \frac{1}{2\ell{+}1} \mathsf{E}_{\mathcal{P}'} \left\{ Y_\ell^2 \right\} ,$$

where

$$Y_\ell = \sum_{i=-\ell}^{\ell} X_i \ .$$

Now, $\mathsf{E}_{\mathcal{P}'}\{X_i\} = 0$ implies that $\mathsf{E}_{\mathcal{P}'}\{Y_\ell\} = 0$. Hence, we can guarantee that the dc component be zero by requiring that the digital sum variation (DSV) of $\mathbf{X}$ be bounded by a prescribed parameter B (see Section 1.5.4).

As mentioned in Section 1.7.3, the suppression of $\Psi_{\mathbf{X}}(f)$ for values of f near zero is desirable in optical recording. To this end, we could use in such applications a B–(d,k)-CRLL encoder (for, say, $(d,k) = (2,10)$), but the charge constraint B could result in a very complex encoder. An alternate solution was presented in Section 1.7.3, where we showed how the DSV can be reduced by bit inversions in the (d,k)-RLL sequence before it is precoded (see Example 1.11). This method can be applied in the EFM code through the occasional freedom in selecting the merging bits (see Example 4.12).

We note, however, that the reduction of the DSV through bit inversions requires, in principle, the knowledge of all future bits in the sequence. Clearly, this is impractical and, therefore, bit inversions are carried out based only on a limited look-ahead at the generated sequence.

Figure 4.8 shows the power spectral density (obtained by simulation) at the low-frequency range of the bipolar output of the 9-EFM(16) code after precoding. The merging bits in every encoded codeword are selected (when possible) as to minimize the absolute value of the sum of the symbols in the bipolar output. There are two curves in the figure, and both are shown in the dB scale. One curve has been generated without look-ahead, namely, it is assumed that the output sequence ends in the currently-encoded codeword; the second curve has been generated by looking ahead at two upcoming codewords.

The suppression of the low-frequency range in the curves of Figure 4.8 is still insufficient for optical recording applications. This was resolved in the compact disk by inserting three merging bits instead of two, resulting in the (proper) EFM code.

Another possible solution is relaxing the requirement that the modification of a codeword is limited only to inverting one of its first two bits. Such a relaxation allows to have codes such as the $(2,10)$-RLL encoder in [Roth00]. The encoder therein is a rate $8:16$ block decodable encoder with four states, 0, 1, 2–5, and 6–8 (with notations bearing the meaning as in the EFM code). Assuming a uniform distribution on the input bytes, almost every second input byte (on average) can be encoded into two different codewords that differ in the *parity* of the number of 1's in them (namely, for one codeword that number is even while it is odd for the other). The power spectral density of this encoder is shown in Figure 4.9, and the power spectral density of the 9-EFM(16) code is also included for comparison; in both encoders, the DSV reduction is obtained by looking ahead at two upcoming codewords (note that due to scaling, the power spectral density values in [Imm95b] and [Roth00] are shifted by 3dB compared to the figures herein). The curve in Figure 4.9 is very similar to the

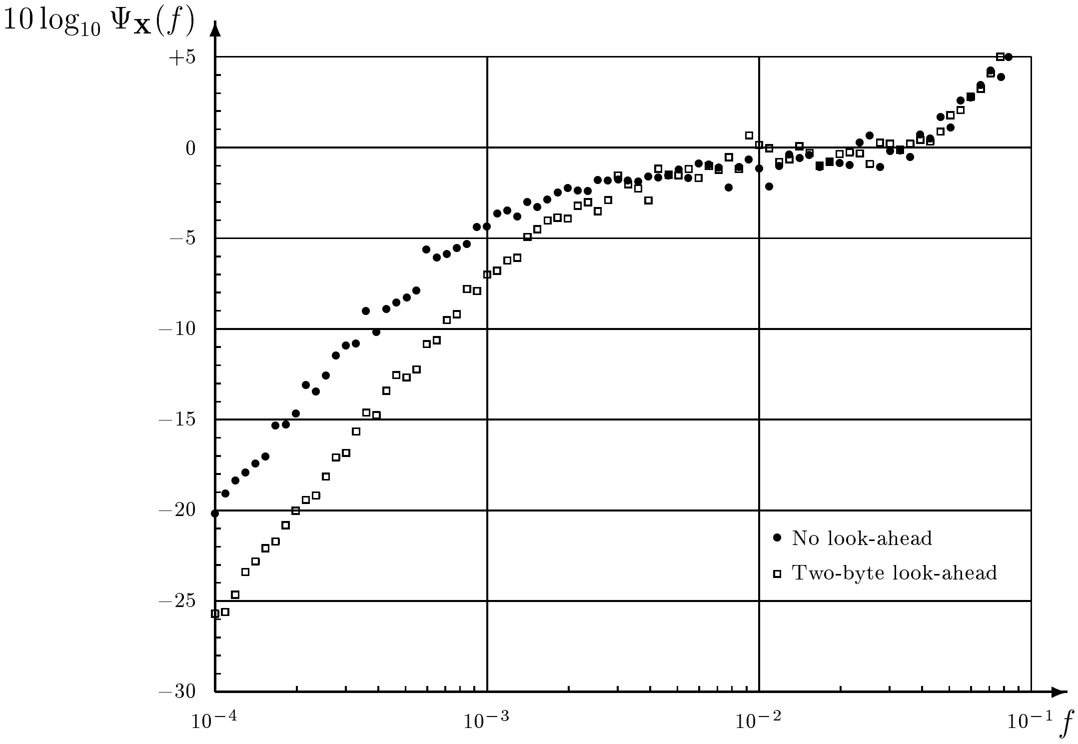

Figure 4.8: Power spectral density of the 9-EFM(16) code.

power spectral density of the EFMPlus code, which is used in the DVD (yet, the EFMPlus code is not block decodable).

Problems

Problem 4.1 Recall that the (d, k, s)-RLL constraint is a subset of the (d, k)-RLL constraint where the runlengths of 0's must be of the form $d + is$, with i a nonnegative integer.

Let S be the $(s-1, \infty, s)$-RLL constraint for a prescribed positive integer s.

1. Show that the capacity of S is $1/s$.

2. Construct a rate $1 : s$ block encoder for S.

Problem 4.2 (Enumerative coding) Let Σ be a finite alphabet and assume some ordering on the elements of Σ. Let $\mathcal{L}$ be a set of distinct words of length ℓ over Σ. The ordering over Σ induces the following lexicographic (dictionary) ordering over $\mathcal{L}$: given two words $\mathbf{w} = w_1 w_2 \ldots w_\ell$ and $\mathbf{z} = z_1 z_2 \ldots z_\ell$ in $\mathcal{L}$, we say that $\mathbf{w} < \mathbf{z}$ if there is $i \in \{1, 2, \ldots, \ell\}$ such that $w_j = z_j$ for $1 \leq j < i$ and $w_i < z_i$.

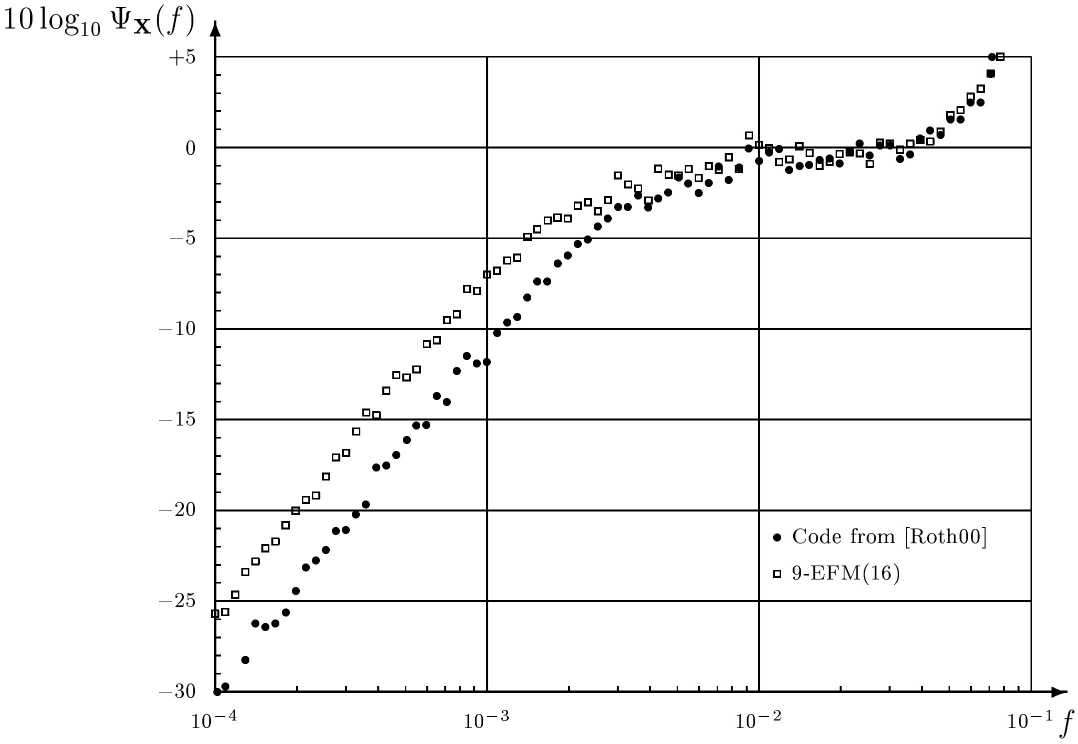

Figure 4.9: Improvement on the dc suppression compared to the 9-EFM(16) code.

For a word $\mathbf{w} \in \mathcal{L}$, denote by $\mathrm{Ind}_{\mathcal{L}}(\mathbf{w})$ the index of $\mathbf{w}$ in $\mathcal{L}$, according to the induced lexicographic ordering (starting with zero as the smallest index). Also, for a word $w_1 w_2 \ldots w_i$ of length $i \le \ell$ over Σ, denote by $N_{\mathcal{L}}(w_1, w_2, \ldots, w_i)$ the number of words in $\mathcal{L}$ whose prefix of length i is given by $w_1 w_2 \ldots w_i$.

1. Show that for every word $\mathbf{w} = w_1 w_2 \ldots w_\ell \in \mathcal{L}$,

$$\mathrm{Ind}_{\mathcal{L}}(\mathbf{w}) = \sum_{i=1}^{\ell} \sum_{a \in \Sigma \,:\, a < w_i} N_{\mathcal{L}}(w_1, w_2, \ldots, w_{i-1}, a)$$

   (a sum over an empty set is defined to be zero).

2. Given an integer r in the range $0 \le r < |\mathcal{L}|$, show that the algorithm in Figure 4.10 produces the word $\mathbf{w} \in \mathcal{L}$ such that $\mathrm{Ind}_{\mathcal{L}}(\mathbf{w}) = r$.

3. Let $G = (V, E, L)$ be a deterministic graph and assume an ordering on the range of $L : E \to \Sigma$. Given two states $u, v \in V$ and a positive integer ℓ, let $\mathcal{L} = \mathcal{L}_G(u, v; \ell)$ be the set of all words of length ℓ that can be generated from state u to state v in G.

   Write an efficient algorithm for implementing an enumerative coder: the algorithm accepts as input the quadruple (G, u, v, ℓ) and an integer r in the range $0 \le r < |\mathcal{L}_G(u, v; \ell)|$, and produces as output the word $\mathbf{w} \in \mathcal{L} = \mathcal{L}_G(u, v; \ell)|$ such that $\mathrm{Ind}_{\mathcal{L}}(\mathbf{w}) = r$.

```
s ← r;
i ← 1;
while (i ≤ ℓ)   {
     a ← smallest element in Σ;
     while (s ≥ N_L(w_1, w_2, ..., w_{i-1}, a))   {
          s ← s − N_L(w_1, w_2, ..., w_{i-1}, a);
          increment a to the next element in Σ;
     }
     w_i ← a;
     i ← i + 1;
}
return w = w_1 w_2 ... w_ℓ;
```

Figure 4.10: Enumerative coding.

Write also an algorithm for implementing the respective enumerative decoder.

Problem 4.3 Explain how an (S^ℓ, n^ℓ)-encoder $\mathcal{E} = (V, E, L)$ can be transformed into an (S, n)-encoder $\mathcal{E}'$. Write an upper bound on the number of states of $\mathcal{E}'$ as a function of $|V|$, n, and ℓ.

Hint: Start with the Moore co-form of $\mathcal{E}$. Then replace the outgoing edges from each state by a respective tree.

Problem 4.4 Let S be the constrained system generated by the graph G in Figure 2.22. Since $\log 3 > 3/2$ then, by Shannon's coding theorem, there is a positive integer ℓ such that there exists a rate $(3\ell) : (2\ell)$ block encoder for S; i.e., there is an $(S^{2\ell}, 2^{3\ell})$-encoder with only one state.

The goal of this question is finding the smallest value of the integer ℓ for which such a block encoder exists.

1. Let ℓ be a positive integer for which there is a rate $(3\ell) : (2\ell)$ block encoder $\mathcal{E}$ for S. Show that there is a state u in G such that all the codewords in $\mathcal{E}$ (of length 2ℓ over the alphabet Σ of S) are generated by cycles in G that originate and terminate in state u.

2. Show that the number of cycles of length 2ℓ in G that originate and terminate in state B equals
$$\frac{1}{5} \cdot \left(2 \cdot 9^\ell + 3 \cdot (-1)^\ell\right) .$$
Hint: $A_{G^2} = P \Lambda P^{-1}$, where Λ is a diagonal matrix; what is P?

3. Obtain expressions, similar to the one in 2, for the number of cycles of length 2ℓ in G that originate and terminate in—

   (a) state A;

(b) state C.

4. Apply the results in 1–3 to find the smallest value of the integer ℓ for which there is a rate $(3\ell) : (2\ell)$ block encoder for S.

Problem 4.5 Let S be an irreducible constrained system with finite memory $\mathcal{M}$ and let G be the Shannon cover of S. Assume there exists a block (S^ℓ, n)-encoder $\mathcal{E}$ (with one state), where n and ℓ are positive integers and $\ell \geq \mathcal{M}$. Denote by $\mathcal{L}$ the set of n words of length ℓ that label the edges of $\mathcal{E}$. Show that if a word $\mathbf{w}$ in $\mathcal{L}$ is generated in G by a path that terminates in state u, then $\mathcal{L} \subseteq \mathcal{F}_G(u)$; recall that $\mathcal{F}_G(u)$ denotes the set of words that can be generated in G by paths originating in state u.

Problem 4.6 Let S be the constrained system presented by the graph G in Figure 4.11.

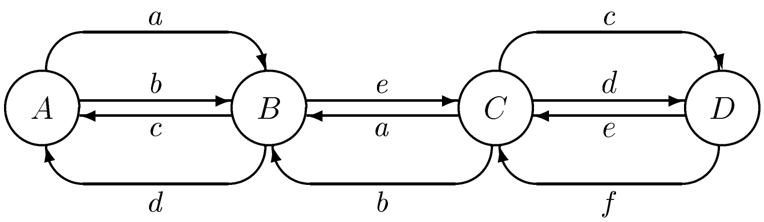

Figure 4.11: Graph G for Problem 4.6.

1. What is the period of G?
2. What is the memory of G?
3. Compute the capacity of S.
4. Obtain the entries of the adjacency matrix of $G^{2\ell}$ as expressions in ℓ. Simplify those expressions as much as possible.

   Hint: Find a diagonal matrix Λ and a nonsingular matrix P such that $A_{G^2} = P\Lambda P^{-1}$.
5. For states u and v in G and a positive integer ℓ, let $N(u, v, \ell)$ denote the number of distinct words of length 2ℓ that can be generated in G both from state u and state v; that is,

$$N(u, v, \ell) = |\mathcal{F}_G(u) \cap \mathcal{F}_G(v) \cap \Sigma^{2\ell}| \ .$$

   For every two states in G, obtain the values of $N(u, v, \ell)$ as expressions in ℓ. Simplify those expressions as much as possible.
6. Based on 4, and 5, and Problem 4.5, show that there is no block $(S^{2\ell}, 2^{3\ell})$-encoder (with one state) for any positive integer ℓ.
7. Construct a rate $2 : 2$ block encoder for S.

Problem 4.7 Let $\mathcal{E}$ be a tagged (S, n)-encoder and let $A_{\mathcal{E}*\mathcal{E}}$ be the adjacency matrix of $\mathcal{E} * \mathcal{E}$ (the input tags are ignored when constructing the fiber product). Denote by $T_{\mathcal{E}*\mathcal{E}}$ the $|V_{\mathcal{E}}|^2 \times |V_{\mathcal{E}}|^2$ matrix with rows and columns indexed by the states of $\mathcal{E} * \mathcal{E}$ and entries defined as follows: for every $u, u', v, v' \in V_{\mathcal{E}}$, the entry $(T_{\mathcal{E}*\mathcal{E}})_{\langle u,u'\rangle,\langle v,v'\rangle}$ equals the number of (ordered) pairs of edges $u \to v$ and $u' \to v'$ in $\mathcal{E}$ that have the same label yet are assigned distinct input tags. Show that $\mathcal{E}$ is (m, a)-sliding-block decodable if and only if

$$A_{\mathcal{E}*\mathcal{E}}^{\mathsf{m}} \, T_{\mathcal{E}*\mathcal{E}} \, A_{\mathcal{E}*\mathcal{E}}^{\mathsf{a}} = 0 \; .$$

Problem 4.8 Prove Proposition 4.10.