An Introduction to Coding for Constrained Systems

Brian H. Marcus[1] Ron M. Roth[2] Paul H. Siegel[3]

Fifth Edition (October 2001)

[1]IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA.
[2]Computer Science Department, Technion — Israel Institute of Technology, Haifa 32000, Israel.
[3]Department of Electrical and Computer Engineering 0407, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA.

Contents

Preface

In most data recording systems and many data communication systems, some sequences are more prone to error than others. Thus, in order to reduce the likelihood of error, it makes sense to impose a constraint on the sequences that are allowed to be recorded or transmitted. Given such a constraint, it is then necessary to encode arbitrary user sequences into sequences that obey the constraint. In this text, we develop a theory of constrained systems of sequences and encoder design for such systems. As a part of this, we include concrete examples and specific algorithms.

We begin in Chapter 1 with a description of several applications of constrained systems of sequences in data recording and data communications. We also give a rough description of the kinds of encoders and corresponding decoders that we will consider. Chapter 2 contains the basic mathematical concepts regarding constrained systems of sequences. Many of these concepts are closely related to fundamental notions in computer science, such as directed graphs and finite-state machines. In Chapter 3, we develop the notion of capacity from three different points of view: combinatorial, algebraic and probabilistic. In the course of doing so, we show how to compute the capacity of an arbitrary constrained system. In Chapter 4 we give a general introduction to finite-state encoders and sliding-block decoders. This includes Shannon's fundamental result that relates the maximal code rate of an encoder for a constrained system to the capacity of a constrained system. In Chapter 5, we develop the state-splitting algorithm, which gives a rigorous procedure for designing finite-state encoders for constrained systems.

There are many other ways of designing encoders for constrained systems. In Chapter 6, we outline some of these techniques and briefly explain how they are related to the state-splitting algorithm. In Chapter 7, we focus on complexity issues and finite procedures relating to encoders and decoders for constrained systems. For instance, we give bounds on the number of states in the smallest encoder for a given constrained system at a given rate. In Chapter 8 we begin with a very brief introduction to error-correction coding, in particular Reed-Solomon codes. We then discuss methods of concatenating constrained codes with error-correction codes. Finally, in Chapter 9 we consider codes which simultaneously have error-correction and constrained properties. These include spectral null codes and forbidden list codes which eliminate likely error events for specific channels. We also extend classical bounds for error-correction codes to combined error-correction–constrained codes.

This text is intended for graduate students and advanced undergraduates in electrical engineering, mathematics and computer science. The main prerequisites are elementary linear algebra and elementary probability at the undergraduate level. It is also helpful, but not necessary, to have had some exposure to information theory (for Chapter 3) and error-correction coding (for Chapters 8 and 9). While Chapter 1 includes discussion of several engineering applications, the remainder of the text develops the theory in a more formal mathematical manner.

We are happy to acknowledge that earlier versions of this text were used as notes for courses on Constrained Systems. One of these versions appeared as Chapter 20 of the Handbook of Coding Theory [PH98]. We are grateful to students, as well as many of our colleagues in industry and universities, for helpful suggestions on these earlier versions.

Chapter 1

Channels and Constraints

The purpose of this chapter is to motivate, via several engineering examples, the subject of coding for constrained systems. Beyond this chapter the text will take on a more mathematical flavor.

1.1 Magnetic Recording

In this section we give a very brief introduction to magnetic recording. For more background on this subject, the reader is referred to Section 1.6 and the book [WT99].

The essential components of a magnetic recording system are a recording head, a read-head and a recording medium, such as a rotating magnetic disk. The disk is divided into concentric tracks. To write a string of data along a track, the recording head is positioned above the track, and current sent through the head magnetizes the track in one of two directions called magnetic polarities. This process is illustrated in Figure 1.1. A clock runs at a constant bit period of T seconds, and at each clock tick (i.e., at each multiple of T), the recording head has the opportunity to change the polarity on the disk: a 1 is recorded by changing the direction of the current, while a 0 is recorded by *not* changing the direction of the current (i.e., doing nothing). In this way, a 1 is represented as a transition in magnetic polarity along a data track, while a 0 is represented as an absence of such a transition. So, we can think of any binary sequence as a sequence of transitions/no-transitions; when we want to emphasize that we are thinking of a binary sequence in this way, we will call it a *transition sequence*.

In reading mode, the read-head, positioned above a track on the rotating disk, responds to a magnetic transition by an induced voltage; naturally, the absence of a transition produces no voltage response in the read-head. A bit cell is a window of length T centered on a clock tick. When a sufficiently high (positive or negative) voltage peak is detected within a bit

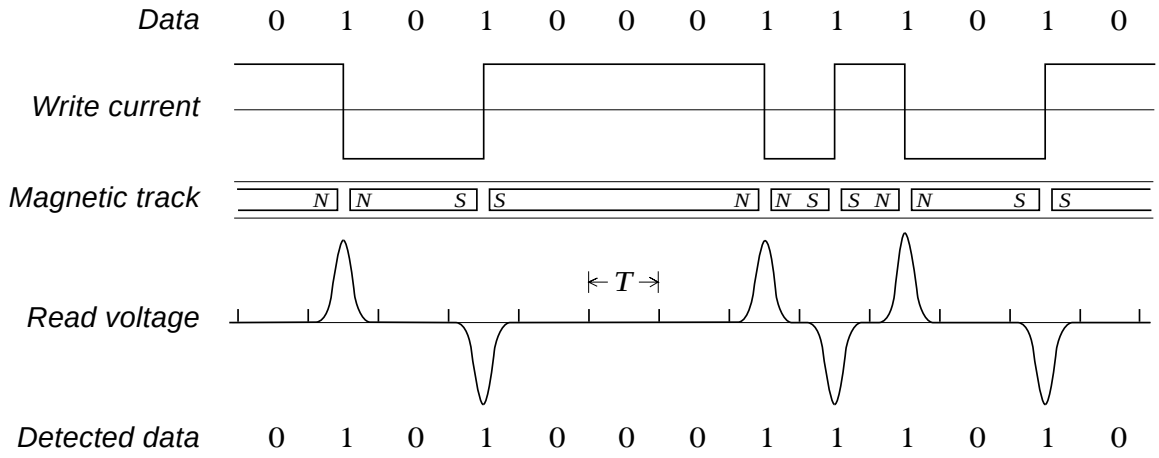


Figure 1.1: Digital magnetic recording.

cell, a 1 is declared; otherwise, a 0 is declared. This scheme is known as *peak detection*. Note that as shown in Figure 1.1, the readback voltages corresponding to successive transitions are of opposite signs; however, in most implementations the peak detector ignores the signs. We also remark that due to the presence of noise, the readback voltage signal in reality is not as smooth as Figure 1.1 might suggest.

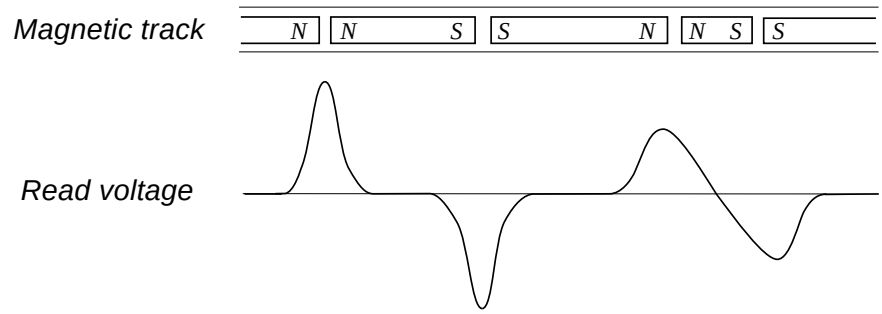


Figure 1.2: Inter-symbol interference.

Now, if successive 1's are too close together, then the voltage responses corresponding to the magnetic transitions may interfere with one another; this is known as *inter-symbol interference*. From Figure 1.2, which shows a reconstruction of the received signal with interference, it is evident that such interference may degrade the signal so that either one or both transitions are missed. In such a case, a recorded 1 will be mis-read as a 0; perhaps, even worse, the bit cell corresponding to a transition may be incorrectly identified so that

a recorded 01 will be mis-read as 10; such an error is called a *peak shift* or *bit shift*. The likelihood of these kinds of errors can be reduced if 1's are separated sufficiently far apart, or equivalently if any two 1's are separated by a sufficiently long run of 0's.

Determination of the correct runlength of 0's between two successive 1's depends critically on accurate clocking. But accurate clocking can well be compromised by various imperfections in the recording system, such as variations in speed of the rotating disk. Thus, clocking needs to be adjusted periodically via a timing control scheme, which adjusts the clock in response to identification of peaks in the received signal. For instance, if a peak is supposed to occur at exactly the mid-point of the bit cell, but instead occurs near the end of the bit cell, then the next clock tick should be delayed. But during a long run of 0's, the ideal received signal will not contain any peaks. Hence, for timing control, it is desirable to avoid long runs of 0's.

This discussion suggests that it may be helpful to impose constraints on the binary sequences that are actually recorded.

1.2 Classical runlength constraints

The (d, k)-*runlength-limited (RLL) constraint* is described as follows.

Let d and k be integers such that $0 \leq d \leq k$. We say that a finite length binary sequence $\mathbf{w}$ satisfies the (d, k)-RLL constraint if the following two conditions hold:

- the runs of 0's have length at most k (the k-*constraint*), and —
- the runs of 0's between successive 1's have length at least d (the d-*constraint*); the first and last runs of 0's are allowed to have lengths smaller than d.

According to the discussion in Section 1.1, for binary transition sequences the d-constraint reduces the effect of inter-symbol interference and the k-constraint aids in timing control.

Many commercial systems, such as magnetic tape recording systems, use the constraint $(d, k) = (1, 7)$ or $(d, k) = (2, 7)$. An example of a sequence satisfying the $(d, k) = (2, 7)$-RLL constraint is

$$\mathbf{w} = 00100001001000000010\ .$$

Other recording standards include the $(1, 3)$-RLL constraint, which can be found in flexible disk drives, and the $(2, 10)$-RLL constraint, which appears in the compact disk (CD) [Imm91, Ch. 2] and the digital versatile disk (DVD) [Imm95b].

The set of all sequences satisfying a (d, k)-RLL constraint is conveniently described by reading the binary labels of paths in the finite directed labeled graph in Figure 1.3. The

graph consists of a finite collection of states (the numbered circles) and a finite collection of labeled directed edges. For each $i = 0, 1, \ldots, k-1$, there is an edge labeled 0 from state i to state $i+1$. For each $j = d, d+1, \ldots, k$, there is an edge labeled 1 from state j to state 0. A path in the graph is a sequence of edges such that the initial state of each edge is the terminal state of the preceding edge.

It can be verified that a sequence $\mathbf{w}$ satisfies the (d, k)-RLL constraint if and only if there is a path in the graph whose edge labeling is $\mathbf{w}$. For this reason, we say that the graph is a *presentation* of (or *presents*) the RLL constraint.

Any set of sequences that can be presented by a labeled graph in this way is called a *constraint* or *constrained system*. We will have much more to say about graph presentations in Chapter 2.

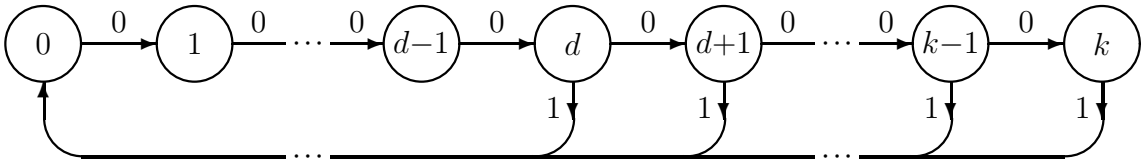


Figure 1.3: Graph presentation of the (d, k)-RLL constraint.

We may extend the class of (d, k)-RLL constraints to include the case where $k = \infty$; namely, there is no upper bound on the length of runs of 0's. In such a case, the constraint is described by the labeled graph in Figure 1.4.

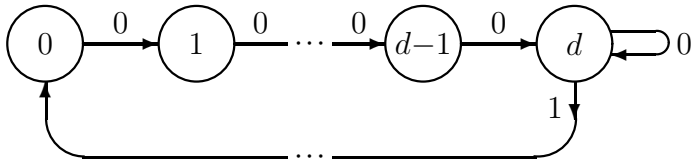


Figure 1.4: Graph presentation of the (d, ∞)-RLL constraint.

A graph presentation, such as Figure 1.3, represents a constraint by showing which sequences are allowed. An alternative representation specifies which sequences are forbidden. For instance, the $(1, 3)$-RLL constraint is the set of sequences that do not contain any word of the list

$$\{11, 0000\}$$

as a sub-string of contiguous symbols. Such a list is called a list of *forbidden words*. The reader may check that every (d, k)-RLL constraint can be defined by a very simple list of forbidden words (Problem 1.3).

1.3 Coding for Channels

In the abstract a channel can be viewed as a "black box" with inputs and outputs. The inputs represent information that is transmitted through the box. The outputs are supposed to faithfully represent the inputs. However, distortions in the channel can adversely affect the output. For this reason, coding is applied to protect the inputs.

One usually thinks of a channel as a communications system, in which information is sent from one point in space to another. Examples of communications systems include telephones, cellular phones, digital subscriber lines and deep space communications. But recording systems, such as the magnetic recording system described in Section 1.1, can also be viewed as channels. The main difference between recording channels and communications channels is that space is replaced by time. That is, in a recording channel, information is recorded at one point in time and retrieved at a later point in time [Berl80].

Current recording applications require storage devices to have very high immunity against errors. On the other hand, the ever-growing demand for storage forces the designers of such devices to write more data per unit area, thereby making the system less reliable. This is manifested in the effects of inter-symbol interference, inaccurate clocking, and random noise.

A *constrained encoder*, also known as a *modulation encoder* or *line encoder* , transforms arbitrary user data sequences into sequences, also called codewords, that satisfy a given constraint, such as an RLL constraint. Naturally, such an encoder will have a corresponding decoder called a *constrained decoder*. We loosely use the term *constrained code* to refer to the constrained encoder and decoder together. In the most general terms, the purpose of a constrained code is to improve the performance of the system by matching the characteristics of the recorded signals to those of the channel; the recorded signals are thereby constrained in such a way as to reduce the likelihood of error.

In addition to constrained coding, an error-correction code (ECC) may be used to protect the data against random noise sources. An ECC must provide an encoder (and corresponding decoder) which translates arbitrary user data sequences into codewords. A good ECC has the property that any two distinct codewords must differ enough so as to be distinguishable even after being subjected to a certain amount of channel noise. While both error-correction coding and constrained coding have been active for fifty years, the former enjoys much greater notoriety. There are many excellent textbooks on ECC, such as those by Berlekamp [Berl84], Blake and Mullin BM, Blahut [Blah83], Lin and Costello [LinCo83], MacWilliams and Sloane [MacS77], McEliece [Mcl77], Peterson and Weldon [PW72], Pless [Pl89], and Wicker [Wic95]. For an extensive summary of what is known in the area, refer to the Handbook of Coding Theory [PH98].

What is the difference between an error-correction code and a constrained code? One difference is that the "goodness" of an error-correction code is measured by how the different codewords relate to one another (e.g., in how many bit locations must any two codewords

differ?), whereas the "goodness" of a constrained code is measured by properties of the individual codewords (e.g., how well does each codeword pass through the channel?).

On the other hand, this distinction is not hard and fast. Clearly, if an error-correction code is to have any value at all, then its codewords cannot be completely arbitrary and therefore must be constrained. Conversely, in recent years, there has been a great deal of interest in constrained codes that also have error-correction properties. Such developments have contributed to a blurring of the lines between these two types of coding. Nevertheless, each subject has its own emphases and fundamental problems that are shaped by the distinction posed in the preceding paragraph.

Figure 1.5 shows the arrangement of error-correction coding and constrained coding in today's recording systems. User messages are first encoded via an error-correction encoder

Figure 1.5: Coding for recording channels: standard concatenation.

and then via a constrained encoder. The resulting message is recorded on the channel. At a later time, the (possibly corrupted) version of this message is retrieved and decoded, first by the constrained decoder and then by the error-correction decoder.

This arrangement places the constrained encoder–decoder pair nearer to the channel than the error-correction encoder–decoder pair. This makes sense since otherwise the constraints imposed by the constrained encoder might be destroyed by the error-correction encoder. On the other hand, in this arrangement the ECC properties imposed by the ECC encoder may be weakened. For this reason, it may be desirable to reverse the order in which the coders are concatenated, although it is not at all obvious how to do this. These issues will be discussed in Chapter 8.

Ideally the messages recorded on a channel should be determined by a single code that has both "pairwise" error-correction properties as well as "individual" constraint properties. Such codes will be studied in Chapter 9. For now, we focus only on constrained codes.

1.4 Encoding and decoding for constrained systems

In this section, we give a rough description of the kinds of encoders and decoders that are used for constrained systems.

The encoder typically takes the form of a finite-state machine, shown schematically in

Figure 1.6. A *rate* $p : q$ *finite-state encoder* accepts an arbitrary input block of p user bits and generates a codeword of length q—referred to as a q-*codeword*—depending on the input block and the current internal state of the encoder. The sequences obtained by concatenating the q-codewords generated by the encoder must satisfy the constraint. There are of course only finitely many states.

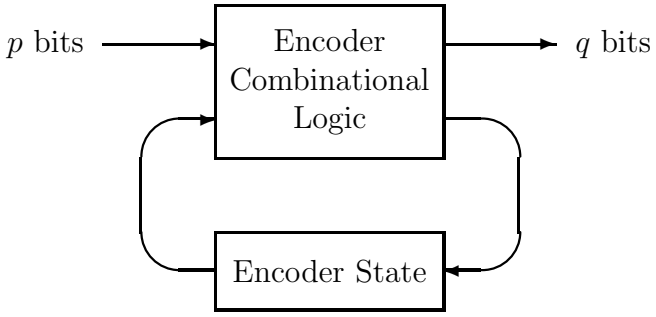


Figure 1.6: Finite-state encoder schematic.

Example 1.1 Figure 1.7 depicts a rate 2 : 3 encoder for the $(0, 1)$-RLL constrained system [Sie85a]. There are two states, A and B. Each state has exactly four outgoing edges. Each edge has two labels: a 2-bit input label and a 3-bit output label. To make a better distinction between input and output labels, we will refer to the former as input *tags*. Note that at each state, the set of four outgoing edges carry all possible 2-bit input tags exactly once.

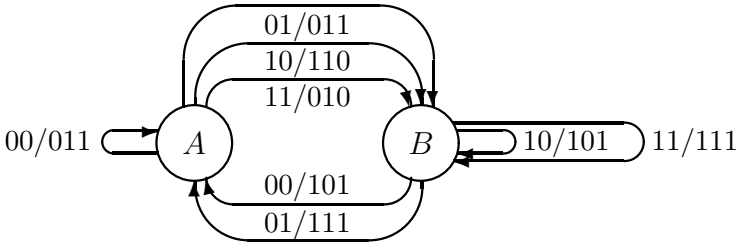


Figure 1.7: Rate 2 : 3 two-state encoder for $(0, 1)$-RLL constrained system.

One encodes as follows. First, fix an an initial state, say state A. To encode an arbitrary 2-bit input uv, first find the unique outgoing edge e, with input tag uv, from state A; the encoded 3-codeword is the output label of edge e. To encode the next 2-bit input, execute the same procedure starting from the terminal state of e. For instance, starting at state A, the sequence

$$01\ 01\ 00\ 10\ 10\ 01\ 00\ 00$$

encodes to

$$011\ 111\ 011\ 110\ 101\ 111\ 011\ 011\ ,$$

with corresponding state sequence:

$$A\ B\ A\ A\ B\ B\ A\ A\ A\ .$$

The encoded sequences do indeed satisfy the $(0, 1)$-RLL constraint, i.e., one can never see two consecutive 0's; this follows from the observations: (1) one can never see two consecutive 0's on an edge output label, and (2) whenever an edge carries an output label that ends in 0, the following output labels all begin with 1. □

The encoders that we construct should be decodable. One type of decoder that we consider is a *state-dependent decoder* which accepts, as input, a q-codeword and generates a length-p block of user bits depending on the internal state, as well as finitely many upcoming q-codewords. Such a decoder will invert the encoder when applied to valid code sequences, effectively retracing the state sequence followed by the encoder in generating the codeword sequence. However, when the code is used in the context of a noisy channel, the state-dependent decoder may run into a serious problem. The noise causes errors in the detection of the codeword sequences, and the decoder must cope with erroneously detected sequences, including sequences that could not be generated by the encoder. It is generally very important that the decoder limit the propagation of errors at the decoder output resulting from such an error at the decoder input. Unfortunately, an error at the input to a state-dependent decoder can cause the decoder to lose track of the encoder state sequence, with no guarantee of recovery and with the possibility of unbounded error propagation.

Example 1.2 Consider the (admittedly artificial) rate 1 : 1 encoder depicted in Figure 1.8.

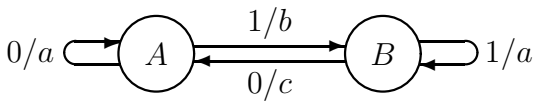


Figure 1.8: Encoder susceptible to unbounded decoder error propagation

If we select state A as the initial state, then the sequence $000000\cdots$ encodes to the sequence $aaaaaa\cdots$. If a single channel error corrupts the first a to b, then the state-dependent decoder will receive the sequence $baaaaa\cdots$ and decode to $111111\cdots$. So, a single channel error at the beginning may cause many decoding errors. □

The decoder therefore needs to have additional properties. Specifically, any symbol error at the decoder input should give rise to a limited number of bit errors at the decoder output. A *sliding-block decoder* makes a decision on a given received q-codeword on the basis of the local context of that codeword in the received sequence: the codeword itself, as well as a

fixed number m of preceding codewords and a fixed number a of later codewords. Figure 1.9 shows a schematic diagram of a sliding-block decoder. It is easy to see that a single error at the input to a sliding-block decoder can only affect the decoding of codewords that fall in a "window" of length at most m+a+1 codewords.

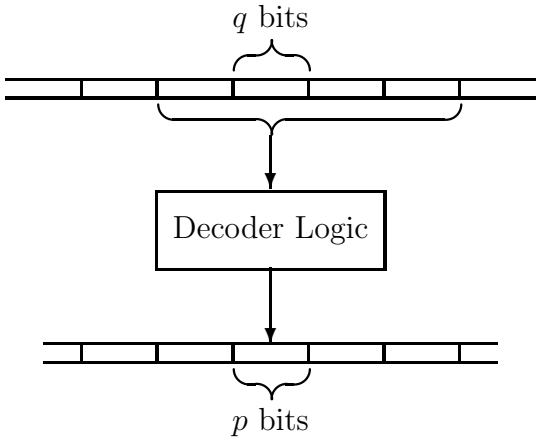


Figure 1.9: Sliding-block decoder schematic.

Example 1.3 Table 1.1 defines a sliding-block decoder for the encoder in Figure 1.7. Entries marked by "—" in the table do not affect the value of the decoded input tag. For

(current codeword)	(next codeword)	(decoded input)
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 1.1: Sliding-block decoder for encoder in Figure 1.7.

example, according to Table 1.1, the codeword 010 decodes to 11; the codeword 011 decodes to 01 if it is followed by 101 or 111, and it decodes to 00 if it is followed by 010, 011 or 110.

Note that this decoder really does invert the encoding in Figure 1.7. For example, the 3-codeword 010 occurs on exactly one edge as an output label. Since the corresponding input tag is 11, the codeword 010 decodes to 11. While the 3-codeword 011 occurs as an output label on two different edges, these edges end at different states. The edge ending at state B can be followed only by edges whose output labels are the codewords 101 or 111. This

is why the codeword 011 decodes to 01 if it is followed by 101 or 111. Similarly, the edge ending at state A can can be followed only by edges whose output labels are the codewords 010, 011 or 110. This is why 011 decodes to 00 if it is followed by 010, 011 or 110.

Consider an application of the decoder in Table 1.1 to the following sequence of eight 3-codewords

$$011\ 111\ 011\ 110\ 101\ 111\ 011\ 011\ ;$$

this is the same sequence that was encoded in Example 1.1. It can be easily verified that the decoder will recover correctly the first seven respective 2-bit inputs

$$01\ 01\ 00\ 10\ 10\ 01\ 00\ ,$$

while the eighth 2-bit input can be either 00 or 01. Indeed, since the last 3-codeword is 011, the unique recovery of the eighth 2-bit input is possible only when the next (ninth) 3-codeword becomes available.

We point out that the inability to recover the eighth 2-bit input without additional information is not a limitation of the decoder, but rather a consequence of the structure of the encoder of Figure 1.7: the first seven 2-bit inputs lead the encoder to state A, from which there are two outgoing edges labeled 011. So, unless more information is provided (e.g., in the form of a ninth 3-codeword) there is no way to tell which edge generated the eighth 3-codeword. □

We emphasize that it is critical for both the encoder and decoder to have knowledge of the correct framing of sequences into 2-bit inputs and 3-codewords. For the former, this is a pretty safe assumption since the input comes from the well-controlled computer CPU. For the latter, this is not so safe. As discussed in Section 1.1 inaccurate clocking could cause the system to lose or gain a bit and therefore lose the framing into codewords. However, the k-constraint itself, as well as the coding/synchronization scheme to be discussed in Section 1.5.2, help to ensure that the decoder has access to the correct framing.

In the literature, the term "code rate" can refer to either the pair of numbers, $p : q$, or the ratio, p/q, between these numbers. This is a genuine ambiguity since $p : q$ is more specific than p/q. For instance, a rate 2:3 encoder is literally different from a rate 4:6 encoder. However, the two uses of the term can usually be distinguished from context. In this text we continue tradition by using the term "code rate" to refer to both $p : q$ and p/q.

Shannon proved [Sha48] that the rate p/q of a constrained encoder cannot exceed a quantity, now referred to as the Shannon capacity, that depends only upon the constraint (see Chapter 3). He also gave a non-constructive proof of the existence of codes at rates less than, but arbitrarily close to, the capacity. Therefore, as well as the issues of decodability mentioned above, any encoding/decoding scheme will be measured by the proximity of its rate, p/q, to the Shannon capacity.

Chapters 4, 5, and 6 deal with general methods of constructing encoders for constrained systems. In particular, in Chapter 5 we present an algorithm for transforming a graph presentation of any given constrained system into a finite-state encoder. Still, the design of a specific encoder for a specific constrained system usually benefits from a good deal of *ad hoc* experimentation as well.

1.5 Examples of constraints

In Sections 1.1 and 1.2, we motivated and described runlength limited constraints. In this section, we describe several other examples of constraints that arise in recording and communications applications.

1.5.1 Anti-whistle constraints

In some applications it is desirable to impose a limit on the maximum length of a particular periodic sequence. If we view a periodic sequence as a "pure tone" or "whistle," then such a constraint can be viewed as an "anti-whistle" constraint.

For example, the k-constraint limits the length of the sequence $000000\cdots$, a sequence of period 1. One may wish to the limit the length of sequences of period 2: $101010\cdots$ or $010101\cdots$, as in the following example.

Example 1.4 Figure 1.10 presents the constrained system which simultaneously satisfies

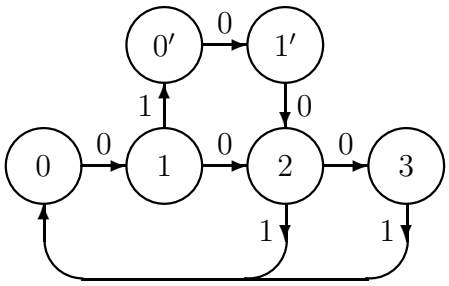


Figure 1.10: Limiting the length of the sequences $010101\cdots$ and $101010\cdots$.

the $(1,3)$-RLL constraint and limits the maximum number of repetitions of 10 or repetitions of 01 to two. □

Constraints of this type have been used in a particular version of the wireless infrared channel for data communication among mobile devices such as laptop computers. In this

application, data is transmitted by infrared pulses, with a 1 in a bit cell indicating the presence of a pulse and a 0 indicating the absence of a pulse. The *duty cycle* of a sequence is simply the total number of 1's in the sequence divided by the length of the sequence. So, a sequence with a high duty cycle will transmit more pulses per unit time than a sequence with a low duty cycle. In order to conserve power, it is therefore desirable to restrict the length of transmitted sequences with high duty cycle.

As in magnetic recording, an RLL constraint is imposed in order to reduce inter-symbol interference and aid in timing control. If the constraint is a $(1, k)$-RLL constraint, then for long sequences the maximum duty cycle is roughly 50% and is achieved only by the sequences $010101\cdots$ and $101010\cdots$. Thus, it is desirable to limit the maximum length of such sequences, as in Example 1.4.

Recently, the Infrared Data Association adopted the Very Fast Infrared (IrDA-VFIr) standard which includes as part of the format a constraint that simultaneously imposes the $(1, 13)$-RLL constraint and limits the maximum number of repetitions of 10 or repetitions of 01 to five [HHH00]. Anti-whistle constraints are also used to aid in timing and gain control algorithms in magnetic recording [Berg96, p.171].

1.5.2 Synchronization constraints

Recall from the end of Section 1.2 that RLL constraints can be defined by lists of forbidden words. The example in this section is most easily defined in this way.

Example 1.5 Consider the constraint consisting of all sequences that do not contain any of the following four 24-bit words

$$010\mathbf{w}_0 01\mathbf{w}_1\ , \quad 010\mathbf{w}_0 10\mathbf{w}_1\ , \quad 001\mathbf{w}_0 01\mathbf{w}_1\ , \quad 001\mathbf{w}_0 10\mathbf{w}_1\ ,$$

where $\mathbf{w}_0 = 000000$ and $\mathbf{w}_0 = 0101010101010$. The first word consists of 01 followed by eight 0's followed by seven repetitions of 10. The second and third words each differ from the first word by a single bit shift. And the fourth word differs from the first word by two bit shifts. We leave it to the reader to construct a (finite) graph presentation of this constraint (Problem 1.12). □

Such a constraint can be useful for synchronization. We describe this as follows.

Recorded data is usually grouped together into sectors consisting of a large number of bits. When a sector is addressed for retrieval, the read-head will jump to the beginning of the sector. However, it is difficult to accurately position on the first bit of the sector. Moreover, in the process of moving to the beginning of a new sector, the clock may be kicked out of synchronization. For this reason, the sector begins with special patterns that help the

clock to reach synchronization quickly, to determine the beginning of encoded data and to determine the correct framing into q-codewords (if a rate $p : q$ code is used) required by a constrained decoder. One of these patterns is called a *sync mark* (see Figure 1.11).

Figure 1.11: Sync marks.

The position of the sync mark is determined by a correlator which examines the output of the read-head. If the sync mark is allowed to occur within the actual encoded data, then one may read a "false sync," and this could very well cause decoding errors. Thus, it is desirable to encode the data so that the sync mark is forbidden from the actual encoded data. Moreover, because of noise in the system it is also helpful to forbid the most likely perturbations of the sync mark. In one particular magnetic tape system, the first word in Example 1.5 above was used as a sync mark and the other three words were the most likely (because of bit shifts) perturbations [AJMS99]. In that system, in addition to forbidding the words in Example 1.5, a $(1, 7)$-RLL constraint as well as an anti-whistle constraint, were imposed simultaneously as well.

Finally, we mention that synchronization is also important in communications systems [Sklar88]. However, for such systems, synchronization is often dealt with using signal processing rather than constrained coding techniques.

1.5.3 Multiple-spaced runlength constraints

We next consider the class of multiple-spaced RLL constraints. These constraints are characterized by parameters (d, k, s), where d and k again define the minimum and maximum allowable runlengths of 0's, and s indicates that the runlengths of 0's must be multiples of s. In particular, $s = 2$ simply means that all runlengths of 0's must be even.

Example 1.6 A graph presentation for the $(2, 18, 2)$-RLL constraint is shown in Figure 1.12. □

The $(d, k, 2)$-RLL constraints were originally investigated by Funk [Funk82] in the context of magnetic recording. More recently, and independently, $(d, k, 2)$-RLL constraints were shown to play a natural role in magneto-optic recording systems. In a particular example of such a system [HRuHC], [RuS89], not only was it detrimental to record odd run-lengths of 0's between successive 1's, but it was actually impossible! We explain this briefly as follows.

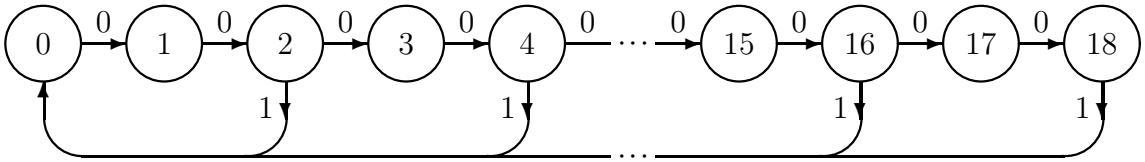


Figure 1.12: Graph presentation of the $(2, 18, 2)$-RLL constraint.

In this system, the recording medium can be magnetized only when it is heated above a certain threshold temperature. The heat is supplied by a laser, and the magnetization by a sinusoidally varying background magnetic field with constant frequency and amplitude.

The period of the field is twice the bit period T. The laser fires only at peaks (positive or negative) of the field. When it fires it marks a portion of the track with one magnetic polarity or the other (see Figure 1.13). In the figure, the cross-hatched regions indicate one

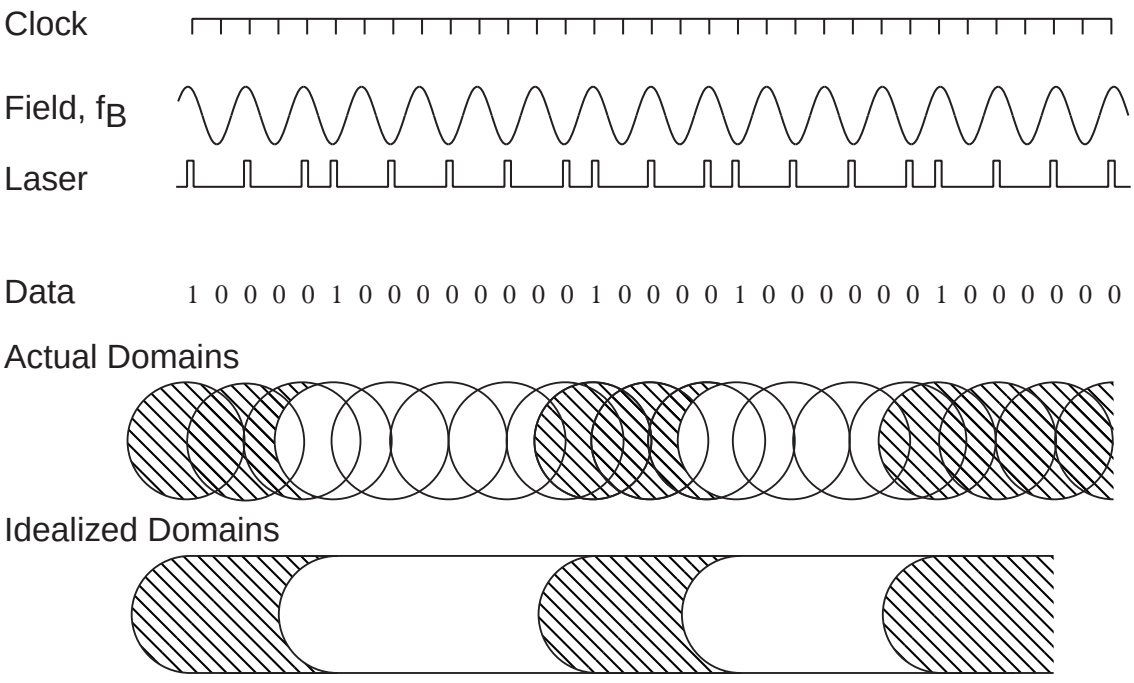


Figure 1.13: A magneto-optic recording system.

polarity and the blank regions indicate the opposite polarity. After the laser fires, it must next fire either T seconds later or $2T$ seconds later. In other words, the laser may skip one clock cycle but can never skip two consecutive clock cycles (the disk velocity is set so that there is enough overlap between marks that the track is continuously written). Thus,

a magnetic transition occurs when and only when the laser does *not* skip a clock cycle. So, if a magnetic transition is caused by firing the laser at bit positions $i-1$ and i, and the next magnetic transition is caused by firing the laser at bit positions $i+k$ and $i+k+1$, then k must be even. If we view the recorded sequence as $1000\ldots0001$, where the first 1 is at bit position i and the next 1 is at bit position $i+k+1$, then the number of intervening 0's is k, an even number.

Data is read by employing the laser in a lower power mode to reflect light off the medium, instead of magnetizing the medium. The magnetic polarities, and therefore the magnetic transitions, can be determined from the reflected light by exploiting a phenomenon known as Kerr rotation. A prototype for this type of magneto-optical recording system, that employs the $(2, 18, 2)$-RLL constraint, was built by IBM [HRuHC], [RuS89], [Weig88]. For more background on optical recording, the reader is referred to [Bouw85], [Heem82], [Imm91, Ch. 2], [Pohl92].

1.5.4 Spectral-null constraints

So far, we have focused on binary sequences, mainly because runlength constraints are naturally defined on binary transition sequences. However, some constraints are naturally defined directly on the sequences of magnetic polarities that are actually recorded. We use the bipolar alphabet $\{+1, -1\}$, often denoted simply $\{+, -\}$, for these sequences, i.e., one polarity is represented by $+1$ and the opposite polarity is represented by -1. When we want to emphasize that a bipolar sequence represents a sequence of polarities, we call it a *polarity sequence*.

We say that a constrained system of bipolar sequences has a *spectral null* at a (normalized) frequency $f = m/n$ if there exists a constant B such that for all sequences $\mathbf{w} = w_0 w_1 \ldots w_{\ell-1}$ that satisfy the constrained system and $0 \leq i \leq i' < \ell$ we have

$$\left| \sum_{s=i}^{i'} w_s e^{-\jmath 2\pi s m/n} \right| \leq B \ , \tag{1.1}$$

where $\jmath = \sqrt{-1}$ [MS87], [Pie84], [YY76] (the *actual* frequency is given by f/T, where T is the bit period). There are other equivalent ways to express this condition in terms of a power spectral density [Sklar88].

Sequences with a spectral null at $f = 0$, often called *dc-free* or *charge-constrained* sequences, have been used in many magnetic tape recording systems employing rotary-type recording heads, such as the R-DAT digital audio tape systems. Related constraints are imposed in optical recording to reduce interaction between the recorded data and the servo system, and also to allow filtering of low-frequency noise resulting from finger-prints [Imm91, Ch. 2]. The dc-free constraint is also used in communication systems, where low frequency

signals tend to be distorted; this includes cable communication lines [GHW92, sec. 4.8.1] as well as fiber optic links [WF83].

When $f = 0$, the maximum value of the left-hand side in (1.1) for a given sequence $\mathbf{w} = w_0 w_1 \ldots w_{\ell-1}$ is called the *digital sum variation (DSV)* of the sequence. The DSV of $\mathbf{w}$ can also be written as

$$\max_{0 \le i \le i' < \ell} \Big| \sum_{s=i}^{i'} w_s \Big| = \Big(\max_{-1 \le r < \ell} \sum_{s=0}^{r} w_s \Big) - \Big(\min_{-1 \le r < \ell} \sum_{s=0}^{r} w_s \Big) \, ,$$

which is the largest difference between any two sums that are computed over prefixes of the sequence (a sum over an empty set being regarded as zero). The larger the value of B, the less reduction there will be in the spectral content at frequencies approaching the spectral null frequency $f = 0$. The set of all sequences with DSV at most B is a constrained system called the *B-charge constraint*; it is presented by the labeled graph of Figure 1.14 (see Problem 1.8).

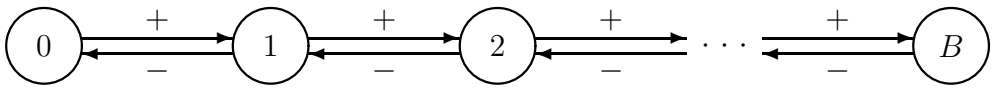


Figure 1.14: B-charge constraint.

Example 1.7 Consider the following sequence $\mathbf{w} = w_0 w_1 \ldots w_{20}$ over $\{+1, -1\}$, where we have also computed the sums $\sum_{s=0}^{i} w_s$ for $0 \le i < 21$:

w_i :	+	+	+	−	−	+	+	+	−	−	−	−	−	+	+	+	−	−	−	−	+
$\sum_{s=0}^{i} w_s$:	1	2	3	2	1	2	3	4	3	2	1	0	-1	0	1	2	1	0	-1	-2	-1

We have

$$\max_{0 \le i \le i' < 21} \Big| \sum_{s=i}^{i'} w_s \Big| = \Big(\max_{-1 \le r < 21} \sum_{s=0}^{r} w_s \Big) - \Big(\min_{-1 \le r < 21} \sum_{s=0}^{r} w_s \Big) = 4 - (-2) = 6 \, .$$

Therefore, the DSV of the sequence $\mathbf{w}$ is 6 (see also Figure 1.18 below). □

Graphs presenting constraints with spectral nulls at rational sub-multiples of the symbol frequency are described in [MS87]. Higher-order spectral-null constraints, which further restrict the spectral content in the vicinity of spectral-null frequencies, are discussed in [ImmB87], [KS91a], [EC91], [RSV94], [MPi89].

1.5.5 Combined charge–runlength constraints

In some applications it is desirable to impose both runlength constraints and charge-constraints. Recall that runlength constraints are expressed in the setting of binary transition sequences, and charge-constraints are expressed in the setting of bipolar polarity sequences. So, in order to describe a combined charge-runlength constraint, we must have a way of converting from one setting to the other. Formally, this is done as follows.

A bipolar polarity sequence $\mathbf{w} = w_0 w_1 w_2 \cdots$ is transformed into a binary transition sequence $\mathbf{z} = z_0 z_1 z_2 \cdots$ by the transformation

$$z_i = |w_i - w_{i-1}|/2. \tag{1.2}$$

Note that $z_i = 1$ if and only if $w_i \neq w_{i-1}$, equivalently if and only if there is a transition in magnetic polarity.

Conversely, a binary transition sequence $\mathbf{z}$ is transformed into a bipolar polarity sequence $\mathbf{w}$ through an intermediate binary sequence $\mathbf{x} = x_0 x_1 x_2 \cdots$ according to the rules

$$x_i = x_{i-1} \oplus z_i \quad \text{and} \quad w_i = (-1)^{x_i}\ , \tag{1.3}$$

where $\oplus$ denotes addition modulo 2. The value of x_{-1} is set arbitrarily to either 0 or 1, thereby giving rise to two sequences $\mathbf{w}$ which are the same up to overall polarity inversion. Observe that $w_i \neq w_{i-1}$ if and only if $z_i = 1$.

The transformation defined by equation (1.3) is called *precoding* because it describes how a binary sequence, such as a runlength limited sequence, must be converted into pulses of electrical current before being written on the medium. Naturally then, the transformation defined by (1.2) is called *inverse precoding*.

The *B–(d,k)-charge–RLL (CRLL) constraint* is the set of binary sequences $\mathbf{z}$ that satisfy the (d,k)-RLL constraint with the additional restriction that the corresponding precoded bipolar sequence $\mathbf{w}$ has DSV no larger than B.

Example 1.8 Consider the labeled graph in Figure 1.15. It can be verified (Problem 1.10) that any precoding of each sequence that can be generated by that graph satisfies the 6-charge constraint. Conversely, every sequence that satisfies the 6-charge constraint is a precoding of a sequence that can be generated by the graph in Figure 1.15. Therefore, the 6–$(1,3)$-CRLL constraint consists of all binary sequences that can be generated simultaneously by the labeled graphs in Figure 1.15 *and* Figure 1.16. In Section 2.4.3, we will show how, given any two labeled graphs, to construct a third labeled graph that presents the constraint defined by both graphs simultaneously. □

The 6–$(1,3)$-CRLL and 6–$(1,5)$-CRLL constraints have found application in commercial tape recording systems [Patel75], [MM77], [Mill77].

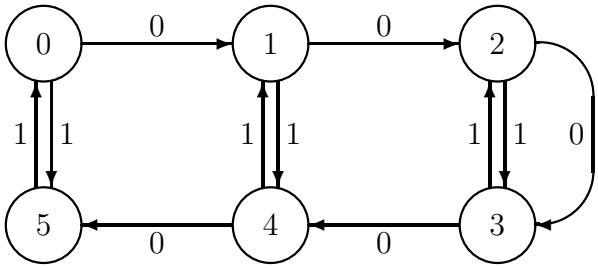


Figure 1.15: Graph presentation of precoding of the 6-charge constraint.

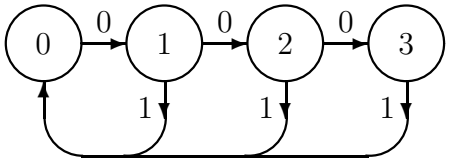


Figure 1.16: Graph presentation of the $(1, 3)$-RLL constraint.

1.5.6 Constraints for PRML

Within the last decade, magnetic recording systems using digital signal processing methods have appeared on the scene. These systems typically employ a scheme, denoted PRML, based on partial-response (PR) signaling, with maximum-likelihood (ML) sequence detection. See [Cid92], [Dol89], [DMU79], [KobT70], [WoodP86]. In many recording applications, PRML has replaced peak detection.

In PRML systems it proves to be desirable to use binary sequences which satisfy not only a "global" k constraint, denoted $\mathbf{G}$, but also a separate "interleaved" k constraint, denoted $\mathbf{I}$, on the even index and odd index subsequences. The $\mathbf{G}$ constraint plays exactly the same role as the k constraint, used for timing control, as in Section 1.2. The $\mathbf{I}$ constraint aids the method (called Viterbi detection) by which data is retrieved in a PRML system. The data stream is divided into its two sub-strings (the even index and odd index), and each is retrieved separately. It turns out that an $\mathbf{I}$ constraint reduces the probability of long delay in the output of the Viterbi detector. See section 1.6.2 for more detail on PRML and the reasons for the $\mathbf{I}$ constraint.

To help distinguish the PRML $(\mathbf{G}, \mathbf{I})$ constraints from the (d, k)-RLL constraints. we will use the notation $(0, \mathbf{G}/\mathbf{I})$; the 0 may be thought of as a $d = 0$ constraint, emphasizing that interference between adjacent transition responses is now acceptable. An example of a sequence satisfying the $(0, 4/4)$ constraint is

$$00100001001000100100 \, .$$

We can represent $(0, \mathbf{G}/\mathbf{I})$ constraints by labeled graphs based on states which reflect the three relevant quantities, the number g of 0's since the last occurrence of 1 in the sequence and the numbers a and b which denote the number of 0's since the last 1 in the two interleaved sub-strings. We name the states by pairs (a, b), where a is the number of 0's in the interleaved sub-string containing the next to last bit, and b is the number in the sub-string containing the last bit. Note that g is a function of a and b, denoted $g(a, b)$:

$$g(a,b) = \begin{cases} 2a+1 & \text{if } a < b \\ 2b & \text{if } a \geq b \end{cases} .$$

In the (a, b) notation, the set of states V for a $(0, \mathbf{G}/\mathbf{I})$ constraint is given by

$$V = \{(a,b) \ : \ 0 \leq a, b \leq \mathbf{I} \ \text{ and } \ g(a,b) \leq \mathbf{G}\}$$

and the labeled edges between states are given by

$$(a,b) \xrightarrow{0} (b, a+1) \ , \ \text{ provided } (b, a+1) \in V$$
$$(a,b) \xrightarrow{1} (b, 0) \ .$$

Example 1.9 A graph presentation for the $(0, \mathbf{G}/\mathbf{I}) = (0, 4/4)$ constraint is shown in Figure 1.17, where state labels (omitted) agree with integer grid coordinates, starting with $(0, 0)$ at the lower left. Only the 0-labeled edges are shown in the figure; the reader can fill in the 1-labeled edges. □

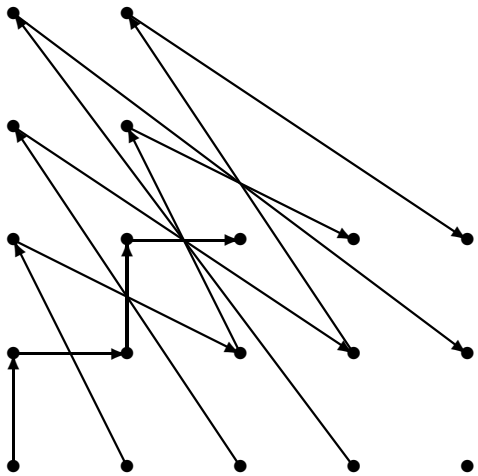


Figure 1.17: PRML $(0, \mathbf{G}/\mathbf{I}) = (0, 4/4)$ constraint: 0-labeled edges in graph presentation.

1.6 Background on magnetic recording

This section is not essential to the remainder of the text. It is intended for those readers who might be interested in understanding in a little more detail the signal processing methods used in digital magnetic recording and the motivation for introducing the constraints described in Sections 1.2 and 1.5.6.

1.6.1 Peak detection

In this section we elaborate on the description of the magnetic recording process given in Section 1.1.

Recall that in magnetic recording systems, the magnetic material at a given position along a track can be magnetized in one of two possible, opposing directions. The normalized input signal applied to the recording head in this process can be thought of as a two-level waveform $w(t)$ which assumes the values $+1$ and -1 over consecutive time intervals of bit period T. In the waveform, the transitions from one level to another, which effectively carry the digital information, are therefore constrained to occur at integer multiples of the bit period T, and we can describe the waveform digitally as a sequence $\mathbf{w} = w_0 w_1 w_2 \cdots$ over the bipolar alphabet $\{+1, -1\}$, where w_i is the signal amplitude in the time interval $(iT, (i+1)T]$.

Example 1.10 Figure 1.18 shows an input waveform that corresponds to the sequence $\mathbf{w}$ of Example 1.7. The figure also contains the integral of $w(t)$ over time. Since $\int_{u=0}^{(i+1)T} w(u)du = \sum_{s=0}^{i} w_s$, one sees from the figure that the DSV of $\mathbf{w}$ is indeed 6. □

Denote by $2h(t)$ the output signal (readback voltage), in the absence of noise, corresponding to a single transition from, say, -1 to $+1$ at time $t = 0$. If we assume that the input-output relationship of the digital magnetic recording channel is linear, then the output signal $y(t)$ generated by the waveform represented by the sequence $\mathbf{w}$ is given by:

$$y(t) = \sum_{i=0}^{\infty} (w_i - w_{i-1})\, h(t - iT) \ ,$$

with $w_{-1} = 1$. Note that the "derivative" sequence $\mathbf{w}'$ of coefficients $w'_i = w_i - w_{i-1}$ consists of elements taken from the ternary alphabet $\{0, \pm 2\}$, and the nonzero values, corresponding to the transitions in the input signal, alternate in sign.

A frequently used model for the transition response $h(t)$ is the function

$$h(t) = \frac{1}{1 + (2t/\tau)^2} \ ,$$

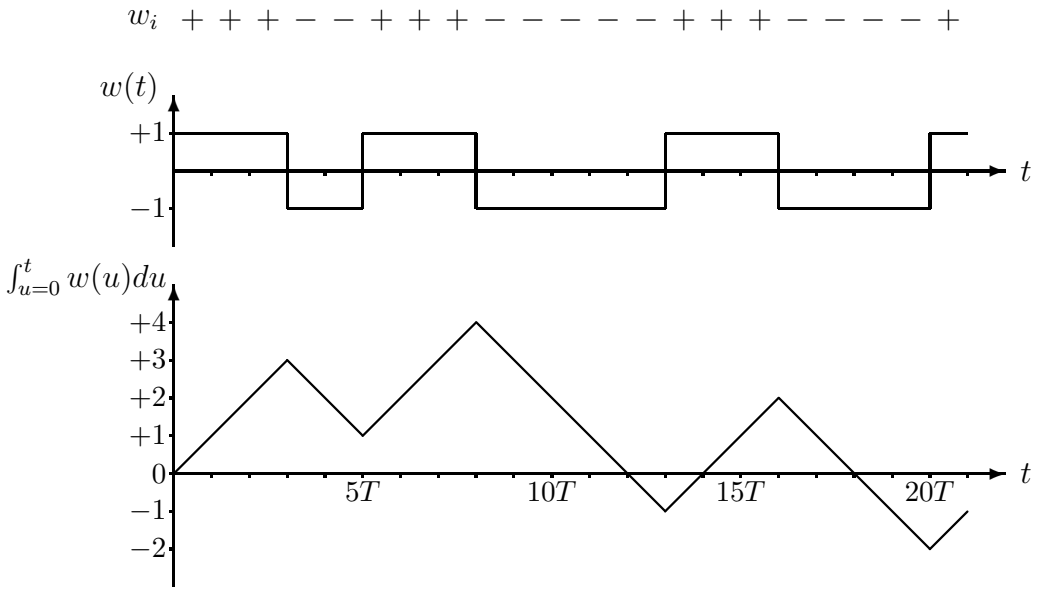


Figure 1.18: Input waveform $w(t)$ and integral $\int_{u=0}^{t} w(u)du$ that correspond to a sequence $\mathbf{w}$.

often referred to as the Lorentzian isolated-step response. The output signal $y(t)$ is therefore the linear superposition of time-shifted Lorentzian pulses with coefficients of magnitude 2 and alternating polarity. Provided that the density of transitions—reflected in the so-called density ratio τ/T—is small enough, the locations of peaks in the output signal will closely correspond to the locations of the transitions in the recorded input signal. With a synchronous clock of period T, one could then, in principle, reconstruct the ternary sequence $\mathbf{w}'$ and the recorded bipolar sequence $\mathbf{w}$.

Recall that the peak detector determines the location of peaks in the (possibly noisy) output signal whose amplitude exceeds a pre-specified level. As described earlier, runlength constraints are desirable for mitigating the effects of inter-symbol interference and improving the performance of timing recovery schemes. Specifically, the runlength constraint on binary transition sequences $\mathbf{z} = z_0 z_1 z_2 \cdots$ can be translated directly into the constraint on $\mathbf{w}'$ by means of precoding (1.3) above:

$$w'_i = w_i - w_{i-1} = (-1)^{x_{i-1}}((-1)^{z_i} - 1) = -(-1)^{x_{i-1}} \cdot 2\, z_i \; .$$

So,

$$|w'_i| = 2\, z_i,$$

and so the runlength constraints on $\mathbf{w}'$ become: the sequence $\mathbf{w}'$ contains at least d symbols and at most k symbols of value zero between successive nonzero values.

1.6.2 PRML detection

At high recording densities, the PRML (partial response maximum likelihood) approach to be discussed here has been shown to provide increased reliability relative to peak detection. The motivation for using $(0, \mathbf{G}/\mathbf{I})$ constraints can be found in the operation of the PRML system, which we now describe in simplified terms. The key difference between PRML and peak detection systems is that PRML reconstructs the recorded information from the *sequence of sample values* of the output signal at times $t = 0, T, 2T, 3T, \ldots$, rather than from individual peak locations. Denote by $\mathrm{sinc}(x)$ the real function $(\sin(\pi x))/(\pi x)$. The PRML system uses an electronic filter to transform the output pulse $2h(t)$ resulting from an isolated transition at time $t = 0$ into a modified pulse $2f(t)$ where

$$f(t) = \mathrm{sinc}\left(\frac{t}{T}\right) + \mathrm{sinc}\left(\frac{t-T}{T}\right) \ . \tag{1.4}$$

Note that at the consecutive sample times $t = 0$ and $t = T$, the function $f(t)$ has the value 1, while at all other times which are multiples of T, the value is 0. This particular partial-response filtering is referred to as "Class-4" [Kretz67]. Through linear superposition, the output signal $y(t)$ generated by the waveform represented by the bipolar sequence $\mathbf{w}$ is given by:

$$y(t) = \sum_{i=-1}^{\infty} (w_i - w_{i-1})\, f(t - iT) \ ,$$

where we set $w_{-2} = w_{-1} = w_0$. Therefore, *at sample times*, the Class-4 transition response results in *controlled interference*, leading to output signal samples $y_i = y(iT)$ that, in the absence of noise, assume values in $\{0, \pm 2\}$. Hence, in the noiseless case, the recorded bipolar sequence $\mathbf{w}$ can be recovered from the output sample values $y_i = y(iT)$, because the interference between adjacent transitions is prescribed. Therefore, unlike the peak detection system, PRML does not require the separation of transitions.

The $(0, \mathbf{G}/\mathbf{I})$ constraints arise from the following considerations. Recall that the parameter $\mathbf{G}$ is comparable to the k constraint in peak detection constraints, ensuring effective operation of the PRML timing recovery circuits, which typically rely upon frequent occurrence of nonzero output samples. Specifically, the $\mathbf{G}$ constraint represents the maximum number of consecutive zero samples allowed in the sample sequence $y_0 y_1 y_2 \cdots$.

The parameter $\mathbf{I}$ is intimately related to the maximum-likelihood detection method used in PRML. Before discussing the ML detection algorithm, it is useful to rewrite the output signal as

$$y(t) = \sum_{i=0}^{\infty} (w_i - w_{i-2})\, \mathrm{sinc}\left(\frac{t - iT}{T}\right) \ ,$$

where, we recall, $w_{-2} = w_{-1} = w_0$. This form can be obtained by simple arithmetic from the original expression for the Class-4 transition response (1.4). This implies the following relation between the noiseless output samples $y_0 y_1 y_2 \cdots$ and the input bipolar sequence $\mathbf{w}$:

$$y_i = w_i - w_{i-2} \ , \quad i \geq 0 \ .$$

A trellis diagram presenting the possible output sample sequences is shown in Figure 1.19. Each state is denoted by a pair of symbols that can be interpreted as the last pair of inputs, $w_{i-2}w_{i-1}$. There is an edge connecting each pair of states $w_{i-2}w_{i-1}$ and $w_{i-1}w_i$, and the label of this edge is $y_i = w_i - w_{i-2}$.

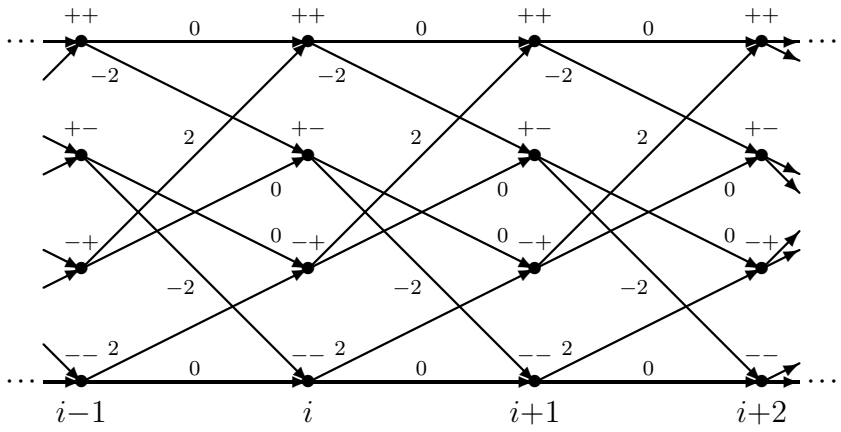


Figure 1.19: Trellis diagram for Class-4 output sequences.

The iterative ML detection algorithm is based upon the technique of dynamic programming in an embodiment of the *Viterbi algorithm*, familiar from decoding of convolutional codes. As shown by Forney [For72] and Kobayashi [Koba71], the Viterbi algorithm is an optimal detector for partial-response (in particular, Class-4) output-signal sample sequences in the presence of additive white Gaussian noise.

The behavior of the ML detector can be described in terms of the trellis diagram in Figure 1.19. Denote by $r_0 r_1 r_2 \cdots$ the sequence of (possibly noisy) received samples. Assume an initial state $u = w_{-2}w_{-1}$ is specified. For each state $v = w_{\ell-2}w_{\ell-1}$ in the diagram that can be reached from u by a path of length ℓ, the ML detector determines the allowable noiseless output sample word $\hat{y}_0\hat{y}_1 \ldots \hat{y}_{\ell-1}$, generated by a path of length ℓ from u to v, that minimizes the squared Euclidean distance

$$\sum_{i=0}^{\ell} (r_i - \hat{y}_i)^2 \ .$$

The words so determined are referred to as *survivor words*.

The representation of the output samples as $y_i = w_i - w_{i-2}$ permits the detector to operate independently on the output subsequences at even and odd time indices. Note that, within each interleave, the nonzero sample values y_i must alternate in sign. Figure 1.20 shows a trellis diagram presenting the possible sequences in each of the interleaves.

There are several formulations of the Viterbi algorithm applied to this system. See [FC89], [Koba72], [SW91], [WoodP86], [Ze87]. The motivation for the **I** constraint is clear from the following description of the ML detection algorithm, essentially due to Ferguson [Ferg72].

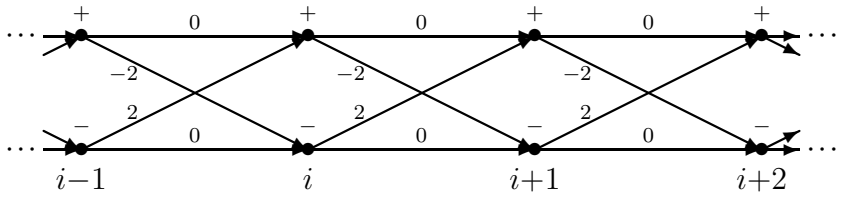


Figure 1.20: Trellis diagram for each of the interleaves.

It can be interpreted in terms of a "dynamic threshold detection scheme," as described by Wood [Wood90]. We will outline the detector operation on the subsequence of received samples at even time indices. The procedure for the odd time indices is entirely analogous.

The detector may operate in one of two modes, denoted by the variable m which takes values in $\{+1, -1\}$, according to whether the detector expects the next non-zero sample value to be positive or negative. In mode $m = +1$ (respectively, $m = -1$), the detector uses a variable R to store the *value* of the largest (respectively, smallest) sample since the last change of the mode variable m. It also maintains a variable J to store the *time index* i of the largest (respectively, smallest) sample value since the last change of the mode m.

The detector mode and variables are initialized by setting $m \leftarrow +1$, $R \leftarrow -\infty$, and $J \leftarrow -1$.

At each successive time instant $i = 2j$, $j \geq 0$, the detector takes one of three possible actions, as determined by m, R, J, and the new noisy sample r_{2j}:

1. If $m \cdot r_{2j} \geq m \cdot R$, then do $\hat{y}_J \leftarrow 0$, $R \leftarrow r_{2j}$, and $J \leftarrow 2j$;
2. else if $m \cdot R - 2 < m \cdot r_{2j} < m \cdot R$, then do $\hat{y}_{2j} \leftarrow 0$;
3. else if $m \cdot r_{2j} \leq m \cdot R - 2$, then do $\hat{y}_J \leftarrow 2m$, $R \leftarrow r_{2j}$, $J \leftarrow 2j$, and $m \leftarrow -m$.

Case 1 corresponds to the situation in which the survivor words at time $2j$ for both states in Figure 1.20 are obtained by extending the survivor word at time $2(j-1)$ for state $u = -m$. Case 2 corresponds to the situation in which the survivor word at time $2j$ for each state u is the extension of the survivor word at time $2(j-1)$ for state u. Finally, case 3 corresponds to the situation in which the survivor words at time $2j$ for both states are obtained by extending the survivor word at time $2(j-1)$ for state $u = m$.

Cases 1 and 3 correspond to "merging" of survivor paths, thereby determining the estimated value $\hat{y}_J$ for the channel output at the index of the previous merge. In the noiseless case, the merges occur when the output sample value is either $+2$ or -2. Case 2, on the other hand, defers the decision about this estimated value. In the noiseless case, this arises when the output sample value is 0. Since the latter case could arise for an arbitrary number of successive time indices, one could encounter a potentially unbounded time span between

time J and the generation of the estimated channel output $\hat{y}_J$—even in the noiseless case. The **I** constraint on the maximum runlength of consecutive zero samples in each interleave of the output sequence is introduced to reduce the probability of such a long delay (or eliminate the possibility in the noiseless case).

In analogy to the RLL constraints, the **G** and **I** constraints on a binary sequence $\mathbf{z}$ translate directly to the corresponding constraints on the ideal output sample sequences $y_0 y_1 y_2 \cdots$. This is accomplished by applying precoding to each of the interleaves of $\mathbf{z}$. This *interleaved precoding* transforms $\mathbf{z}$ into a bipolar polarity sequence $\mathbf{w}$ via an intermediate binary sequence $\mathbf{x}$ according to the rules

$$x_i = x_{i-2} \oplus z_i \quad \text{and} \quad w_i = (-1)^{x_i} \ ,$$

where $x_{-2} = x_{-1} = 0$ and, as before, $\oplus$ denotes addition modulo 2. The constraint on the runlengths of consecutive 0's in the output sample sequence and in the even/odd subsequences are then reflected in corresponding $(0, \mathbf{G}/\mathbf{I})$ constraints on the binary sequences $\mathbf{z}$.

1.7 Coding in optical recording

We describe here the coding methods used in two optical-recording applications: the compact disk (CD) [Imm91, Ch. 2] and digital versatile disk (DVD) [Imm95b]. We start by a very brief and superficial description of the physical characterization of the stamped compact disk. For a much more detailed information, see [Bouw85, Ch. 7], [Heem82], [Imm91, Ch. 2], [IO85], and [Pohl92, Chapter 3].

1.7.1 The compact disk

The stamped disk consists of a round metal film that is coated by a transparent plastic material. The latter serves as a magnifying glass for the laser light that is beamed at the bottom side of the disk (i.e., the side without the label), and the reflection from the metal surface is received during readback by a light detector. Data is recorded by imprinting on the top side of the metal a sequence of pits along tracks, starting from the inner circumference to the outer (so, from the bottom side of the film, the pits seem as bumps). The pits are of varying lengths, and so are the gaps in between them. The lengths of the pits and the gaps range between $3T$ and $11T$, where $T \approx .2777\mu$m. Figure 1.21 shows portions of three adjacent tracks on the metal surface, with the lengths of the shortest and longest pits, the width of the pits, the distance between tracks (the track pitch), and the diameter of the laser spot (seen in the middle track). The lengths of the pits and gaps are determined by a $(2, 10)$-RLL constrained sequence, which is shown for the upper track ("Track i") at the

top of the figure, along with a profile of the upper track as it would have appeared from the thin side of the disk, had the disk been cut along that track.

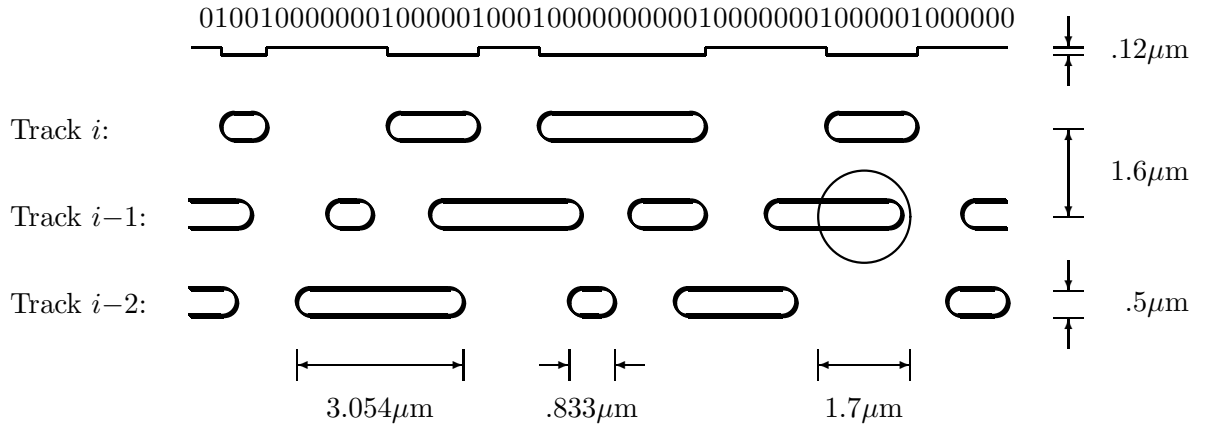


Figure 1.21: Tracks in the compact disk.

The height of the bumps (i.e, the depth of the pits) is approximately one-quarter of the wavelength of the laser beam (as the beam passes through the transparent plastic material). Therefore, a beam reflected from the bumps will destructively interfere with a reflection from the surrounding 'land.' During readback, a laser beam is aimed at a particular track and the light detector receives the reflection as the disk rotates. In this process, the bumps will appear darker to the light detector, compared to the gaps.

The parameter $T \approx .2777\mu$m has been selected so that different pit (or gap) lengths can be distinguished in the received signal. The parameters $d = 2$ and $k = 10$ have been set due to reasons akin to those that exist in magnetic recording. Specifically, pits or gaps are constrained not to be shorter than $(d+1)T = 3T$, or else the detector might have missed them (note that the diameter of the laser spot is still more than twice this shortest length). The upper bound of $(k+1)T = 11T$ is imposed to prevent any clock drift and allow clock synchronization. This requirement is particularly important in optical recording: since the linear density along a track (i.e., the parameter T) is fixed regardless of the location of the track, the angular velocity of the disk needs to be adjusted so that the *linear* velocity remains the same for all tracks. Such an adjustment is possible only if the clock can be extracted from the readback signal.

1.7.2 EFM code at rate $8 : 16$

We next describe a rate $8 : 16$ encoder for the $(2, 10)$-RLL constraint. This rate has been selected so that it matches the sub-division of the encoded bit stream into bytes.

Consider words of length 14 that satisfy the $(2, 10)$-RLL constraint, with the additional property that the first and last runlengths in each word are at most 8. It turns out that there are exactly 257 words of length 14 in the $(2, 10)$-RLL constraint that satisfy this property. We construct a table T that consists of 256 of those words.

A possible encoding strategy would be to encode each input byte $\mathbf{s} \in \{0, 1\}^8$ into the respective entry, $\mathsf{T}[\mathbf{s}]$, in the table T. Yet, the sequence obtained by the concatenation of such words may violate the constraint. One possible solution is adding a fixed number of *merging bits* in between the generated words. Those merging bits will be used to 'glue' the words correctly.

A simple check shows that two merging bits are sufficient to guarantee that the sequence of words satisfies the $(2, 10)$-RLL constraint. Table 1.2 shows an assignment of merging bits, depending on (the range of values of) the last runlength of the previously-encoded 14-bit word and (the range of values of) the first runlength of the 14-bit word that is currently encoded.

Last Encoded Runlength	First Runlength in $\mathsf{T}[\mathbf{s}]$		
	0	1–7	8
0	00		
1	00		01
2–8	00	10	

Table 1.2: Merging bits for a $(2, 10)$-RLL encoder.

Table 1.2 thus suggests an encoding process which can be represented as a rate $8 : 16$ encoder with three states. Those states are named 0, 1, and 2–8, with each name standing for (the range of values that contains) the last runlength of the previously-encoded 16-bit codeword. Given an input byte $\mathbf{s} \in \{0, 1\}^8$, the current codeword is obtained by preceding the entry $\mathsf{T}[\mathbf{s}]$ in T by two merging bits; those bits depend on the current encoder state and the first runlength of $\mathsf{T}[\mathbf{s}]$ according to Table 1.2. The next encoder state in then determined by the last runlength in $\mathsf{T}[\mathbf{s}]$.

We refer to the resulting encoder as a 3-*state eight-to-fourteen modulation code at rate* $8 : 16$, where the numbers 8 and 14 refer to the lengths of the addresses and entries, respectively, of T. This code will be denoted hereafter by 3-EFM(16), where 3 stands for the number of states and 16 is the codeword length.

The 3-EFM(16) code has a very simple sliding-block decoder that reconstructs each input byte from the corresponding 16-bit codeword, without requiring the knowledge of any previous or future codewords. Specifically, the decoder deletes the first two bits of the codeword, and the address in T of the resulting 14-bit word is the decoded input byte.

1.7.3 Dc control

While Table 1.2 lists one possible pair of merging bits for each encoder state and word in T, in certain cases there is a freedom of selection of those bits. Specifically, it can be verified that among the 257 words that can be taken as entries in T, there are 113 words in which the first runlength is between 2 and 7. When any of these words is generated from state 1 in the 3-EFM(16) code, then we could select 01 as merging bits instead of 00. In other words, we can invert the second bit of the generated codeword in this case without violating the constraint.

This freedom of inserting a bit inversion allows to reduce the digital sum variation (DSV) of the recorded sequence (see Section 1.5.4). We demonstrate this in the next example.

Example 1.11 Consider the $(2, 10)$-RLL constrained sequence

$$\mathbf{z}_1 = 010010000000100000100 \, ,$$

which is precoded into the signal $w_1(t)$, as shown in Figure 1.22 (refer to Section 1.5.5 to the precoding rule (1.3), and also to Example 1.7 and Figure 1.18). The value of the integral $\int_{u=0}^{t} w_1(u)du$ ranges between -1 and $+7$ and, therefore, the DSV of the recorded signal equals 8.

Suppose now that the eighth bit in $\mathbf{z}_1$ is inverted to form the sequence

$$\mathbf{z}_2 = 010010010000100000100 \, .$$

Denoting by $w_2(t)$ the respective recorded signal, the integral $\int_{u=0}^{t} w_2(u)du$ now ranges between -3 and $+3$, thereby reducing to DSV to 6. □

As explained earlier, the recording in optical media is based on changing the reflectivity of the recording surface. During readback, the surface is illuminated by a laser beam and the readback signal is obtained by measuring the amplitude of the reflection beam: high reflection ('bright surface') means a value $+1$, while low reflection ('dark surface') stands for -1. To determine whether the reflection is high or low, the detector needs to be calibrated to the zero level (referred to as the 'dc,' or 'direct current' level) of the signal. To this end, the recorded signal is constrained to have an average value close to zero over a given window length. This, in turn, allows the required calibration simply by computing the average of the amplitude of the reflected beam over a given period of time.

In Section 1.5.5, we introduced the combined charge-runlength constraints. If a B–$(2, 10)$-CRLL encoder were used for some value B, then, certainly, we would have an average value close to zero for every *individual* constrained sequence generated by the encoder. However, a small value for B would force the encoding ratio to drop below $8/16 = .5$, while a large value of B would result in a too complex encoder.

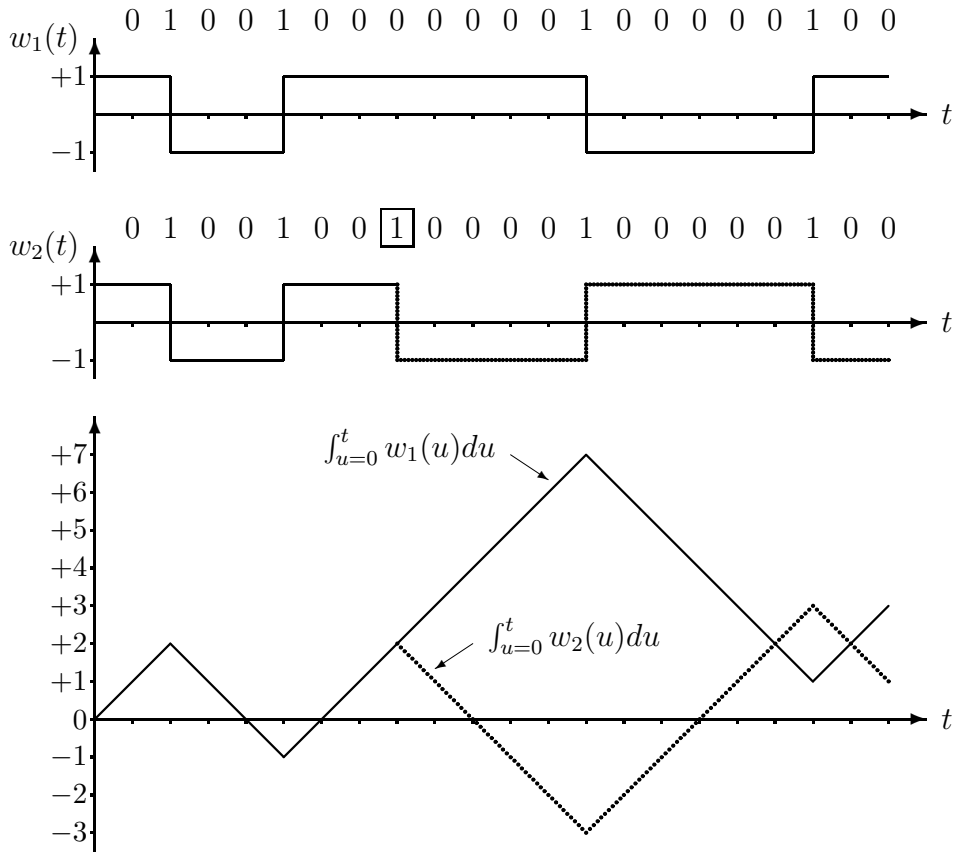


Figure 1.22: Effect of inverting a bit on DSV.

On the other hand, for the purpose of training on the zero level of the signal, it would be possible to weaken the charge constraint so that the DSV would be bounded over a sufficiently long sequence with very high probability (yet not necessarily for every individual sequence), where the probability is computed assuming (say) a uniform distribution over the sequences of input bytes. Indeed, the freedom of selecting the merging bits provides such a *dc (or DSV) control*: we will show in Section 4.7 (Example 4.12) that under a uniform distribution on the input bytes, once in approximately every ten input bytes (on average), we will be at state 1 with the freedom of selecting either 00 and 01 as merging bits

The freedom of selecting the merging bits can be increased by refining the definition of the states in the 3-EFM(16) code and designating a separate state for each possible value of the last runlength in a codeword. That is, instead of having only three states, 0, 1, and 2–8, we will have nine states that will be denoted 0 through 8; so, an input byte $\mathbf{s}$ will lead the encoder to state i, where i is the last runlength of $\mathsf{T}[\mathbf{s}]$. The resulting encoder will be called

a 9-EFM(16) code.

To see the advantage of the 9-EFM(16) code, consider the following example. Suppose that while at state 2–8 in the 3-EFM(16) code, the input byte $\mathbf{s}$ is such that $\mathsf{T}[\mathbf{s}]$ starts with a runlength 1. By Table 1.2, the merging bits in this case are 10, and, since it is possible to enter state 2–8 after generating a codeword that ends with a runlength 8, we cannot allow any other values for the merging bits in this case. On the other hand, if each of the runlength values 2 through 8 leads to a separate state, then, while at any of the states 2 through 7, the merging bits 00 are also admissible for the mentioned byte $\mathbf{s}$.

Clearly, the advantage of the 9-EFM(16) code comes with a price: this code is somewhat more complex than the 3-EFM(16) code. Table 1.3 presents the possible selection of merging bits for each state in the 9-EFM(16) code and each first runlength in $\mathsf{T}[\mathbf{s}]$.

	First Runlength in $\mathsf{T}[\mathbf{s}]$								
State	0	1	2	3	4	5	6	7	8
0	00								
1	00		00, 01						01
2	00	00, 10	00, 01, 10					01, 10	
3	00	00, 10	00, 01, 10				01, 10		
4	00	00, 10	00, 01, 10			01, 10			
5	00	00, 10	00, 01, 10		01, 10				
6	00	00, 10	00,01,10	01, 10					
7	00	00, 10	01, 10						
8	00	10	01, 10						

Table 1.3: Admissible merging bits in the 9-EFM(16) code.

While the freedom of selecting the merging bits in the 9-EFM(16) code could be sufficient for calibrating the zero level of the readback signal, charge constraints are required in optical recording also for the suppression of the power spectral density of the recorded signal at the low-frequency range. Such a suppression is necessary so that low-frequency control signals and noise can be filtered out without distorting the recorded signal (see [Imm91, p. 27]).

It turns out that the dc control that is attainable by the 9-EFM(16) code does not provide sufficient reduction of the power spectral density at the low-frequency range. This was resolved in the compact disk by inserting three merging bits instead of two, resulting in the (proper) EFM code , at rate $8 : 17$, which we denote by EFM(17) [Imm99, Section 14.4.1].

The solution in the DVD was different. Here, the encoding is not based on merging bits; instead, the encoder has four states, and each state has its own encoding table that maps input bytes into 16-bits codewords. In addition, 88 out of the 256 input bytes can be

encoded at each state into two possible codewords, thereby introducing freedom that allows sufficient suppression of the low-frequency range of the power spectral density (see [Imm95b] and [Imm99, Section 14.4.2]). The resulting encoder, called the EFMPlus code, has rate $8 : 16$. Yet, the decoding of an input byte requires the knowledge of *two* consecutive 16-bit codewords, compared to the EFM(16) and the EFM(17) codes where one codeword is sufficient. Alternative coding schemes at rate $8 : 16$ have been suggested where the desired dc control can be achieved with a decoding window length of one codeword only; see [Roth00] and [Imm99, Section 14.4.4].

1.8 Two-dimensional constraints

Constrained systems, as we have defined them so far, are sets of sequences; each sequence can be viewed as a one-dimensional stream of symbols in a one-dimensional space. In an analogous way, one can define constraints on two-dimensional (or even higher-dimensional) arrays.

Example 1.12 Consider the set of all two-dimensional arrays that satisfy the following condition: whenever one sees a 0 in the array, at least one of its four neighbors (up, down, left or right) is also a 0; in other words, the pattern:

$$\begin{array}{ccc} & 1 & \\ 1 & 0 & 1 \\ & 1 & \end{array} \tag{1.5}$$

is forbidden. □

One application of this kind of constraint occurs in holographic recoding systems, as shown in Figure 1.23 [HBH94]. In such a system, a laser illuminates a programmable array called a spatial light modulator (SLM). The SLM is an array of shutters, each of which can be open or closed; the open shutters allow light to pass through, while the closed shutters prevent light from passing through. So, if a two-dimensional array of 0's and 1's is programmed onto the SLM, illumination of the SLM by the laser will create an optical representation, called the *object beam*, of the array. The object beam is then focused through a lens. A *reference beam*, which is a simple plane wave propagating at some angle with respect to the medium, interferes with the focused object beam at a spot on the recording medium and writes a pattern (the hologram).

The object beam can be reproduced at a later time by illuminating the medium with the same reference beam that was used to record it; light from the reproduced object beam is passed through a lens and then collected on a photosensitive array of detectors, known as a charge coupled device (CCD). In this way, the original binary array of data can be recovered.

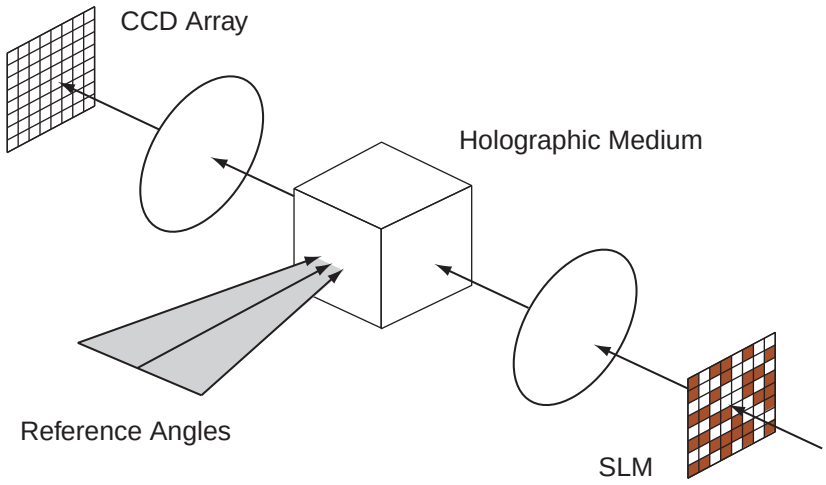


Figure 1.23: A holographic data storage system (figure taken from [HBH94]).

By varying the angle of propagation of the reference beam, several holograms can be recorded in the same physical spot, in a scheme known as angular multiplexing. Since many holograms can be recorded in one spot, holographic data storage holds the promise of very high data density. Since an entire array of data can be retrieved all at once, it also holds the promise of very high data rate.

Now, inter-pixel interference (IPI), the two-dimensional analog of inter-symbol interference, tends to occur when a 0 is surrounded by 1's; specifically, IPI may be provoked by the pattern (1.5) above. Since IPI degrades the performance of a holographic recording system, it may be desirable to forbid such a pattern. The collection of $N \times N$ arrays which do not contain the pattern (1.5) is a two-dimensional constrained system of arrays. It then becomes important to find methods of encoding arbitrary binary sequences into such a constrained system.

Recently, there has been a great deal of interest in coding for two-dimensional constrained systems. While some of this has been specifically geared towards applications such as holographic recording (e.g., [AM98], [BM99], [KN99]) a great deal of it has been focused on efforts to extend the general one-dimensional theory of constrained systems, as presented in this text, to two dimensions (e.g., [KZ98], [RSW00]). At this point, the two-dimensional theory is still sketchy. For instance, while we will see in Chapter 3 that there is an explicit formula for the capacity of any one-dimensional constrained system, there is nothing like this known in two dimensions.

Problems

Problem 1.1 Any sequence $\mathbf{z}$ that satisfies the (d,k)-RLL constraint begins with a certain number, j, of 0's, where $0 \le j \le k$. As a function of j, identify the states in Figure 1.3 from which $\mathbf{z}$ can be generated.

Problem 1.2 Show that a sequence satisfies the $(0,1)$-RLL constraint if and only if the inverted sequence (obtained by changing each 0 to 1 and each 1 to 0) satisfies the $(1,\infty)$-RLL constraint.

Problem 1.3 For given d and k, exhibit a list of exactly $d+1$ words that constitute a forbidden list for the (d,k)-RLL constraint. Is $d+1$ the minimal size of such a list?

Problem 1.4 A binary sequence satisfies the asymmetric (d_0,k_0,d_1,k_1)-RLL constraint if it consists of alternating runs of 0's and 1's such that the runlengths of 0's lie in between d_0 and k_0 and the runlengths of 1's lie in between d_1 and k_1. Draw a graph presentation of the $(1,3,2,5)$-RLL constraint.

Problem 1.5 Change the assignment of input tags on the encoder in Figure 1.8 so that it has a corresponding sliding-block decoder.

Problem 1.6 For a given (d,k)-RLL constraint and a given integer N, identify the sequences of length N that satisfy the constraint and have maximal duty cycle (i.e., maximal percentage of 1's).

Problem 1.7 Verify that the transformations, precoding (1.3) and inverse precoding (1.2), are inverses of one another in the following sense:

1. the inverse precoding of the precoding of $\mathbf{z}$ is $\mathbf{z}$;
2. the precoding of the inverse precoding of $\mathbf{w}$ is either $\mathbf{w}$ or the sequence obtained from $\mathbf{w}$ by interchanging 1's and -1's.

Problem 1.8 Let $\mathbf{w} = w_0 w_1 w_2 \cdots$ be a sequence over the alphabet $\{+1,-1\}$. Show that $\mathbf{w}$ can be generated by the labeled graph in Figure 1.14 starting at state j if and only if

$$-\min_{-1\le r<\ell} \sum_{s=0}^{r} w_s \le j \le B - \max_{-1\le r<\ell} \sum_{s=0}^{r} w_s \ .$$

Deduce that $\mathbf{w}$ satisfies the B-charge constraint if and only if it can be generated by the labeled graph in Figure 1.14.

Problem 1.9 Show that Figure 1.24 generates all the sequences obtained by precoding, given by formula (1.3), of sequences that satisfy the $(1,3)$-RLL constraint. Draw the corresponding picture for arbitrary d and k.

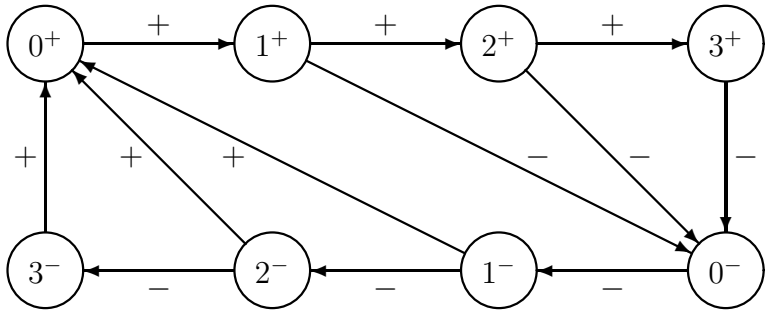


Figure 1.24: Graph presentation of precoding of the $(1, 3)$-RLL constraint.

Problem 1.10 Show that a sequence satisfies the 6-charge constraint if and only if it can be obtained by precoding of a sequence generated by the labeled graph in Figure 1.15.

Problem 1.11 Fill in the edges labeled by 1 in the graph presentation Figure 1.17.

Problem 1.12 Construct a (finite) graph presentation of the constraint described in Example 1.5.

Problem 1.13 Let ℓ be a positive integer and let $Z_{2\ell}$ denote the set of all sequences of length 2ℓ that satisfy the $(1, \infty)$-RLL constraint. Define the mapping $f : Z_{2\ell} \to \{+, -\}^{2\ell}$ as follows: $f(z_0, z_1, \ldots, z_{2\ell-1}) = w_0 w_1 \ldots w_{2\ell-1}$, where for each $i = 0, 1, \ldots, \ell-1$, the values of w_{2i} and w_{2i+1} depend on the values of z_{2i-1}, z_{2i}, and z_{2i+1} according to Table 1.4 (we assume that $z_{-1} = 0$).

z_{2i-1}	z_{2i}	z_{2i+1}	w_{2i}	w_{2i+1}
0	0	0	−	+
0	0	1	+	+
0	1	0	+	−
1	0	0	−	−
1	0	1	+	−

Table 1.4: Mapping for Problem 1.13.

1. Show that the mapping f is one-to-one.
2. Let $w_0 w_1 \ldots w_{2\ell-1}$ be an image of $z_0 z_1 \ldots z_{2\ell-1}$ under f. Show that for each $i = 0, 1, \ldots, \ell-1$,
$$\sum_{s=0}^{2i+1} w_s = 2z_{2i+1} \ .$$
3. Show that the images of f satisfy the 2-charge constraint.

4. Let $\mathbf{w} = w_0 w_1 \ldots w_{2\ell-1}$ be a sequence that satisfies the 2-charge constraint. Show that $\mathbf{w}$ is an image under f if one of the following conditions holds:

   (a) $w_{2i} \neq w_{2i+1}$ for every $0 \leq i < \ell$, or else —

   (b) if i is the first index such that $w_{2i} = w_{2i+1}$, then $w_{2i} = w_{2i+1} = +$.

5. Deduce that the number of sequences of length 2ℓ that satisfy the 2-charge constraint is at least—but no more than twice—the size of $Z_{2\ell}$.

Problem 1.14 (Enumeration of (d,k)-RLL sequences) Fix d and k to be integers $0 \leq d \leq k < \infty$, and denote by $x(\ell)$ the number of words of length ℓ that satisfy the (d,k)-RLL constraint. Also, denote by $x_0(\ell)$ the number of such words that do not contain any 1's among their first d bits. Define $x(\ell) = x_0(\ell) = 0$ for $\ell < 0$ and $x(0) = x_0(0) = 1$.

1. Show that $x_0(\ell)$ satisfies for $\ell \geq 0$ the recurrence

$$x_0(\ell) = u(k-\ell) + \sum_{i=d+1}^{k+1} x_0(\ell-i) \ ,$$

   where

$$u(t) = \begin{cases} 0 & \text{if } t < 0 \\ 1 & \text{if } t \geq 0 \end{cases} \ .$$

2. Show that for every ℓ,

$$x(\ell) = \sum_{i=0}^{d} x_0(\ell-i) \ .$$

3. Conclude from parts 1 and 2 that $x(\ell)$ satisfies for $\ell > k$ the recurrence

$$x(\ell) = r(k+d+1-\ell) + \sum_{i=d+1}^{k+1} x(\ell-i) \ ,$$

   where

$$r(t) = \begin{cases} 0 & \text{if } t < 0 \\ t & \text{if } t \geq 0 \end{cases} \ .$$

   (Note that this recurrence becomes linear when $\ell > k+d$.)

4. Show that the values of $x(\ell)$ for $0 \leq \ell \leq k$ satisfy

$$x(\ell) = \begin{cases} \ell+1 & \text{for } 0 \leq \ell \leq d \\ x(\ell-1) + x(\ell-d-1) & \text{for } d < \ell \leq k \end{cases} \ .$$

5. Assume now that $k = \infty$. Show that for $\ell > d$,

$$x_0(\ell) = x_0(\ell-1) + x_0(\ell-d-1) \ ,$$

   with the initial conditions

$$x_0(\ell) = 1 \ , \quad 0 \leq \ell \leq d \ .$$

6. Show that when $k = \infty$, the values $x(\ell)$ are related to $x_0(\ell)$ by

$$x(\ell) = x_0(\ell+d) \; ;$$

so, $x(\ell)$ satisfies for $\ell > d$,

$$x(\ell) = x(\ell-1) + x(\ell-d-1) \; ,$$

with the initial conditions

$$x(\ell) = \ell+1 \; , \quad 0 \leq \ell \leq d \; .$$

Chapter 2

Constrained Systems

2.1 Labeled graphs and constraints

First, we recall a convenient diagrammatic method used to present a constrained system of sequences. An encoder, in turn, may generate sequences only from this set.

A *labeled graph* (or a *finite labeled directed graph*) $G = (V, E, L)$ consists of —

- a finite set of states $V = V_G$;
- a finite set of edges $E = E_G$ where each edge e has an *initial state* $\sigma_G(e)$ and a *terminal state* $\tau_G(e)$, both in V;
- an edge labeling $L = L_G : E \to \Sigma$ where Σ is a finite alphabet.

We will also use the notation $u \xrightarrow{a} v$ to denote an edge labeled a from state u to state v in G.

Figure 2.1 shows a "typical" labeled graph.

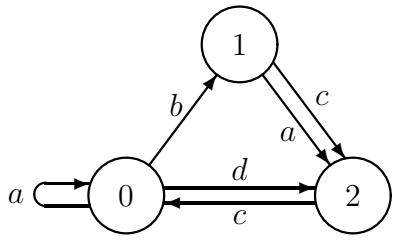


Figure 2.1: Typical labeled graph.

While some of the properties of interest to us do not depend on the labeling L, most do. We will omit the labeling qualifier from the term 'graph' in those cases where the labeling is immaterial.

There are a few features worth highlighting. Since the graph is directed, each edge can be traversed in only one direction, as indicated by the arrow. *Self-loops*, meaning edges that start and terminate in the same state, are allowed. Also, there can be more than one edge connecting a given state to another state; these are called *parallel edges*. However, we assume that distinct edges that share the same initial and terminal states have distinct labels. A graph is called *essential* if every state has at least one outgoing edge and at least one incoming edge; we will sometimes need to assume that graphs are essential, but then we will make this assumption explicitly. The *out-degree* of a state in a graph is the number of edges outgoing from that state. The *minimum out-degree* of a graph is the smallest among all out-degrees of the states in that graph.

A path γ in a graph G is a finite sequence of edges $e_1 e_2 \dots e_\ell$ such that $\sigma_G(e_{i+1}) = \tau_G(e_i)$ for $i = 1, 2, \dots, \ell-1$. The length of a path γ is the number of edges along the path and is denoted by $\ell(\gamma)$. The *state sequence* of a path $e_1 e_2 \dots e_\ell$ is the sequence of states $\sigma_G(e_1)\sigma_G(e_2)\dots\sigma_G(e_\ell)\tau_G(e_\ell)$. A *cycle* in a graph is a path $e_1 e_2 \dots e_\ell$ where $\tau_G(e_\ell) = \sigma_G(e_1)$. We will also use the term right-infinite path for an infinite sequence of edges $e_1 e_2 \cdots$ in G such that $\sigma_G(e_{i+1}) = \tau_G(e_i)$ for $i \geq 1$. Similarly, a bi-infinite path is a bi-infinite sequence of edges $\cdots e_{-1} e_0 e_1 e_2 \cdots$ with $\sigma_G(e_{i+1}) = \tau_G(e_i)$ for all i.

A labeled graph can be used to generate finite symbol sequences by reading off the labels along paths in the graph. A finite sequence of symbols over a given alphabet will be called a *word* or a *block*. The length of a word $\mathbf{w}$—which is the number of symbols in $\mathbf{w}$—will be denoted by $\ell(\mathbf{w})$. A word of length ℓ will be called an *ℓ-block*. If a path γ in a graph G is labeled by a word $\mathbf{w}$, we say that $\mathbf{w}$ is generated by γ (and G). For example, in Figure 2.1, the 5-block *abccd* is generated by the path

$$0 \xrightarrow{a} 0 \xrightarrow{b} 1 \xrightarrow{c} 2 \xrightarrow{c} 0 \xrightarrow{d} 2 \,.$$

We also define the *empty word* as a 0-block: it is generated by a *zero-length path* which consists of one state and no edges. The empty word will be denoted by $\boldsymbol{\epsilon}$. A *sub-word* of a word $\mathbf{w} = w_1 w_2 \dots w_\ell$ is either the empty word or any of the words $w_i w_{i+1} \dots w_j$, where $1 \leq i \leq j \leq \ell$. Such a sub-word is *proper* if $1 < i \leq j < \ell$. Observe that every word that is generated by an essential graph is a proper sub-word of some other word that is generated by that graph.

Let $G_1 = (V_1, E_1, L_1)$ and $G_2 = (V_2, E_2, L_2)$ be labeled graphs. We say that G_1 and G_2 are *(labeled-graph) isomorphic* if there is a one-to-one mapping ψ from V_1 onto V_2 such that $u \xrightarrow{a} v$ is an edge in G_1 if and only if $\psi(u) \xrightarrow{a} \psi(v)$ is an edge in G_2.

The underlying finite directed graph of a labeled graph is conveniently described by a matrix as follows. Let G be a graph. The *adjacency matrix* $A = A_G = \left((A_G)_{u,v} \right)_{u,v \in V_G}$ is

the $|V_G| \times |V_G|$ matrix where the entry $(A_G)_{u,v}$ is the number of edges from state u to state v in G. For instance, the adjacency matrix of the graph in Figure 2.1 is

$$A_G = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 2 \\ 1 & 0 & 0 \end{pmatrix} .$$

The adjacency matrix of course has nonnegative, integer entries. It is a useful artifice; for example, the number of paths of length ℓ from state u to state v is simply $(A_G^\ell)_{u,v}$, and the number of cycles of length ℓ is simply the trace of A_G^ℓ.

The fundamental object considered in the theory of constrained coding is the set of words generated by a labeled graph. A *constrained system* (or *constraint*), denoted S, is the set of all words (i.e., finite sequences) obtained from reading the labels of paths in a labeled graph G (although sometimes we will consider *right-infinite* sequences $x_0x_1x_2\cdots$ and sometimes *bi-infinite* sequences $\cdots x_{-2}x_{-1}x_0x_1x_2\cdots$). We say that G *presents* S or is a *presentation* of S, and we write $S = S(G)$. The *alphabet* of S is the set of symbols that actually occur in words of S and is denoted $\Sigma = \Sigma(S)$.

As central examples of constrained systems, we have the (d, k)*-RLL constrained systems*, which are presented by the labeled graph in Figures 1.3, and the B*-charge constrained systems*, which are presented by the labeled graph in Figure 1.14.

A constrained system is equivalent in automata theory to a regular language which is recognized by an automaton, the states of which are all accepting [Hopc79]. A constrained system is called a sofic system (or sofic shift) in symbolic dynamics [LM95]—except that a sofic system usually refers to the bi-infinite symbol sequences generated by a labeled graph. Earlier expositions on various aspects of constrained systems can be found in [Béal93a], [KN90], [LM95], and [MSW92].

A constrained system should not be confused with any particular labeled graph, because a given constrained system can be presented by many different labeled graphs. For example, the $(0, 1)$-RLL constrained system is presented by all labeled graphs in Figures 2.2 through 2.5, which are very different from one another. This is good: one presentation may be preferable because it has a smaller number of states, while another presentation might be preferable because it could be used as an encoder.

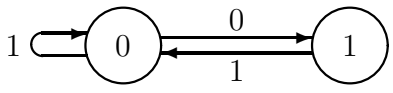


Figure 2.2: Labeled graph for $(0, 1)$-RLL constrained system.

It should be quite clear at this point why we assume that labeled graphs do not contain parallel edges that are labeled the same: the set of words generated by a given graph would not change if such parallel edges were added.

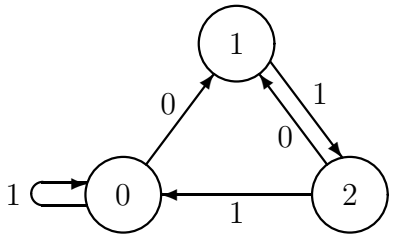


Figure 2.3: Another labeled graph for $(0, 1)$-RLL constrained system.

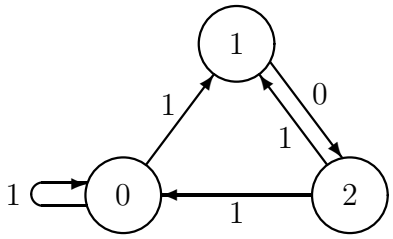


Figure 2.4: Yet another labeled graph for $(0, 1)$-RLL constrained system.

2.2 Properties of labelings

2.2.1 Deterministic presentation

For purposes of encoder construction, it will be important to consider labelings with special properties. The most fundamental special property is as follows.

A labeled graph is *deterministic* if at each state the outgoing edges are labeled distinctly. In other words, at each state, any label generated from that state uniquely determines an outgoing edge from that state. The labeled graphs in Figures 1.3, 1.14, 2.2, and 2.3 are deterministic while the labeled graphs in Figures 2.4 and 2.5 are not. Constrained systems in the engineering literature are usually presented by deterministic graphs. In fact, any

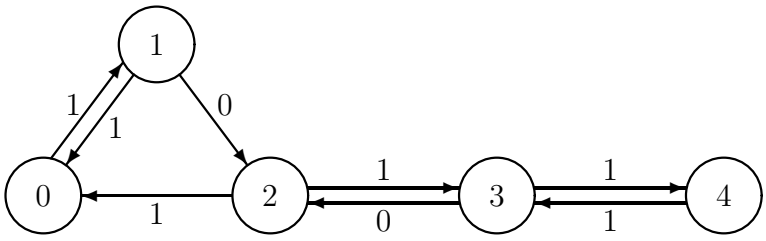


Figure 2.5: One more labeled graph for $(0, 1)$-RLL constrained system.

constrained system can be presented in this way, as we show next.

Let G be a labeled graph. We define the *determinizing graph* H of G in the following manner. For any word $\mathbf{w}$ and state $v \in V_G$, let $T_G(\mathbf{w}, v)$ denote the subset of states in G which are accessible from v by paths in G that generate $\mathbf{w}$. When $\mathbf{w}$ is the empty word $\boldsymbol{\epsilon}$, define $T_G(\boldsymbol{\epsilon}, v) = \{v\}$. The states of H are the distinct nonempty subsets $\{T_G(\mathbf{w}, v)\}_{\mathbf{w},v}$ of V_G. As for the edges of H, for any two states $Z, Z' \in V_H$ we draw an edge $Z \xrightarrow{b} Z'$ in H if and only if there exists a state $v \in V_G$ and a word $\mathbf{w}$ such that $Z = T_G(\mathbf{w}, v)$ and $Z' = T_G(\mathbf{w}b, v)$. In other words, each state of G in Z' is accessible in G from some state in Z by an edge labeled b. By construction, the determinizing graph H is deterministic. We have also the following.

Lemma 2.1 *Let H be the determinizing graph of a labeled graph G. Then $S(H) = S(G)$.*

Proof. If a word $\mathbf{w} = w_1 w_2 \dots w_\ell$ is generated by paths in G starting at state v, then $\mathbf{w}$ is also generated by the path

$$\{v\} = T_G(\boldsymbol{\epsilon}, v) \xrightarrow{w_1} T_G(w_1, v) \xrightarrow{w_2} T_G(w_1 w_2, v) \xrightarrow{w_3} \cdots \xrightarrow{w_\ell} T_G(w_1 w_2 \dots w_\ell, v)$$

in H. Conversely, if $\mathbf{w}$ is generated by H starting at a state $Z = T_G(\mathbf{w}', v)$, then, by the construction of H, $\mathbf{w}'\mathbf{w}$ is generated in G by a path that starts at state v. □

By Lemma 2.1 we can conclude the next result.

Proposition 2.2 *Any constrained system can be presented by some deterministic labeled graph.*

We also have the notion of *co-deterministic*, obtained by replacing "outgoing" with "incoming" in the definition.

'Deterministic' is called *right-resolving* in symbolic dynamics [LM95].

2.2.2 Finite anticipation

Encoder synthesis algorithms usually begin with a deterministic presentation and transform it into a presentation which satisfies the following weaker version of the deterministic property.

A labeled graph G has *finite local anticipation* (or, in short, *finite anticipation*) if there is an integer N such that any two paths of length $N+1$ with the same initial state and labeling must have the same initial edge. The *(local) anticipation* $\mathcal{A}(G)$ of G is the smallest N for which this holds. Hence, knowledge of the initial state of a path and the first $\mathcal{A}(G)+1$

symbols that it generates is sufficient information to determine the initial edge of the path. In case G does not have finite anticipation, we define $\mathcal{A}(G) = \infty$.

We also define the *(local) co-anticipation* of a labeled graph G as the anticipation of the labeled graph obtained by reversing the directions of the edges in G.

Note that to say that a labeled graph is deterministic is to say that it has zero anticipation. The labeled graph in Figure 2.4 is a presentation of the $(0, 1)$-RLL constrained system with anticipation 1 but not 0. Figure 2.5 depicts a presentation that does not have finite anticipation.

'Finite anticipation' is also called *right-closing* (in symbolic dynamics [LM95]) or *lossless of finite order* [Huff59], [Even65].

2.2.3 Finite memory

A labeled graph G is said to have *finite memory* if there is an integer N such that the paths in G of length N that generate the same word all terminate in the same state. The smallest N for which this holds is called the *memory* of G and denoted $\mathcal{M}(G)$.

2.2.4 Definite graphs

A labeled graph is (m, a)-*definite* if, given any word $\mathbf{w} = w_{-\mathsf{m}} w_{-\mathsf{m}+1} \dots w_0 \dots w_{\mathsf{a}}$, the set of paths $e_{-\mathsf{m}} e_{-\mathsf{m}+1} \dots e_0 \dots e_{\mathsf{a}}$ that generate $\mathbf{w}$ all agree in the edge e_0. We say that a labeled graph is definite if it is (m, a)-definite for some finite nonnegative m and a. Definite graphs are referred to in the literature also as graphs with *finite memory-and-anticipation*.

Note the difference between this concept and the concept of finite anticipation: we have replaced knowledge of an initial state with knowledge of a finite amount of memory. Actually, definiteness is a stronger condition, as we show in Proposition 2.3.

Figure 2.3 shows a labeled graph that is $(2, 0)$-definite, while Figure 2.6 shows a labeled graph that has finite anticipation (in fact, is deterministic and co-deterministic) but is not definite.

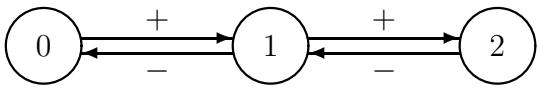


Figure 2.6: Labeled graph for a 2-charge constrained system

Note that, in contrast to the anticipation and the memory, we did not require a and m to be minimal in any sense while talking about (m, a)-definiteness. It would be natural to

require that m+a be minimal, but even that does not specify m and a uniquely; for instance, the labeled graph in Figure 2.7 is $(1,0)$-definite and also $(0,1)$-definite.

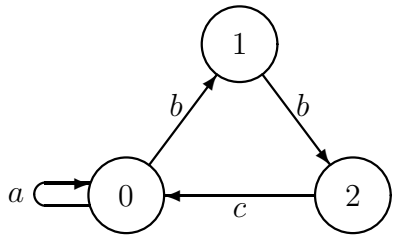


Figure 2.7: Labeled graph which is both $(1,0)$-definite and $(0,1)$-definite.

2.2.5 Lossless graphs

A labeled graph is *lossless* if any two distinct paths with the same initial state and terminal state have different labelings. All of the pictures of labeled graphs that we have presented so far are lossless. Figure 2.8 shows a presentation of the $(0,1)$-RLL constrained system that is not lossless.

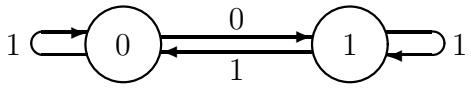


Figure 2.8: Graph which is not lossless.

2.2.6 Summary of terms

The following proposition summarizes the relationships among the labeling properties introduced so far.

Proposition 2.3 *For essential graphs,*

$$\begin{array}{ccccc} & & & & \textit{Co-deterministic} \\ & & & & \Downarrow \\ \textit{Finite memory} & \Rightarrow & \textit{Definite} & \Rightarrow & \textit{Finite co-anticipation} \\ \Downarrow & & \Downarrow & & \Downarrow \\ \textit{Deterministic} & \Rightarrow & \textit{Finite anticipation} & \Rightarrow & \textit{Lossless} \end{array}$$

Proof. *Finite memory* $\Rightarrow$ *Deterministic* and *Finite memory* $\Rightarrow$ *Definite:* Let G be a labeled graph with finite memory $\mathcal{M}$. All paths, representing the same word, of length $\mathcal{M}+1$ in G must agree in their last two states. Furthermore, since G does not contain parallel edges with the same label, all these paths agree in their last edge. Hence, G is $(\mathcal{M}, 0)$-definite. This also implies that G is deterministic: For a state u in G, let γ be a path of length $\mathcal{M}$ that terminates in u. Then for every edge e outgoing from u, the labeling of γe determines e.

Definite $\Rightarrow$ *Finite anticipation:* Suppose that G is (m, a)-definite for some m and a. Let u be a state in G and let $\gamma = e_0 e_1 \dots e_{\mathsf{a}}$ and $\gamma' = e'_0 e'_1 \dots e'_{\mathsf{a}}$ be paths of length $\mathsf{a}+1$ which start at u and generate the same word. We need to show that $e_0 = e'_0$. Let γ'' be any path of length m which terminates in state u. Then, the concatenated paths $\gamma''\gamma$ and $\gamma''\gamma'$ both generate the same word. Since G is (m, a)-definite, we have $e_0 = e'_0$ as desired. A similar proof yields the implication *Definite* $\Rightarrow$ *Finite co-anticipation.*

Deterministic $\Rightarrow$ *Finite anticipation:* As pointed out earlier, a labeled graph is deterministic if and only if it has zero anticipation. The implication *Co-deterministic* $\Rightarrow$ *Finite co-anticipation* is similar.

Finite anticipation $\Rightarrow$ *Lossless:* Let G have anticipation $\mathcal{A}$. Given two paths, γ and γ', with the same initial state u, terminal state v, and the same labeling $\mathbf{w}$, let γ'' be a path of length $\mathcal{A}$ which starts at v. Then $\gamma\gamma''$ and $\gamma'\gamma''$ start at the same state and generate the same word; so, $\gamma = \gamma'$, as desired. The implication *Finite co-anticipation* $\Rightarrow$ *Lossless* is proved in a similar way. □

2.2.7 State labeling

In the graph presentations that we have seen so far, the labels are put on the *edges*. However, in the literature, one can find graph presentations where the labels are put on the *states*, and the respective constrained system is defined as the set of words that are obtained by reading off the labels of states along the finite paths in the graph. It is straightforward to see that every such constrained system can be presented by an (edge-)labeled graph, where the incoming edges to each state all have the same label. In fact, every constrained system can be presented by such a labeled graph, as we now show.

Let G be a labeled graph. The *Moore form* of G is a labeled graph H where $V_H = E_G$ and $e_1 \xrightarrow{a} e_2$ is an edge in H if and only if $\tau_G(e_1) = \sigma_G(e_2)$ and $L_G(e_2) = a$. For example, Figure 2.3 shows the Moore form of the labeled graph in Figure 2.2 that presents the $(0, 1)$-RLL constrained system. It can be easily verified that $S(H) = S(G)$ and that the edges incoming to each state in H all have the same labeling. It thus follows that every constrained system can be presented by a state-labeled graph. The anticipation of G is preserved in H and the co-anticipation is increased by 1. In particular, if G is deterministic, so is its Moore form. Also, by construction, there are no parallel edges in a Moore form, so its adjacency

matrix is always a 0–1 matrix.

We have also the notion of a *Moore co-form* of a labeled graph G which is identical to the Moore form except for the labeling: The edge $e_1 \stackrel{a}{\rightarrow} e_2$ in a Moore co-form inherits the labeling of e_1 in G, rather than that of e_2. For example, Figure 2.4 is the Moore co-form of the labeled graph of Figure 2.2. If H is a Moore co-form of a labeled graph G, then $S(H) = S(G)$ and the edges *outgoing* from each state in H all have the same labeling. The co-anticipation of G is preserved in H and the anticipation is increased by 1. Therefore, if G is co-deterministic, so is H.

2.3 Finite-type constraints

In this section, we consider some special classes of constraints. The properties that define these constraints will be useful for encoder construction.

A constrained system S is *finite-type* (in symbolic dynamics, *shift of finite type* [LM95]) if it can be presented by a definite graph. As an example, the (d, k)-RLL constraint is finite-type: the labeled graph in Figure 1.3 is $(k, 0)$-definite—i.e., for any given word $\mathbf{w}$ of length at least $k+1$, all paths that generate $\mathbf{w}$ end with the same edge.

It is important to recognize that there are "bad" presentations of finite-type constrained systems, meaning labeled graphs that are not definite. For example, the labeled graph in Figure 2.5 represents the $(0, 1)$-RLL constrained system, but it is not definite, as can be seen by considering the paths that generate words consisting of all 1's.

Given the existence of bad labeled graphs, one might begin to worry about potential problems in determining whether or not a constrained system is finite-type. However, there is an intrinsic characterization of finite-type constrained systems that resolves this difficulty.

A constrained system S is said to have *finite memory* if there is an integer N such that, for any symbol $b \in \Sigma(S)$ and any word $\mathbf{w} \in S$ of length at least N, we have $\mathbf{w}b \in S$ if and only if $\mathbf{w}'b \in S$ where $\mathbf{w}'$ is the suffix of $\mathbf{w}$ of length N. The smallest such integer N, if any, is called the *memory* of S and is denoted by $\mathcal{M}(S)$.

It is can be readily verified that the (d, k)-RLL constrained system has memory k.

Lemma 2.4 *A constrained system S has finite memory if and only if there is a presentation G of S with finite memory. Furthermore, the memory of S is the smallest memory of any presentation of S with finite memory.*

Proof. Clearly, if a constrained system S has a presentation G with finite memory $\mathcal{M}(G)$, then $\mathcal{M}(S) \leq \mathcal{M}(G)$.

On the other hand, let S be a constrained system with finite memory $\mathcal{M}(S) = \mathcal{M}$. Then all presentations of S have memory which is bounded from below by $\mathcal{M}$. We construct a labeled graph H with $\mathcal{M}(H) = \mathcal{M}$ as follows: For each word $\mathbf{w}$ of length $\mathcal{M}$ in S, we associate a state $u_{\mathbf{w}}$ in H. Given two words, $\mathbf{w} = w_1 w_2 \dots w_{\mathcal{M}}$ and $\mathbf{z} = z_1 z_2 \dots z_{\mathcal{M}}$, in S, we draw an edge $u_{\mathbf{w}} \stackrel{b}{\rightarrow} u_{\mathbf{z}}$ in H if and only if the following three conditions hold:

(a) $z_j = w_{j+1}$ for $j = 1, 2, \dots, \mathcal{M}-1$;

(b) $b = z_{\mathcal{M}}$;

(c) $\mathbf{w}b \in S$.

It is easy to verify that H is a presentation of S (and, so $\mathcal{M}(H) \geq \mathcal{M}$). On the other hand, the paths in H of length $\mathcal{M}$ that generate the word $\mathbf{w}$ all terminate in state $u_{\mathbf{w}}$. Hence, $\mathcal{M}(H) \leq \mathcal{M}$. □

Proposition 2.5 *A constrained system is finite-type if and only if it has finite memory.*

Proof. Suppose that S has finite memory and let G be a presentation of S with finite memory $\mathcal{M}$. As such, G is also $(\mathcal{M}, 0)$-definite and, so, S is finite-type.

Now, suppose that S is finite-type and let G be an (m, a)-definite presentation of S. If $\mathsf{a} = 0$, then G has memory $\leq \mathsf{m}{+}1$ and we are done. When $\mathsf{a} > 0$, it suffices to find a presentation of S which is $(\mathsf{m}{+}\mathsf{a}, 0)$-definite.

Such a presentation H can be obtained as follows: The states of H are pairs $[u, \mathbf{w}]$, where $u \in V_G$ and $\mathbf{w}$ is a word of length a that can be generated by a path in G that starts at u. Let u and v be states in G and $\mathbf{w} = w_1 w_2 \dots w_{\mathsf{a}}$ and $\mathbf{z} = z_1 z_2 \dots z_{\mathsf{a}}$ be two words that can be generated in G from u and v, respectively. We draw an edge $[u, \mathbf{w}] \stackrel{b}{\rightarrow} [v, \mathbf{z}]$ in H if and only if the following three conditions hold:

(a) $z_j = w_{j+1}$ for $j = 1, 2, \dots, \mathsf{a}{-}1$;

(b) $b = z_{\mathsf{a}}$;

(c) there is an edge $u \stackrel{w_1}{\rightarrow} v$ in G.

We now define a mapping from the set of all paths of length $\mathsf{m}{+}\mathsf{a}{+}1$ in G onto the set of paths of length $\mathsf{m}{+}1$ in H as follows. The path

$$\gamma_G = u_0 \stackrel{b_1}{\longrightarrow} \cdots \stackrel{b_{\mathsf{m}}}{\longrightarrow} u_{\mathsf{m}} \stackrel{b_{\mathsf{m}+1}}{\longrightarrow} u_{\mathsf{m}+1} \stackrel{b_{\mathsf{m}+2}}{\longrightarrow} \cdots \stackrel{b_{\mathsf{m}+\mathsf{a}}}{\longrightarrow} u_{\mathsf{m}+\mathsf{a}} \stackrel{b_{\mathsf{m}+\mathsf{a}+1}}{\longrightarrow} u_{\mathsf{m}+\mathsf{a}+1}$$

in G is mapped to the path

$$\gamma_H = [u_0, b_1 b_2 \dots b_{\mathsf{a}}] \stackrel{b_{\mathsf{a}+1}}{\longrightarrow} \cdots \stackrel{b_{\mathsf{m}+\mathsf{a}}}{\longrightarrow} [u_{\mathsf{m}}, b_{\mathsf{m}+1} b_{\mathsf{m}+2} \dots b_{\mathsf{m}+\mathsf{a}}] \stackrel{b_{\mathsf{m}+\mathsf{a}+1}}{\longrightarrow} [u_{\mathsf{m}+1}, b_{\mathsf{m}+2} b_{\mathsf{m}+3} \dots b_{\mathsf{m}+\mathsf{a}+1}]$$

in H. It is easy to verify that this mapping is indeed onto. Since G is (m, a)-definite, the word $b_1 b_2 \dots b_{\mathsf{m}+\mathsf{a}+1}$ uniquely defines the edge $u_{\mathsf{m}} \xrightarrow{b_{\mathsf{m}+1}} u_{\mathsf{m}+1}$ in the path γ_G in G. It thus follows that the last two states in γ_H are uniquely defined, and so is the last edge of γ_H. Hence, H is $(\mathsf{m}+\mathsf{a}+1, 0)$-definite. □

The following result gives another equivalent formulation of the notion of finite-type systems in terms of lists of forbidden words. This notion was alluded to at the end of Section 1.2 and in Section 1.5.2.

Proposition 2.6 *A constrained system S is finite-type if and only if there is a finite list $\mathcal{L}$ of words such that $\mathbf{w} \in S$ if and only if $\mathbf{w}$ does not contain any word of $\mathcal{L}$ as a sub-word.*

We leave the proof of Proposition 2.6 as an exercise for the reader (Problem 2.7).

Not every constrained system of interest is finite-type. For example, the 2-charge constrained system described by Figure 2.6 is not. This can be seen easily by considering the condition above: the symbol '+' can be appended to the word

$$-+-+-+\cdots-+$$

but not to the word

$$++-+-+-+\cdots-+\,.$$

As a second example, consider the $(0, \infty, 2)$-RLL constrained system, which is commonly referred to as the *even* constraint. This constrained system consists of all binary words in which the runs of 0's between successive 1's have even lengths. A graph presentation of this constraint is shown in Figure 2.9. We leave it as an exercise to show that this constraint is

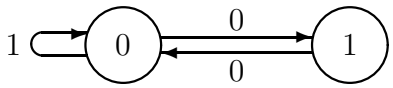


Figure 2.9: Shannon cover of the even constrained system.

not finite-type (Problem 2.26, part 1).

However, both the charge constraint and the even constraint fall into a natural broader class of constrained systems, called almost-finite-type systems; these systems should be thought of as "locally finite-type" (perhaps that would have been a better name). A constrained system is *almost-finite-type* if it can be presented by a labeled graph that has both finite anticipation and finite co-anticipation.

By Proposition 2.3, we know that definiteness implies finite anticipation and finite co-anticipation. Thus, every constrained system which is finite-type is also almost-finite-type,

and so the almost-finite-type systems do indeed include the finite-type systems. From Figure 2.6, we see that the charge constrained systems are presented by labeled graphs with zero anticipation (i.e., deterministic) and zero co-anticipation (i.e., co-deterministic); thus, these systems are almost-finite-type, but not finite-type. Most constrained systems used in practical applications are in fact almost-finite-type.

Recall that every constrained system has a deterministic presentation (and hence finite anticipation); likewise, every constrained system has a co-deterministic presentation (and hence finite co-anticipation). So, the essential feature of the almost-finite-type definition is that there is a presentation that simultaneously has finite anticipation and finite co-anticipation.

As with finite-type systems, we have the problem that a given constrained system may have some presentation that satisfies the finite anticipation and co-anticipation conditions and another presentation that does not. There is an intrinsic condition that defines almost-finite-type, but it is a bit harder to state [Will88]. We will give an example of a constrained system which is not almost-finite-type at the end of Section 2.6.

2.4 Some operations on graphs

In this section, we introduce three graph constructions that create new constraints from old.

2.4.1 Power of a graph

As mentioned in Chapter 1, a rate $p : q$ finite-state encoder will generate a word, composed of q-codewords (q-blocks) that when hooked together belong to the desired constrained system S. For a constrained system S presented by a labeled graph G, it will be very useful to have an explicit description of the words in S, decomposed into such non-overlapping "chunks" of length q.

Let G be a labeled graph. The *qth power of G*, denoted G^q, is the labeled graph with the same set of states as G, but one edge for each path of length q in G, labeled by the q-block generated by that path. For a constrained system S presented by a labeled graph G, the *qth power of S*, denoted S^q, is the constrained system presented by G^q. So, S^q is the constrained system obtained from S by grouping the symbols in each word into non-overlapping "chunks" of length q (in particular, the definition of S^q does not depend on which presentation G of S is used).

For example, Figure 2.10 shows the third power G^3 of the labeled graph G in Figure 2.2 that presents the $(0, 1)$-RLL constrained system.

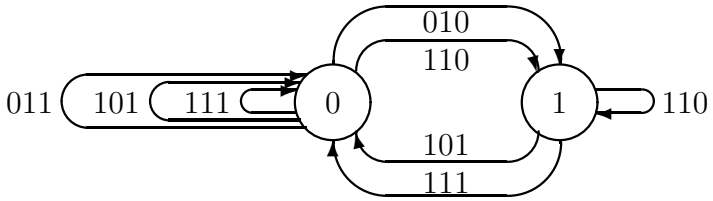


Figure 2.10: Third power of labeled graph in Figure 2.2.

2.4.2 Higher edge graph

The qth *higher edge graph* $G^{[q]}$ is the labeled graph whose states are paths in G of length $q-1$ with an edge for each path of length q in G: the edge $e_1e_2 \dots e_q$ has initial state $e_1e_2 \dots e_{q-1}$, terminal state $e_2 \dots e_q$, and inherits the labeling of $e_1e_2 \dots e_q$. For a constrained system S presented by a labeled graph G, the qth *higher order system of* S, denoted $S^{[q]}$, is the constrained system presented by $G^{[q]}$.

Observe that $S^{[q]}$ is the constrained system whose alphabet is the set of q-blocks of S, obtained from S by replacing each word $w_1w_2 \dots w_\ell$ by the word

$$(w_1w_2 \dots w_q)(w_2w_3 \dots w_{q+1}) \dots (w_{\ell-q+1}w_{\ell-q+2} \dots w_\ell) \ .$$

Note how S^q differs from $S^{[q]}$: the former divides words into non-overlapping blocks; the latter divides words into blocks which overlap by $q-1$ symbols.

Figure 2.11 shows the edge graph $G^{[2]}$ for the $(0,1)$-RLL labeled graph G in Figure 2.2, and $G^{[3]}$ is shown in Figure 2.12. The reader should contrast this with the third power G^3 in Figure 2.10.

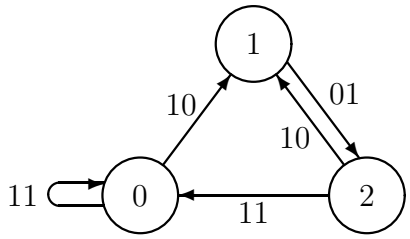


Figure 2.11: Second higher edge graph of labeled graph in Figure 2.2.

The Moore form and co-form of G which were introduced in Section 2.2.7 are almost identical to $G^{[2]}$: To obtain the Moore form (respectively, the Moore co-form), just delete the first (respectively, the second) symbol in each edge label of $G^{[2]}$.

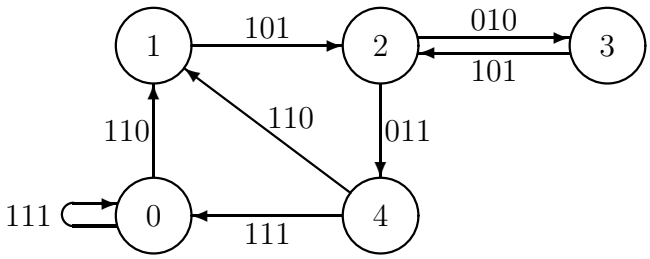


Figure 2.12: Third higher edge graph of labeled graph in Figure 2.2.

2.4.3 Fiber product of graphs

Let G and H be two labeled graphs. We define the *fiber product* of G and H as the labeled graph $G * H$, where

$$V_{G*H} = V_G \times V_H = \{\langle u, u' \rangle \,|\, u \in V_G,\ u' \in V_H\}\ ,$$

and $\langle u, u' \rangle \xrightarrow{a} \langle v, v' \rangle$ is in E_{G*H} if and only if $u \xrightarrow{a} v \in E_G$ and $u' \xrightarrow{a} v' \in E_H$. It is easy to verify that the fiber product presents the intersection of the constraints defined by G and H, namely, $S(G * H) = S(G) \cap S(H)$.

Finally, we state a result which asserts that the operations introduced in this section all preserve the properties of labelings introduced in Section 2.2. We leave the proof to the reader.

Proposition 2.7 *The power of a graph, higher edge graph, and fiber product graph all preserve the deterministic, finite anticipation (co-anticipation), and definiteness properties.*

2.5 Irreducibility

2.5.1 Irreducible graphs

A graph G is *irreducible* (or *strongly-connected*) if, for any ordered pair of states u, v, there is a path from u to v in G. A graph is *reducible* if it is not irreducible. Note our use of the term 'ordered': for a given pair of states u, v, we must be able to travel from u to v and from v to u.

All of the graphs in Figures 2.2 through 2.5 are irreducible, while Figure 2.13 shows a reducible graph which presents the system of unconstrained binary words.

Observe that the property of being irreducible does not depend on the labeling and can

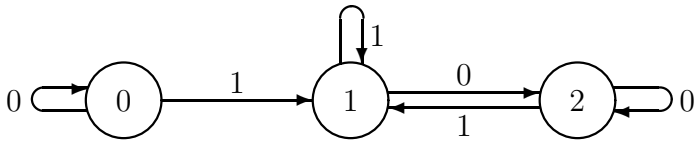


Figure 2.13: Reducible labeled graph for unconstrained binary words.

be described in terms of the adjacency matrix: namely, for every (ordered) pair of states u, v, there exists some ℓ such that $(A_G^\ell)_{u,v} > 0$.

It will be useful later to know that any reducible graph can, in some sense, be broken down into "maximal" irreducible pieces. To make this more precise we introduce the concept of an irreducible component. An *irreducible component* of a graph G is a maximal (with respect to inclusion) irreducible subgraph of G. The irreducible components of a graph are simply the subgraphs consisting of all edges whose initial and terminal states both belong to an equivalence class of the following relation: $u \sim v$ if there is a path from u to v and a path from v to u (we allow paths to be empty so that $u \sim u$).

An *irreducible sink* is an irreducible component H such that any edge which originates in H must also terminate in H. An *irreducible source* is an irreducible component H such that any edge which terminates in H must also originate in H.

Any graph can be broken down into irreducible components with 'transient' connections between the components. The irreducible sinks can have transient connections entering but not exiting. Every graph has at least one irreducible sink (and, similarly, at least one irreducible source). To see this, we argue as follows. Pick an irreducible component and check if it is an irreducible sink. If so, stop. If not, there must be a path leading to another irreducible component. Repeat the procedure on the latter component. The process must eventually terminate in an irreducible sink H; otherwise, the original decomposition into irreducible components would be contradicted. The picture of the irreducible components and their connections is perhaps best illustrated via the adjacency matrix: by reordering the states, $A = A_G$ can be written in block upper triangular form with the adjacency matrices of the irreducible components as block diagonals, as shown in Figure 2.14.

$$A = \begin{pmatrix} A_1 & B_{1,2} & B_{1,3} & \cdots & B_{1,k} \\ & A_2 & B_{2,3} & \cdots & B_{2,k} \\ & & A_3 & \ddots & \vdots \\ & & & \ddots & B_{k-1,k} \\ & & & & A_k \end{pmatrix}.$$

Figure 2.14: Writing matrix in upper triangular form.

Figure 2.15 shows the irreducible components of the graph in Figure 2.13; one is an

irreducible sink and the other is an irreducible source.

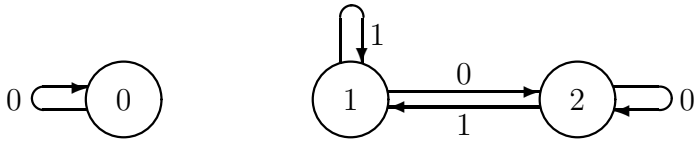


Figure 2.15: Irreducible components of labeled graph in Figure 2.13.

From the point-of-view of finite-state encoder construction, it turns out that, by passing to irreducible components, we can concern ourselves primarily with irreducible labeled graphs; we explain why in Section 4.1.

There are times when the qth power of a graph G will not be irreducible, even when G is. For example, Figure 2.6 shows a labeled graph describing a 2-charge constrained system. Its second power G^2, shown in Figure 2.16, has two irreducible components, G_0 and G_1 (note that in these graphs, the label $+-$ is different from $-+$). This example illustrates the general situation: it can be shown that, if G is an irreducible graph, then any power G^q is either irreducible or decomposes into isolated, irreducible components (see also Figures 2.2 and 2.10). We elaborate upon this in Section 3.3.2.

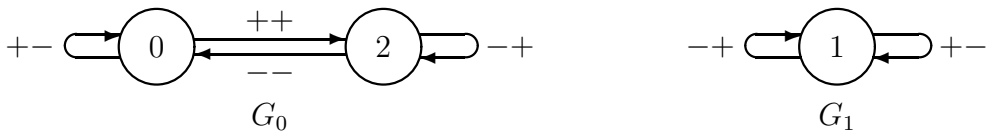


Figure 2.16: Second power of labeled graph in Figure 2.6.

2.5.2 Irreducible constrained systems

A constrained system S is *irreducible*, if for every pair of words $\mathbf{w}, \mathbf{w}'$ in S, there is a word $\mathbf{z}$ such that $\mathbf{wzw}'$ is in S. A constrained system that is not irreducible is called *reducible*.

The following shows that irreducibility of a constrained system can be reformulated in terms of irreducible labeled graphs.

Lemma 2.8 *Let S be a constrained system. The following are equivalent:*

(a) S is irreducible;

(b) S is presented by some irreducible (and in fact, deterministic) labeled graph.

Proof. For (b) $\Rightarrow$ (a), simply connect the terminal state of a path that generates $\mathbf{w}$ to the initial state of a path that generates $\mathbf{w}'$. For (a) $\Rightarrow$ (b), replace inclusion with equality in the stronger statement of the next lemma. $\square$

Lemma 2.9 *Let S be an irreducible constrained system and let G be a labeled graph such that $S \subseteq S(G)$. Then for some irreducible component G' of G, $S \subseteq S(G')$.*

Proof. Let $G_1, G_2, \ldots, G_k$ denote the irreducible components of G. We prove the lemma by contradiction. Suppose that for each $i = 1, 2, \ldots, k$, there is a word $\mathbf{w}_i$ in S but not in $S(G_i)$. Since S is irreducible, there is a word $\mathbf{w}$ that contains a copy of each $\mathbf{w}_i$; moreover, there is a word $\mathbf{z}$ that contains $|V_G|+1$ non-overlapping copies of $\mathbf{w}$. Let γ be a path in G that generates $\mathbf{z}$. Then γ can be written as $\gamma = \gamma_1\gamma_2 \ldots \gamma_{|V_G|+1}$, where each γ_j has a sub-path which generates $\mathbf{w}$. For some $r < t$, the initial states of γ_r and γ_t coincide and, therefore, $\gamma_r\gamma_{r+1} \ldots \gamma_{t-1}$ is a cycle and has a sub-path that generates $\mathbf{w}$. Now, by definition, any cycle in a graph must belong to some irreducible component, say G_i, and thus $\mathbf{w}_i$ is in $S(G_i)$, contrary to the definition of $\mathbf{w}_i$. $\square$

All of the constrained systems that we have considered so far are irreducible, while Figure 2.17 presents a reducible constrained system.

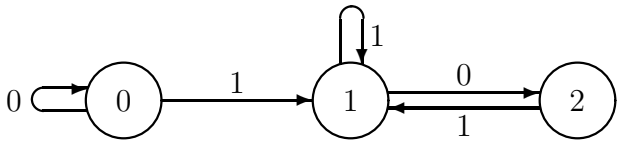


Figure 2.17: Reducible constrained system.

2.6 Minimal presentations

When treating constrained systems, it is useful to present them in a standard manner. Among the various possible presentations of a given constrained system S, the Shannon cover is usually chosen as the canonical presentation of S.

A *Shannon cover* of a constrained system S is a deterministic presentation of S with a smallest number of states.

In general, the Shannon cover is not unique [Jon95], [LM95, Section 3.3]. However, it is unique, up to labeled graph isomorphism, for irreducible constrained systems. We show this in Theorem 2.12 below.

2.6.1 Follower sets and reduced labeled graphs

Let u be a state in a labeled graph G. The *follower set* of u in G, denoted $\mathcal{F}_G(u)$, is the set of all (finite) words that can be generated from u in G. Two states u and u' in a labeled graph G are said to be *follower-set equivalent*, denoted $u \simeq u'$, if they have the same follower set. It is easy to verify that follower-set equivalence satisfies the properties of an equivalence relation.

A labeled graph G is called *reduced* if no two states in G are follower-set equivalent. If a labeled graph G presents a constrained system S, we can construct a reduced labeled graph H from G that presents the same constrained system S by merging states in G which are follower-set equivalent. More precisely, each equivalence class C of follower-set equivalent states becomes a state in H, and we draw an edge $C \stackrel{a}{\rightarrow} C'$ in H if and only if there exists an edge $u \stackrel{a}{\rightarrow} u'$ in G for states $u \in C$ and $u' \in C'$. It is easy to verify that, indeed, $S(H) = S(G)$, and, if G is deterministic, so is H; see [LM95].

2.6.2 The Moore algorithm

The Moore algorithm is an efficient procedure for finding the equivalence classes of the follower-set equivalence relation of states in a deterministic graph. The algorithm is described in [Huff54], [Koh78, Ch. 10], [Moore56].

Let G be a deterministic graph. For a state u in G, let $\mathcal{F}_G^\ell(u)$ denote the set of all words of length ℓ that can generated from u in G. Two states u and v in G are said to be *ℓ-(follower-set-)equivalent* in G, if $\mathcal{F}_G^m(u) = \mathcal{F}_G^m(v)$ for $m = 1, 2, \ldots, \ell$. Indeed, ℓ-equivalence is an equivalence relation, and we denote by Π_ℓ the partition of V_G which is induced by the classes of this relation. Also, we denote by $|\Pi_\ell|$ the number of classes in Π_ℓ.

The Moore algorithm iteratively finds the partitions Π_ℓ for increasing values of ℓ, until we reach the partition induced by the follower-set equivalence relation. The partition Π_0 contains one class, namely, V_G. As for 1-equivalence, two states u and v belong to the same equivalence class if and only if the sets of labels of the edges outgoing from state u and from state v are the same. Therefore, the partition Π_1 is easily found from G.

The following is a typical iteration of the Moore algorithm. Assume we have found Π_ℓ for some $\ell \geq 1$. Now, every two $(\ell+1)$-equivalent states in G must also be ℓ-equivalent. Hence, $\Pi_{\ell+1}$ is a *refinement* of Π_ℓ (and so $|\Pi_{\ell+1}| \geq |\Pi_\ell|$). More specifically, we put two states u and v in the same class in $\Pi_{\ell+1}$ if and only if (i) u and v belong to the same class in Π_ℓ, and (ii) for each pair of edges, $u \stackrel{a}{\rightarrow} u'$ and $v \stackrel{a}{\rightarrow} v'$, in G (with the same label a), the states u' and v' belong to the same class in Π_ℓ.

Example 2.1 Consider the deterministic graph G in Figure 2.18. We start with the

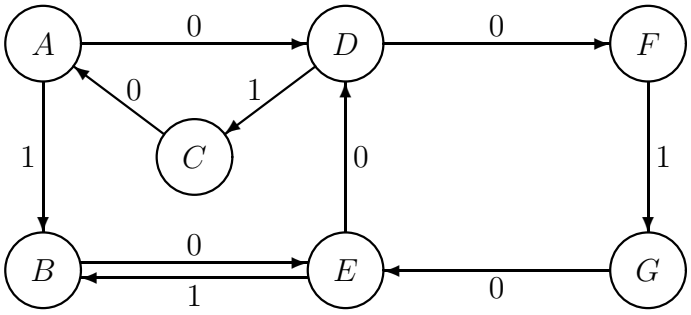


Figure 2.18: Graph G for Example 2.1.

trivial partition, Π_0, which consists of one equivalence class that contains all states, namely,

$$\Pi_0 = \{A, B, C, D, E, F, G\} .$$

The partition Π_1 is obtained by looking at the set of labels of the outgoing edges from each state. That set is $\{0\}$ for states B, C, and G; it is $\{1\}$ for state F; and it is $\{0, 1\}$ for states A, D, and E. Hence,

$$\Pi_1 = \{B, C, G\}\{F\}\{A, D, E\} .$$

To obtain the partition Π_2, we see that the outgoing edges (labeled 0) from states B, C, and G terminate in E, A, and E, respectively, and these terminal states belong to the same equivalence class in Π_1. Hence, $\{B, C, G\}$ will form an equivalence class also in the partition Π_2. On the other hand, the outgoing edges labeled 0 from A, D, and E terminate in D, F, and D, respectively, thus implying that state D should be separated from states A and E in Π_2. Since the outgoing edges labeled 1 from A and E both terminate in state B, we conclude that A and E are 2-equivalent and, so,

$$\Pi_2 = \{B, C, G\}\{F\}\{A, E\}\{D\} .$$

The next iteration will generate no refinement i.e., we end up with $\Pi_3 = \Pi_2$. □

In general, the algorithm finds the partitions Π_ℓ for increasing values of ℓ until $\Pi_{\ell+1} = \Pi_\ell$. Denote by $\ell_{\max}$ the smallest ℓ for which this equality is met. For $\ell > \ell_{\max}$ we will have $\Pi_\ell = \Pi_{\ell_{\max}}$ and, so, the follower-set equivalence relation is identical to the $\ell_{\max}$-equivalence relation. Furthermore,

$$1 = |\Pi_0| < |\Pi_1| < \cdots < |\Pi_{\ell_{\max}-1}| < |\Pi_{\ell_{\max}}| = |\Pi_{\ell_{\max}+1}| \leq |V_G| .$$

Therefore, $\ell_{\max} \leq |V_G|-1$, which thus bounds from above the number of iterations in the Moore algorithm. In fact, we have also the following.

Proposition 2.10 *Let G be a deterministic essential graph. Then, for every pair of states u and v in G,*

$$\mathcal{F}_G(u) = \mathcal{F}_G(v) \quad \textit{if and only if} \quad \mathcal{F}_G^{|V_G|-1}(u) = \mathcal{F}_G^{|V_G|-1}(v) \; .$$

.

Having found the partition $\Pi_{\ell_{\max}}$ induced by the follower-set equivalence relation, we can construct a reduced deterministic graph H from G that presents the same constrained system $S(G)$, as was described in Section 2.6.1.

If we apply the reduction to the deterministic graph in Example 2.1, we obtain the graph presentation of the $(1,3)$-RLL constraint as shown in Figure 1.16, with the following equivalence relation between the states in that figure and the partition elements of Π_2:

$$0 \longleftrightarrow \{B, C, G\} \; , \quad 1 \longleftrightarrow \{A, E\} \; , \quad 2 \longleftrightarrow \{D\} \; , \quad \text{and} \quad 3 \longleftrightarrow \{F\} \; .$$

2.6.3 Homing words

A *homing word* for a state v in a labeled graph G is a word $\mathbf{w}$ such that all paths in G that generate $\mathbf{w}$ terminate in v.

We will make use of the following lemma.

Lemma 2.11 *Let G be a reduced deterministic graph. Then there is a homing word for at least one state in G. Furthermore, if G is also irreducible, then there is a homing word for every state in G.*

Proof. Since G is reduced, for any two distinct states $u, v \in V_G$, there is a *distinguishing word* $\mathbf{w}$ that can be generated in G from one of these states but not from the other. Start with the set $V_0 = V_G$ and assume that $|V_0| > 1$. Let $\mathbf{w}_1$ be a distinguishing word for some pair of states in V_0. Clearly, $\mathbf{w}_1$ can be generated in G by at most $|V_0| - 1$ paths starting in V_0. Let V_1 denote the set of terminal states of these paths. We can now apply the same argument on V_1 with a distinguishing word $\mathbf{w}_2$ for two states in V_1 and continue this way until we end up with a set $V_m = \{v\}$. The word $\mathbf{w}_1\mathbf{w}_2 \ldots \mathbf{w}_m$ is a homing word for v in G.

If G is also irreducible, then we can extend the homing word for v to a homing word for every state in G. □

We can use the notion of homing words to obtain the following equivalent definition for labeled graphs with finite memory: a labeled graph G has finite memory if there is an integer N such that all words of length N in $S(G)$ are homing (and the memory of G is then the smallest such N).

2.6.4 Shannon cover of irreducible constrained systems

The following result summarizes the main properties of the Shannon cover of irreducible constrained systems. See [Fi75a], [Fi75b], [KN90].

Theorem 2.12 *Let S be an irreducible constrained system.*

(a) The Shannon cover of S is unique, up to labeled graph isomorphism. In fact, the Shannon cover is the unique presentation of S which is irreducible, deterministic, and reduced.

(b) For any irreducible deterministic presentation G of S, the follower sets of G coincide with the follower sets of the Shannon cover.

Proof. First, note that any minimal deterministic presentation of S must be reduced. Otherwise we could merge all states with the same follower sets, as was described in Section 2.6.1. Furthermore, by Lemma 2.9, any minimal deterministic presentation must be irreducible.

For the proof of (a), suppose that H and H' are irreducible reduced deterministic graphs that present S. Let u be an arbitrary state in H. By Lemma 2.11, there is a homing word $\mathbf{w}$ for u in H and a homing word $\mathbf{w}'$ for some state in H'. Since S is irreducible, there exists a word $\mathbf{z}$ such that $\mathbf{w}'' = \mathbf{w}'\mathbf{z}\mathbf{w} \in S$. Clearly, $\mathbf{w}''$ is a homing word for the state u in H and for some state u' in H'. Since $S(H) = S(H') = S$, we must have $\mathcal{F}_H(u) = \mathcal{F}_{H'}(u')$. Furthermore, since both H and H' are deterministic, the 'outgoing picture' from state u in H must be the same as that from state u' in H'. Hence, if $u \xrightarrow{a} v \in E_H$, then we must have $u' \xrightarrow{a} v' \in E_{H'}$ and $\mathcal{F}_H(v) = \mathcal{F}_{H'}(v')$. Continuing this way, it follows that for every state $u \in V_H$ there is a state $u' \in V_{H'}$ with the same follower set, and vice versa. Since both H and H' are reduced, we must have $H = H'$, up to labeled graph isomorphism.

For the proof of (b), we form a new graph H from G by merging all states of the latter with the same follower sets. It is not hard to see that H is irreducible, deterministic, and reduced. Then, apply part (a) to see that H is isomorphic to the Shannon cover. Clearly, G and H have the same follower sets. □

The following lemma generalizes part (b) of Theorem 2.12 to the situation where H presents a subsystem of $S(G)$.

Lemma 2.13 *Let G and H be two irreducible deterministic graphs. Then $S(H) \subseteq S(G)$ if and only if for every $v \in V_H$ there exists $u \in V_G$ such that $\mathcal{F}_H(v) \subseteq \mathcal{F}_G(u)$.*

Proof. The sufficiency is immediate. As for the necessity, by Lemma 2.8, $S(H)$ is irreducible. Thus, by Lemma 2.9, there is an irreducible component G' of the fiber product

$G * H$ such that $S(G') = S(G * H) = S(G) \cap S(H) = S(H)$. By Theorem 2.12 (part (b)), for every state $v \in V_H$ there exists a state $\langle u, u' \rangle \in V_{G'} \subseteq V_{G*H}$ such that

$$\mathcal{F}_H(v) = \mathcal{F}_{G'}\Big(\langle u, u' \rangle\Big) \subseteq \mathcal{F}_{G*H}\Big(\langle u, u' \rangle\Big) \subseteq \mathcal{F}_G(u),$$

as desired. □

Given some presentation of an irreducible constrained system S, the Shannon cover can be constructed as follows: First, use the determinizing construction of Section 2.2.1 to find a deterministic presentation G of S. By Lemma 2.9, S is presented by one of the irreducible components, say H, of G. Using Lemma 2.13, we can identify H among the irreducible components of G: for every other irreducible component H' of G, we must have $S(H') \subseteq S(H)$. Next, use the Moore algorithm of Section 2.6.2 (in particular, Proposition 2.10) to merge follower-set equivalent states in H to obtain an irreducible reduced deterministic graph. The latter is, by Theorem 2.12(a), the Shannon cover of S.

As an example, the labeled graph in Figure 2.3 is a deterministic presentation of the $(0, 1)$-RLL constrained system, but it is not the Shannon cover because states 0 and 2 have the same follower set. Indeed, the labeled graph in Figure 2.2 is the Shannon cover of the $(0, 1)$-RLL constrained system because it is deterministic, irreducible, and 0 is the label of an outgoing edge from state 0, but not from state 1. Note that if we merge states 0 and 2 in the labeled graph of Figure 2.3, we get the Shannon cover in Figure 2.2. The reader can verify that the Shannon cover of an RLL constrained system is the labeled graph depicted in Figure 1.3 in Chapter 1 and that Figure 1.14 displays the Shannon cover of a charge constrained system.

We end this section by pointing out the intrinsic nature of the follower sets of the states in the Shannon cover of an irreducible constrained system.

For a constrained system S and a word $\mathbf{w} \in S$, the *tail set* $\mathcal{F}_S(\mathbf{w})$ is the set of all words $\mathbf{z}$ such that $\mathbf{wz} \in S$. A word $\mathbf{w} \in S$ is a *magic word* if, whenever $\mathbf{zw}$ and $\mathbf{wz'}$ are in S, so is $\mathbf{zwz'}$.

Proposition 2.14 *Let S be an irreducible constrained system. The homing words of the Shannon cover of S coincide with the magic words of S, and the follower sets of the states of the Shannon cover coincide with the tail sets of the magic words of S.*

The proof of this proposition is left to the reader (Problem 2.22).

2.6.5 Shannon cover of finite-type constrained systems

The Shannon cover can be used to detect the finite-type and almost-finite-type properties.

Proposition 2.15 *An irreducible constrained system is finite-type (respectively, almost-finite-type) if and only if its Shannon cover has finite memory (respectively, finite co-anticipation).*

Proof. We first prove this for finite-type systems. Let S be an irreducible finite-type constrained system. By Proposition 2.5 and Lemma 2.4, there is a presentation G of S with finite memory. Also, by Lemma 2.9, there is an irreducible component G' of G such that $S = S(G')$. Since G' is an irreducible graph with finite memory, then it must be deterministic. By merging states in G', we obtain the Shannon cover of S. Therefore, the Shannon cover of an irreducible finite-type constrained system S must have finite memory.

Now, assume that S is an irreducible almost-finite-type constrained system. There is a presentation H of S which has finite co-anticipation and finite anticipation. It is not hard to see that the determinizing construction, introduced in Section 2.2.1, preserves finite co-anticipation. Thus, by Lemma 2.1, we may assume that H is actually deterministic. By Lemma 2.9, we may assume that H is irreducible. Then, by Theorem 2.12, the reduced labeled graph obtained from H is the Shannon cover G of S. In particular, this gives a graph homomorphism from H to G; that is, there is a mapping f^* from states of H to states of G and a mapping f from edges of H to edges of G_S such that f is label-preserving and the initial (respectively, terminal) state of $f(e)$ is $f^*(\sigma_H(e))$ (respectively, $f^*(\tau_H(e))$).

Suppose, for the moment, that H is also co-deterministic. Create a new labeled graph $\bar{H}$ from H by replacing the edge label of an edge e by $f(e)$. Then $\bar{H}$ presents the constrained system $\bar{S}$ defined by labeling the edges of G distinctly. Moreover, $\bar{H}$ is co-deterministic since H is. When we merge H to form G, we are also merging $\bar{H}$ to form the Shannon cover $\bar{G}$ of $\bar{S}$, and by reversing the arrows, we see that any two states in $\bar{H}$ that are merged via f^* have the exact same set of incoming f-labels. So, whenever $f^*(v') = v$, the set of f-labels of the incoming edges to v' is precisely the set of incoming edges to v.

We claim that G is co-deterministic. If not, then there are two edges e_1 and e_2 in G with the same terminal state v and label. Let v' be any state of $\bar{H}$ such that $f^*(v') = v$. Then there are edges e_1' and e_2' in $\bar{H}$ with terminal state v' and labels e_1 and e_2. The edges e_1' and e_2', viewed as edges in H, then have the same labels, contradicting the co-determinism of H. Thus, G is co-deterministic and in particular has finite co-anticipation.

Now, the general case can be reduced to the special case where H is co-deterministic by a backwards determinizing procedure. This procedure shows that, in fact, the co-anticipation of G is at most the co-anticipation of H. We leave the details of this to the reader. □

From the previous result, we see that the constrained system presented by the labeled graph in Figure 2.19 is not almost-finite-type. Specifically, one can easily verify that the labeled graph in Figure 2.19 is the Shannon cover of the constrained system it presents. However, it does not have finite co-anticipation, as can be confirmed by looking at the paths that generate words of the form $\cdots aaaab$.

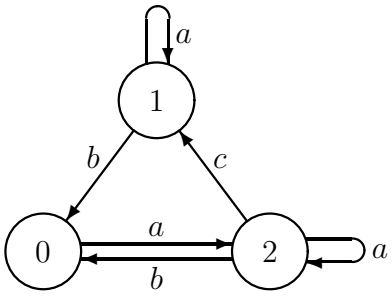


Figure 2.19: Constrained system which is not almost-finite-type.

2.7 Testing algorithms

In this section, we outline efficient algorithms for testing losslessness, finite anticipation, finite memory, and definiteness of a given labeled graph.

2.7.1 Testing for losslessness

The algorithm for testing losslessness of a given labeled graph is due to Even [Even65] and is based on the following proposition (see also [Huff59] and [Koh78, Ch. 14]).

Proposition 2.16 *A labeled graph G is lossless if and only if for every $u, u' \in V_G$, there is no path in the fiber product $G * G$ from state $\langle u, u \rangle$ to state $\langle u', u' \rangle$ that passes through a state of the form $\langle v, v' \rangle$, $v \neq v'$.*

(Recall that we assume that labeled graphs do not contain parallel edges that are labeled the same; such graphs would necessarily be lossy and Proposition 2.16 would not apply to them.)

Proposition 2.16 implies the following algorithm for testing the losslessness of a given labeled graph G. We start by constructing the fiber product $G * G$. Let U denote the states in $G * G$ of the form $\langle u, u \rangle$ for some $u \in V_G$, and let W be the set of all states $\langle v, v' \rangle$ in $G * G$, $v \neq v'$, with an incoming edge from a state in U. To verify that no path which starts in W terminates in U, we proceed as follows: For $\ell = 0, 1, \ldots, |V_G|^2 - 1$, we mark iteratively the states in $G * G$ that can be reached from W by a path of length $\leq \ell$ (this is known as the breadth-first-search (BFS) procedure [Even79]). Then check whether any of the states in U has been marked.

2.7.2 Testing for finite anticipation

A similar algorithm, also due to Even [Even65], allows us to find the anticipation of a given labeled graph. The algorithm is based on the following.

Proposition 2.17 *Let G be a labeled graph and denote by W the set of all states $\langle v, v' \rangle$ in $G * G$, $v \neq v'$, with an incoming edge from a state of the form $\langle u, u \rangle$. Then, G has finite anticipation if and only if no path in $G * G$ that starts at any state in W contains a cycle. If W is empty, then G is deterministic and $\mathcal{A}(G) = 0$. Otherwise, the length of the longest path from W equals $\mathcal{A}(G) - 1$.*

The anticipation of G can therefore be efficiently computed by constructing a sequence of graphs $H_0, H_1, H_2, \ldots, H_t$, where $H_0 = G * G$ and H_i is obtained from H_{i-1} by deleting all states in H_{i-1} with no outgoing edges. The procedure terminates when $H_{i-1} = H_i$ or when H_i contains no states that belong to the set W. In the latter case, the number of iterations, t, equals the anticipation of G. Otherwise, if H_t does contain states of W, then G has infinite anticipation.

Noting that $\langle v, v' \rangle$ and $\langle v', v \rangle$ are follower-set equivalent states in $G * G$, we can construct a reduced labeled graph G' out of $G * G$, where each such pair of states merges into one state of G'. The labeled graph G' will contain at most $|V_G|$ states of the form $\langle u, u \rangle$, $u \in V$, and at most $|V_G|(|V_G|-1)/2$ states of the form $\langle v, v' \rangle$, $v \neq v'$. Now, Proposition 2.17 applies also to the paths in G'. The longest path in G' that neither visits the same state twice, nor visits states of the form $\langle u, u \rangle$, is of length $|V_G|(|V_G|-1)/2 - 1$. Hence, we have the following.

Corollary 2.18 *Let G be a labeled graph. If G has finite anticipation, then $\mathcal{A}(G) \leq |V_G|(|V_G|-1)/2$.*

There are constructions of labeled graphs G that attain the bound of Corollary 2.18 for every value of $|V_G|$ [Koh78, Appendix 14.1].

2.7.3 Testing for finite memory

The following is basically contained in [PRS63] and [Koh78, Ch. 14] (see Problem 2.24).

Proposition 2.19 *Let G be a labeled graph. Then, G has finite memory if and only if there exists an integer N such that all paths in $G * G$ of length N terminate in states of the form $\langle u, u \rangle$, $u \in V_G$. The smallest such N, if any, equals $\mathcal{M}(G)$.*

In particular, in view of Proposition 2.15, given an irreducible constrained system S, we can apply Proposition 2.19 to the Shannon cover of S to check whether S has finite memory.

The following corollary is the counterpart of Corollary 2.18 for the memory of a labeled graph.

Corollary 2.20 *Let G be a labeled graph. If G has finite memory, then $\mathcal{M}(G) \leq |V_G|(|V_G|-1)/2$.*

The bound of Corollary 2.20 is tight [Koh78, Ch. 14, Problems].

2.7.4 Testing for definiteness

Next we outline an efficient test for determining whether a given labeled graph G is (m, a)-definite. (Note that we could use a test for definiteness also for testing finite memory. However, to this end, Proposition 2.19 provides a faster algorithm.)

Let G be a labeled graph and let A_{G*G} be the adjacency matrix of $G*G$. Denote by B_{G*G} the $|V_G|^2 \times |V_G|^2$ matrix whose rows and columns are indexed by the states of $G*G$ and for every $u, u', v, v' \in V_G$, the entry $(B_{G*G})_{\langle u,u'\rangle,\langle v,v'\rangle}$ equals the number of pairs of *distinct* edges $u \to v$ and $u' \to v'$ in G that have the same label. In other words,

$$(B_{G*G})_{\langle u,u'\rangle,\langle v,v'\rangle} = \begin{cases} (A_{G*G})_{\langle u,u'\rangle,\langle v,v'\rangle} & \text{if } u \neq u' \text{ or } v \neq v' \\ (A_{G*G})_{\langle u,u'\rangle,\langle v,v'\rangle} - (A_G)_{u,v} & \text{if } u = u' \text{ and } v = v' \end{cases} .$$

Proposition 2.21 *A labeled graph G is (m, a)-definite if and only if*

$$A_{G*G}^{\mathsf{m}}\, B_{G*G}\, A_{G*G}^{\mathsf{a}} = 0 .$$

The proof is left as an exercise (Problem 2.25).

Problems

Problem 2.1 Let G be a labeled graph.

1. Show that G has a unique maximal essential subgraph H.
2. Let S be the constrained system defined by G, and let S' be the subset of S consisting of all words $\mathbf{w}$ such that for any integer ℓ there are words $\mathbf{z}$ and $\mathbf{z}'$ of length ℓ such that $\mathbf{zwz}'$ belongs to S. Show that S' is the constrained system presented by H.

Problem 2.2 Let G be a graph and let G' and G'' be the Moore form and Moore co-form of G, respectively. Prove the following claims:

1. Edges with the same terminal state in G' have the same labels, and edges with the same initial state in G' have the same labels.
2. If the out-degrees of the states in G are all equal to n, then so are the out-degrees of the states in G' and G''.
3. $\mathcal{A}(G') = \mathcal{A}(G)$.
4. $\mathcal{A}(G'') \leq \mathcal{A}(G) + 1$. When is this inequality strict?

Problem 2.3 Let G be the graph in Figure 2.20 with labels over the alphabet $\Sigma = \{a, b, c\}$. Show

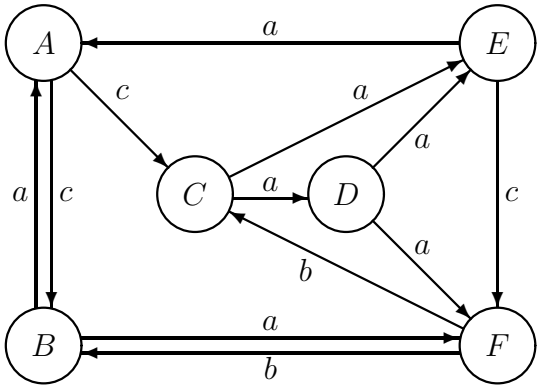


Figure 2.20: Graph G for Problem 2.3.

that the anticipation of G is 3.

Problem 2.4 Let G be the graph in Figure 2.21 with labels over the alphabet $\Sigma = \{a, b, c\}$. Show that the anticipation of G is 4.

Problem 2.5 Find the memory of the graph G in Figure 2.22.

Problem 2.6 Let S be a constrained system with finite memory $\mathcal{M} \geq 1$ over an alphabet Σ.

1. Show that for every constrained system S' over Σ,

$$S' \subseteq S \quad \text{if and only if} \quad (S' \cap \Sigma^i) \subseteq (S \cap \Sigma^i) \quad \text{for every} \quad i = 1, 2, \ldots, \mathcal{M}+1 \, .$$

2. Show that there exists a constrained system S' that is *not* contained in S, yet satisfies the containment

$$(S' \cap \Sigma^i) \subseteq (S \cap \Sigma^i) \quad \text{for every} \quad i = 1, 2, \ldots, \mathcal{M} \, .$$

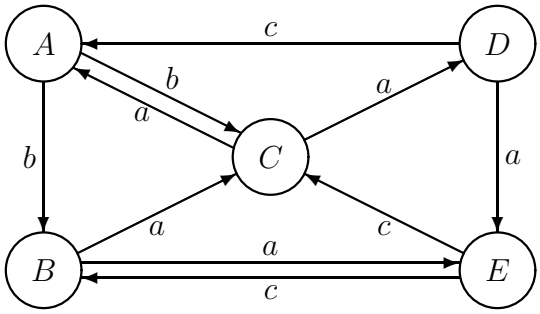


Figure 2.21: Graph G for Problem 2.4.

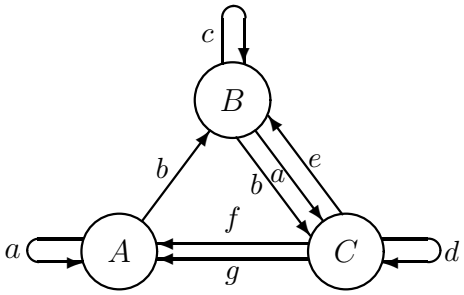


Figure 2.22: Graph G for Problem 2.5.

Problem 2.7 Prove Proposition 2.6.

Problem 2.8 Show that if S_1 and S_2 are constrained systems that are almost-finite-type, then so is $S_1 \cap S_2$.

Problem 2.9 Let G_1 and G_2 be graphs and $G_1 * G_2$ be the fiber product of G_1 and G_2.

1. Show that $S(G_1 * G_2) = S(G_1) \cap S(G_2)$.
2. Prove or disprove the following:
   (a) If G_1 and G_2 are both lossless, then so is $G_1 * G_2$.
   (b) If $G_1 * G_2$ is lossless, then so are both G_1 and G_2.

Problem 2.10 Let G_1 and G_2 be graphs with finite memory. Show that

$$\mathcal{M}(G_1 * G_2) \leq \max\{\mathcal{M}(G_1), \mathcal{M}(G_2)\} .$$

Problem 2.11 Let G be the graph presentation of a 4-charge constrained system in Figure 2.23.

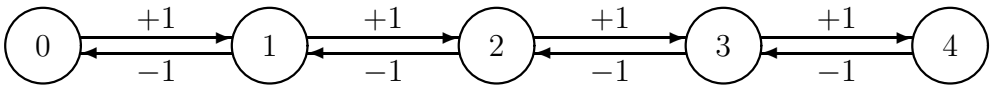


Figure 2.23: Graph G for Problem 2.11.

1. Find a shortest homing word for every state in G.
2. What can be said about the memory of G?

Problem 2.12 Let G be the graph in Figure 2.24.

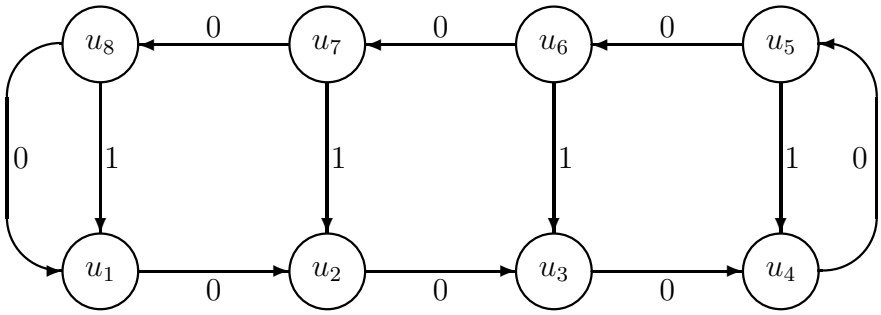


Figure 2.24: Graph G for Problem 2.12.

1. Find a shortest homing word in G.
2. What can be said about the memory of G?

Problem 2.13 Let S be the constrained system presented by the graph G in Figure 2.25.

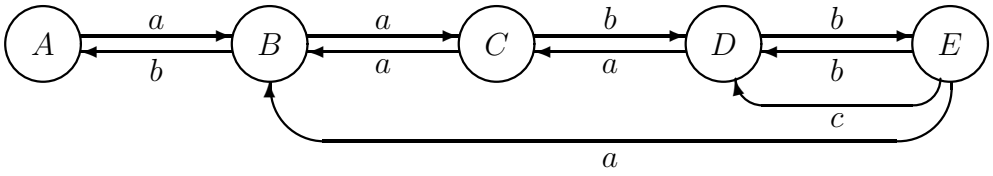


Figure 2.25: Graph G for Problem 2.13.

1. Find a shortest homing word for every state in G.
2. Construct the graph G^2.

3. Find the memory of each irreducible component of G^2.

4. What can be said about the memory of G^2?

Problem 2.14 Let G_1 and G_2 be graphs. Show that $\mathbf{w}$ is a homing word of $G_1 * G_2$ if and only if $\mathbf{w}$ is a homing word of both G_1 and G_2.

Problem 2.15 Let G be an irreducible graph and let G' and G'' be the Moore form and Moore co-form of G, respectively. Show that both G' and G'' are irreducible.

Problem 2.16 Let G be the fiber product of the graphs in Figure 1.15 and Figure 1.16; note that G presents the 6–$(1,3)$-CRLL constrained system.

1. Draw the graph G.

2. Show that one of the irreducible components of G^2 can be reduced to the graph H in Figure 2.26.

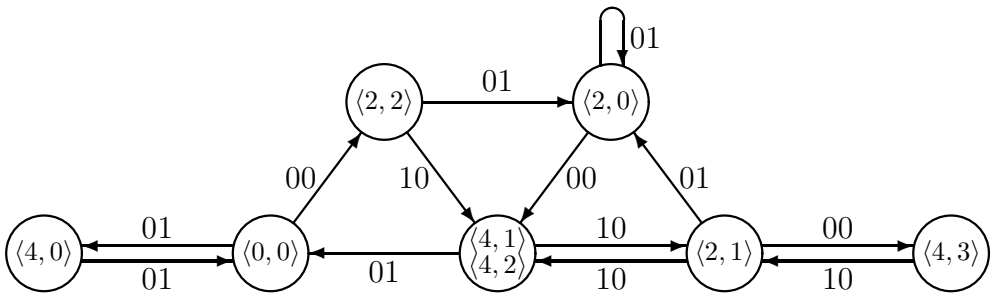


Figure 2.26: Graph H for Problem 2.16.

Problem 2.17 Let S_0 be an irreducible constrained system and let G be a graph such that $S_0 \subseteq S(G)$. Show that there is an irreducible component H of G such that $S_0 \subseteq S(H)$.

Problem 2.18 Let S_1 and S_2 be irreducible constrained systems such that $S_1 \subseteq S_2$.

1. Show that there is an irreducible deterministic presentation H_1 of S_1 that is a subgraph of an irreducible deterministic presentation H_2 of S_2.

   Hint: Let G_1 and G_2 be the Shannon covers of S_1 and S_2, respectively, and, as in the proof of Lemma 2.13, consider an irreducible component H_1 of $G_1 * G_2$ that presents S_1. Show how H_1 can be extended to an irreducible deterministic graph H_2 such that $S(H_2) = S_2$.

2. Show by example that not always can H_1 be taken as the Shannon cover of S_1; that is, for the provided example, the Shannon cover of S_1 is not a subgraph of any *irreducible* deterministic presentation of S_2.

3. Show by example that not always can H_2 be taken as the Shannon cover of S_2.

Problem 2.19 Let G and H be deterministic graphs where H is irreducible. Suggest an efficient algorithm for determining whether $S(H) \subseteq S(G)$.

Problem 2.20 Let G be a deterministic graph that presents an irreducible constrained system S (but G is not necessarily irreducible). It follows from Lemmas 2.8 and 2.9 that G contains an irreducible component G' such that $S(G) = S(G')$. Suggest an efficient algorithm for finding G'.

Hint: See Problem 2.19

Problem 2.21 Let S be an irreducible constrained system with finite memory. Show that the memory of S equals the memory of the Shannon cover of S.

Problem 2.22 Prove Proposition 2.14.

Problem 2.23 Let G be a labeled graph and let W be the subset of states of $G * G$ as defined in Proposition 2.17. Denote by $\mathbf{x} = (x_{\langle v,v' \rangle})_{\langle v,v' \rangle}$ the characteristic vector of W as a subset of the states of $G * G$; that is, the entries of $\mathbf{x}$ are indexed by the states of $G * G$, and

$$x_{\langle v,v' \rangle} = \begin{cases} 1 & \text{if } \langle v, v' \rangle \in W \\ 0 & \text{otherwise} \end{cases} .$$

1. Show that G has finite anticipation if and only if there is a nonnegative integer ℓ such that

$$\mathbf{x} A_{G*G}^{\ell} = \mathbf{0} .$$

2. Show that if G has finite anticipation, then its anticipation is the smallest nonnegative integer ℓ that satisfies the equality in 1.

Problem 2.24 Let $G = (V, E, L)$ be a deterministic graph and let H be the graph obtained from $G * G$ by deleting the set of states $\{\langle v, v \rangle : v \in V\}$, with their incident edges.

1. Show that G has finite memory if and only if H has no cycles.

2. Show that if G has finite memory, then the memory is bounded from above by $|V|(|V|-1)/2$.

Problem 2.25 Prove Proposition 2.21.

Problem 2.26 Let U be a set of positive integers (U may be either finite or infinite). The *U-gap system* is defined as the set of all sub-words of all binary words in which each runlength of 0's belongs to U.

1. Let U be the set of even integers. Show that the U-gap system is a constrained system but has no finite memory.
2. Let U be the set of prime integers. Show that the U-gap system is not a constrained system.
3. Formulate complete necessary and sufficient conditions on U for the U-gap system to be—
   (a) a constrained system;
   (b) a constrained system with finite memory.

Chapter 3

Capacity

In this chapter we introduce and study the notion of capacity, which is one of the most important parameters related to constrained systems. In the context of coding, the significance of capacity will be made apparent in Chapter 4, where we show that it sets an (attainable) upper bound on the rate of any finite-state encoder for a given constrained system.

The definition of capacity is given in Section 3.1. We then provide two other characterizations of capacity—an algebraic and a probabilistic one. The algebraic characterization leads to a method for computing capacity from any lossless graph presentation of the constrained system (see Theorem 3.4 below).

3.1 Combinatorial characterization of capacity

Let S be a constrained system over an alphabet Σ and denote by $N(\ell; S)$ the number of words of length ℓ in S. The *base*-2 *Shannon capacity*, or simply *capacity* of S, is defined by

$$\mathsf{cap}(S) = \limsup_{\ell \to \infty} \frac{1}{\ell} \log N(\ell; S) \ .$$

Hereafter, if the base of the logarithms is omitted then it is assumed to be 2.

The Shannon capacity $\mathsf{cap}(S)$ of S measures the growth rate of the number of words of length ℓ in S, in the sense that the $N(\ell; S)$ is well-approximated by $2^{\ell\, \mathsf{cap}(S)}$ for large enough ℓ.

If $|S| = \infty$ then $0 \leq \mathsf{cap}(S) \leq \log |\Sigma|$. Otherwise, if S is finite, then $\mathsf{cap}(S) = -\infty$. In the latter case, there are no cycles in any presentation G of S; so each irreducible component of G is a trivial graph with one state and no edges.

Example 3.1 Let S be the $(0, 1)$-RLL constrained system which is presented by the

graph G shown in Figure 3.1. For $u \in \{0,1\}$, denote by $x_u(\ell)$ the number of words of length

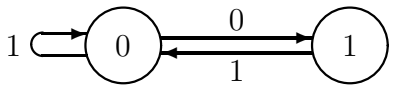


Figure 3.1: Shannon cover of the $(0,1)$-RLL constrained system.

ℓ that can be generated from state u in G. Then for $\ell \geq 1$,

$$\begin{aligned} x_0(\ell) &= x_0(\ell-1) + x_1(\ell-1) \ , \\ x_1(\ell) &= x_0(\ell-1) \ , \end{aligned}$$

and the initial conditions are obtained for $\ell = 0$ (the empty word) by

$$x_0(0) = x_1(0) = 1 \ .$$

So, for $\ell \geq 2$,

$$x_0(\ell) = x_0(\ell-1) + x_0(\ell-2)$$

with the initial conditions $x_0(0) = 1$, $x_0(1) = 2$. Hence, $x_0(\ell)$ are Fibonacci numbers and can be written as

$$x_0(\ell) = c_1\lambda^\ell + c_2(-\lambda)^{-\ell} \ ,$$

where $\lambda = (1+\sqrt{5})/2$ (the golden mean ratio) and $c_1 > 0$. Since $\mathcal{F}_G(1) \subseteq \mathcal{F}_G(0)$ we have $\mathcal{F}_G(0) = S$ and

$$N(\ell; S) = x_0(\ell) \ .$$

Therefore,

$$\mathsf{cap}(S) = \log \frac{1+\sqrt{5}}{2} \approx .6942 \ .$$

□

The 'lim sup' in the definition of capacity can be replaced by a proper limit, and this can be shown in two ways. The most direct method is as follows: first, one shows that $\log N(\ell; S)$ is a *subadditive function*—i.e., for all ℓ and m, $(\log N(\ell+m; S)) \leq (\log N(\ell; S)) + (\log N(m; S))$; then one shows that for any subadditive function $f(\ell)$, the limit $\lim_{\ell\to\infty}(f(\ell)/\ell)$ exists (see [LM95, Lemma 4.1.7]). An alternative method is provided by Theorem 3.4 below.

The following theorem shows that the capacity of a constrained system S is determined by the irreducible components of a graph presentation of S.

Theorem 3.1 *Let $S = S(G)$ be a constrained system and let $G_1, G_2, \ldots, G_k$ be the irreducible components of G, presenting the irreducible systems $S_i = S(G_i)$. Then,*

$$\mathsf{cap}(S) = \max_{1\leq i\leq k} \mathsf{cap}(S_i) \ .$$

Proof. Clearly, $\mathsf{cap}(S) \geq \mathsf{cap}(S_i)$ for all $i = 1, 2, \ldots, k$. We now prove the inequality in the other direction. Any word $\mathbf{w} \in S \cap \Sigma^\ell$ can be decomposed into sub-words in the form

$$\mathbf{w} = \mathbf{w}_1 z_1 \mathbf{w}_2 z_2 \ldots z_{r-1} \mathbf{w}_r \ ,$$

where each $\mathbf{w}_j$ (possibly the empty word) is generated wholly within one of the irreducible components of G and each z_j is a label of an edge that links two irreducible components. Due to the partial ordering on the irreducible components of G, once we leave such a component we will not visit it again in the course of generating $\mathbf{w}$. Hence, $r \leq k$ and

$$N(\ell; S) = |S \cap \Sigma^\ell| \leq 2^k \cdot |\Sigma|^{k-1} \cdot \sum_{(\ell_1, \ell_2, \ldots, \ell_k)} \prod_{i=1}^{k} N(\ell_i; S_i) \ , \tag{3.1}$$

where $(\ell_1, \ell_2, \ldots, \ell_k)$ ranges over all nonnegative integer k-tuples such that $\ell_1 + \ell_2 + \ldots + \ell_k \leq \ell$. In (3.1), the term 2^k stands for the number of combinations of the traversed irreducible components; the term $|\Sigma|^{k-1}$ bounds from above the number of possible linking symbols z_j; and ℓ_i stands for the length of the sub-word of $\mathbf{w} \in S \cap \Sigma^\ell$ that is generated in the irreducible component G_i.

Without loss of generality we assume that $\mathsf{cap}(S_i)$ is nonincreasing with i and that h is the largest index i, if any, for which $\mathsf{cap}(S_i) \geq 0$; namely, $G_{h+1}, G_{h+2}, \ldots, G_k$ are the irreducible components of G with one state and no edges. Note that when no such h exists then $\mathsf{cap}(S) = -\infty$ and the theorem holds trivially.

By the definition of capacity, it follows that for every $i \leq h$ and $m \in \mathbb{N}$ we have

$$N(m; S_i) \leq \exp\Big\{m(\mathsf{cap}(S_i) + \varepsilon(m))\Big\} \leq \exp\Big\{m(\mathsf{cap}(S_1) + \varepsilon(m))\Big\} \ ,$$

where exponents are taken to base 2 and $\lim_{m \to \infty} \varepsilon(m) = 0$. Plugging this with $m = \ell_i$ into (3.1) we obtain

$$\begin{aligned} N(\ell; S) &\leq (2|\Sigma|)^k \cdot \sum_{(\ell_1, \ell_2, \ldots, \ell_h)} \exp\Big\{\Big(\sum_{i=1}^{h} \ell_i\Big)\mathsf{cap}(S_1)\Big\} \cdot \exp\Big\{\sum_{i=1}^{h} \ell_i \varepsilon(\ell_i)\Big\} \\ &\leq (2|\Sigma|)^k \cdot (\ell+1)^h \cdot \exp\Big\{\ell\, \mathsf{cap}(S_1)\Big\} \cdot \left(\max_{(\ell_1, \ell_2, \ldots, \ell_h)} \exp\Big\{\sum_{i=1}^{h} \ell_i \varepsilon(\ell_i)\Big\} \right) \ , \end{aligned}$$

where $(\ell_1, \ell_2, \ldots, \ell_h)$ ranges over all nonnegative integer h-tuples such that $\ell_1 + \ell_2 + \ldots + \ell_h \leq \ell$. Defining

$$\delta(\ell) = \frac{1}{\ell} \max_{(\ell_1, \ell_2, \ldots, \ell_h)} \sum_{i=1}^{h} \ell_i \varepsilon(\ell_i) \ ,$$

we obtain

$$\frac{1}{\ell} \log N(\ell; S) \leq \mathsf{cap}(S_1) + \frac{k \log(2|\Sigma|(\ell+1))}{\ell} + \delta(\ell) \ .$$

Hence, in order to complete the proof, it suffices to show that $\lim_{\ell\to\infty} \delta(\ell) = 0$. We leave this as an exercise (Problem 3.2). □

The following useful fact is a straightforward consequence of the definition of capacity. The proof is left as an exercise (see Problem 3.1).

Proposition 3.2 *For any constrained system S and positive integer ℓ,*

$$\mathsf{cap}(S^\ell) = \ell \cdot \mathsf{cap}(S) \ .$$

3.2 Algebraic characterization of capacity

In this section, we present an algebraic method for computing the capacity of a given constrained system. This method is based on Perron-Frobenius theory of nonnegative matrices. Our full treatment of Perron-Frobenius theory is deferred to Section 3.3. Still, we will provide here a simplified version of the theorem so that we can demonstrate how it is applied to the computation of capacity. We start with the following definition.

A nonnegative real square matrix A is called *irreducible* if for every row index u and column index v there exists a nonnegative integer $\ell_{u,v}$ such that $(A^{\ell_{u,v}})_{u,v} > 0$.

For a square real matrix A, we denote by $\lambda(A)$ the *spectral radius* of A—i.e., the largest of the absolute values of the eigenvalues of A.

The following is a short version of Perron-Frobenius theorem for irreducible matrices.

Theorem 3.3 *Let A be an irreducible matrix. Then $\lambda(A)$ is an eigenvalue of A and there are right and left eigenvectors associated with $\lambda(A)$ that are strictly positive; that is, each of their components is strictly positive.*

The following theorem expresses the capacity of an irreducible constrained system S in terms of the adjacency matrix of a lossless presentation of S.

Theorem 3.4 *Let S be an irreducible constrained system and let G be an irreducible lossless (in particular, deterministic) presentation of S. Then,*

$$\mathsf{cap}(S) = \log \lambda(A_G) \ .$$

We break the proof of Theorem 3.4 into two lemmas.

Lemma 3.5 *Let A be an irreducible matrix. Then, for every row index u,*

$$\lim_{\ell\to\infty}\frac{1}{\ell}\log\Big(\sum_{v}(A^{\ell})_{u,v}\Big)=\log\lambda(A)\ .$$

Furthermore,

$$\lim_{\ell\to\infty}\frac{1}{\ell}\log\Big(\sum_{u,v}(A^{\ell})_{u,v}\Big)=\log\lambda(A)\ .$$

Proof. We make use of a positive right eigenvector $\mathbf{x}$ associated with the eigenvalue $\lambda=\lambda(A)$. Let $x_{\max}$ and $x_{\min}$ denote the maximal and minimal components of $\mathbf{x}$, respectively. Both $x_{\max}$ and $x_{\min}$ are strictly positive. For each row index u we have

$$x_{\min}\sum_{v}(A^{\ell})_{u,v}\le\sum_{v}(A^{\ell})_{u,v}x_v=\lambda^{\ell}x_u\ .$$

Thus,

$$\sum_{v}(A^{\ell})_{u,v}\le\frac{x_u}{x_{\min}}\cdot\lambda^{\ell}\ .$$

Replacing $x_{\min}$ by $x_{\max}$ and reversing the direction of the inequalities, we get

$$\sum_{v}(A^{\ell})_{u,v}\ge\frac{x_u}{x_{\max}}\cdot\lambda^{\ell}\ .$$

It thus follows that the ratio of $\sum_v(A^{\ell})_{u,v}$ to λ^{ℓ} is bounded above and below by positive constants and, so, these two quantities grow at the same rate. The same holds with respect to $\sum_{u,v}(A^{\ell})_{u,v}$. □

Lemma 3.6 *Let S be an irreducible constrained system and let G be an irreducible lossless presentation of S. Then,*

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

Proof. Recall that $\sum_{u,v}(A_G^{\ell})_{u,v}$ is the number of paths of length ℓ in G. Now, every word of length ℓ in S can be generated by at least one path in G. On the other hand, since G is lossless, every word in S can be generated by at most $|V_G|^2$ paths in G. Hence, the number, $N(\ell;S)$, of words of length ℓ in S is bounded from below and above by

$$\frac{1}{|V_G|^2}\cdot\sum_{u,v}(A_G^{\ell})_{u,v}\le N(\ell;S)\le\sum_{u,v}(A_G^{\ell})_{u,v}\ .$$

Therefore,

$$\mathsf{cap}(S)=\lim_{\ell\to\infty}\frac{1}{\ell}\log N(\ell;S)=\lim_{\ell\to\infty}\frac{1}{\ell}\log\Big(\sum_{v}(A^{\ell})_{u,v}\Big)=\log\lambda(A)\ .$$

Note that we have established here that 'lim sup' can indeed be replaced by a proper limit. □

Example 3.2 For the $(0,1)$-RLL constrained system presented by the deterministic graph in Figure 3.1, the adjacency matrix is

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

with largest eigenvalue $\lambda = (1+\sqrt{5})/2$ and capacity $\log \lambda \approx .6942$. □

Example 3.3 For $0 \leq d \leq k < \infty$, let $G(d,k)$ denote the Shannon cover in Figure 1.3 of the (d,k)-RLL constrained system. It can be shown that $\lambda(A_{G(d,k)})$ is the largest positive solution of the equation

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

(see Problem 3.19). This in turn, allows to compute the capacity of any (d,k)-RLL constrained system. Table 3.1 (taken from [Imm91]) contains the capacity values of several (d,k)-RLL constrained systems. □

$k\backslash d$	0	1	2	3	4	5	6
1	.6942						
2	.8791	.4057					
3	.9468	.5515	.2878				
4	.9752	.6174	.4057	.2232			
5	.9881	.6509	.4650	.3218	.1823		
6	.9942	.6690	.4979	.3746	.2669	.1542	
7	.9971	.6793	.5174	.4057	.3142	.2281	.1335
8	.9986	.6853	.5293	.4251	.3432	.2709	.1993
9	.9993	.6888	.5369	.4376	.3630	.2979	.2382
10	.9996	.6909	.5418	.4460	.3746	.3158	.2633
11	.9998	.6922	.5450	.4516	.3833	.3282	.2804
12	.9999	.6930	.5471	.4555	.3894	.3369	.2924
13	.9999	.6935	.5485	.4583	.3937	.3432	.3011
14	.9999	.6938	.5495	.4602	.3968	.3478	.3074
15	.9999	.6939	.5501	.4615	.3991	.3513	.3122
∞	1.0000	.6942	.5515	.4650	.4057	.3620	.3282

Table 3.1: Capacity values of several (d,k)-RLL constrained systems.

Example 3.4 Consider the 2-charge constrained system whose Shannon cover is given by the graph G in Figure 3.2. The adjacency matrix of G is given by

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

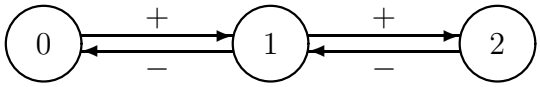


Figure 3.2: Shannon cover of the 2-charge constrained system.

with eigenvalues $\pm\sqrt{2}$ and 0. Hence, the capacity of the 2-charge constrained system is $\log\sqrt{2} = 1/2$.

More generally, if G_B is the Shannon cover in Figure 1.14 of the B-charge constrained system, then it can be shown that

$$\lambda(A_{G_B}) = 2\cos\left(\frac{\pi}{B+2}\right)$$

(see Problem 3.20). Table 3.2 lists the values of $\log\lambda(A_{G_B})$ for several values of B. □

B	1	2	3	4	5	6	7	8	9	10	11	12
	.0000	.5000	.6942	.7925	.8495	.8858	.9103	.9276	.9403	.9500	.9575	.9634

Table 3.2: Capacity values of several B-charge constrained systems.

It turns out that Theorem 3.4 and Lemma 3.6 hold for any constrained system S and lossless graph G, irreducible or reducible. We show this next.

Theorem 3.7 *Let S be a constrained system and let G be a lossless presentation of S. Then, there is an irreducible constrained system $S' \subseteq S$ such that*

$$\mathsf{cap}(S') = \mathsf{cap}(S) = \log\lambda(A_G)\ .$$

Proof. Let $G_1, G_2, \ldots, G_k$ be the irreducible components of G and denote by A_i the adjacency matrix of G_i. By reordering the states, we can assume that the adjacency matrix A of G has the block-triangular form of Figure 3.3. Since the set of eigenvalues of A is the union of the set of eigenvalues of the matrices A_i, we obtain

$$\lambda(A) = \max_{i=1}^{k}\lambda(A_i)\ .$$

The result now follows from Theorem 3.1. □

$$A = \begin{pmatrix} A_1 & B_{1,2} & B_{1,3} & \cdots & B_{1,k} \\ & A_2 & B_{2,3} & \cdots & B_{2,k} \\ & & A_3 & \ddots & \vdots \\ & & & \ddots & B_{k-1,k} \\ & & & & A_k \end{pmatrix}.$$

Figure 3.3: Block-triangular form.

3.3 Perron-Frobenius theory

In this section, we present a more extensive treatment of Perron-Frobenius theory. We have already exhibited one application of this theory—namely, providing a means for computing capacity. In fact, as we show in Chapters 5 and 7, this theory also serves as a major tool for constructing and analyzing constrained systems.

3.3.1 Irreducible matrices

Recall that a nonnegative real square matrix A is called irreducible if for every row index u and column index v there exists a nonnegative integer $\ell_{u,v}$ such that $(A^{\ell_{u,v}})_{u,v} > 0$. A nonnegative real square matrix that is not irreducible is called *reducible* .

The 1×1 matrix $A = \begin{pmatrix} 0 \end{pmatrix}$ will be referred to as the *trivial* irreducible matrix. The trivial irreducible matrix is the adjacency matrix of the trivial irreducible graph (which has one state and no edges).

Irreducibility of a nonnegative real square matrix A depends on the locations (row and column indexes) of the nonzero entries in A, and not on their specific values. For example, irreducibility would be preserved if we changed each nonzero entry in A to 1. Therefore, the following definition is useful.

Let A be a nonnegative real square matrix. The *support graph* of A is a graph G with a state for each row in A and an edge $u \to v$ if and only if $(A)_{u,v} > 0$. Note that A is irreducible if and only if its support graph G is irreducible, and G is irreducible if and only if its adjacency matrix A_G is irreducible.

In analogy with graphs, we can now define an *irreducible component* of a nonnegative real square matrix A as an irreducible submatrix of A whose support graph is an irreducible component of the support graph of A. The term *irreducible sink* extends to matrices in a straightforward manner. By applying the same permutation on both the rows and columns of A, we can obtain a matrix in upper block-triangular form with its irreducible components, $A_1, A_2, \ldots, A_k$, as the block diagonals, as shown in Figure 3.3.

We will use in the sequel the following notations. Let A and B be real matrices (in particular, vectors) of the same order. We write $A \geq B$ (respectively, $A > B$) if the weak (respectively, strict) inequality holds component by component. We say that A is *strictly positive* if $A > 0$.

3.3.2 Primitivity and periodicity

Let G be a nontrivial irreducible graph. We say that G is *primitive* if there exists a (strictly) positive integer ℓ such that for every ordered pair of states (u, v) of G there is a path of length ℓ from u to v. Equivalently, A_G^ℓ is strictly positive; note that this implies that A_G^m is strictly positive for every $m > \ell$, since the adjacency matrix of a nontrivial irreducible graph cannot have all-zero rows or columns. Observe that the trivial matrix is not a primitive matrix.

Let G be a nontrivial irreducible graph. The *period* of G is the greatest common divisor of the lengths of all cycles in G. We say that G is *aperiodic* if its period is 1.

It is not difficult to check that the graph in Figure 3.1 (which presents the $(0,1)$-RLL constrained system) is aperiodic and the graph in Figure 3.2 (which presents the 2-charge constrained system) has period 2.

Proposition 3.8 *A nontrivial irreducible graph is aperiodic if and only if it is primitive.*

Proof. Let G be a primitive graph. Then there exists a positive integer ℓ such that A_G^ℓ is strictly positive, and therefore so is $A_G^{\ell+1}$. In particular, there exist cycles in G of lengths ℓ and $\ell+1$. Hence, G is aperiodic.

Conversely, assume that $G = (V, E, L)$ is aperiodic and let $\Gamma_1, \Gamma_2, \ldots, \Gamma_k$ be cycles in G of lengths $t_1, t_2, \ldots, t_k$, respectively, such that $\gcd(t_1, t_2, \ldots, t_k) = 1$. By the extended Euclidean algorithm, there exist integers $b_1, b_2, \ldots, b_k$ such that $\sum_{i=1}^k b_i t_i = \gcd(t_1, t_2, \ldots, t_k) = 1$. Define the constants

$$M = (2|V| - 1) \max_{i=1}^{k} |b_i|$$

and

$$a_{i,j} = M - jb_i \ , \quad i = 1, 2, \ldots, k \ , \quad j = 0, 1, \ldots, 2|V| - 2 \ .$$

Note that each $a_{i,j}$ is a positive integer.

For $i = 1, 2, \ldots, k$, let u_i be the initial (and terminal) state of the cycle Γ_i and let π_i be a path from u_i to u_{i+1} in G (see Figure 3.4). For $j = 0, 1, \ldots, 2|V| - 2$, define the path γ_j by

$$\gamma_j = \Gamma_1^{a_{1,j}} \pi_1 \Gamma_2^{a_{2,j}} \pi_2 \ldots \pi_{k-1} \Gamma_k^{a_{k,j}} \ .$$

That is, γ_j starts at state u_1, then circles $a_{1,j}$ times along Γ_1, then follows the edges of π_1 to reach u_2, next circles $a_{2,j}$ times along Γ_2, and so on, until it terminates in u_k. Denote by r

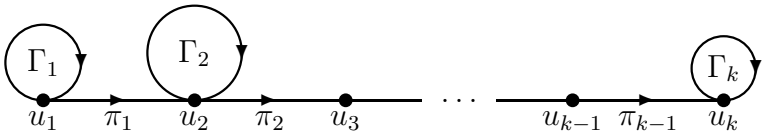


Figure 3.4: Paths for the proof of Proposition 3.8.

the length of the path $\pi_1\pi_2\ldots\pi_{k-1}$. For each $j = 0, 1, \ldots, 2|V| - 2$, the length of γ_j is given by

$$r + \sum_{i=1}^{k} a_{i,j}t_i = r + \sum_{i=1}^{k}(M - jb_i)t_i = \Big(r + M \cdot \sum_{i=1}^{k} t_i\Big) - j\sum_{i=1}^{k} b_i t_i = \ell - j \ ,$$

where $\ell = r + M \cdot \sum_{i=1}^{k} t_i$ is independent of j.

Now, let u and v be states in G and let π_0 and π_k be the shortest paths in G from u to u_1 and from u_k to v, respectively. Since π_0 and π_k each has length smaller than $|V|$, there exists one path γ_j such that the length of $\pi_0\gamma_j\pi_k$ is exactly ℓ. Hence, $(A_G^\ell)_{u,v} > 0$. □

Lemma 3.9 *Let u and v be two states in a nontrivial irreducible graph G with period p. Then all paths in G from state u to state v have congruent lengths modulo p.*

Proof. Let γ_1 and γ_2 be two paths from u to v in G of lengths ℓ_1 and ℓ_2, respectively. Also, let γ_3 be a path of length ℓ_3 from v to u. Since $\gamma_1\gamma_3$ and $\gamma_2\gamma_3$ are cycles, their lengths must be divisible by p. Therefore,

$$\ell_1 + \ell_3 \equiv \ell_2 + \ell_3 \equiv 0 \pmod{\mathsf{p}} \ .$$

Hence the result. □

Let G be a nontrivial irreducible graph with period p. Two states u and v in G are called *congruent*, denoted $u \equiv v$, if there is a path in G from u to v of length divisible by p. It can be readily verified that congruence is an equivalence relation that induces a partition on the states into equivalence classes.

Let C_0 be such an equivalence class, and for $r = 1, 2, \ldots, \mathsf{p}-1$, let C_r be the set of terminal states of edges in G whose initial states are in C_{r-1}, thus forming the sequence

$$C_0 \to C_1 \to \ldots \to C_{\mathsf{p}-1} \ .$$

The sets $C_0, C_1, \ldots, C_{\mathsf{p}-1}$ are necessarily all distinct, or else we would have a cycle in G whose length is less than p. The outgoing edges from $C_{\mathsf{p}-1}$ end path of length p that originate in C_0 and, so, their terminal states belong to C_0. It follows that the sets C_r form a partition of the set of states of G. In fact, each C_r is an equivalence of the congruence relation. Indeed, consider two states $u, v \in C_r$. There are paths in G,

$$u \to u_{r+1} \to u_{r+2} \to \ldots \to u_{\mathsf{p}-1} \to u_0 \ ,$$

and

$$v_0 \to v_1 \to v_2 \to \ldots \to v_{r-1} \to v \ ,$$

of lengths $\mathsf{p}-r$ and r, respectively, where $u_r, v_r \in C_r$; and since u_0 and v_0 are congruent, then so are u and v.

Example 3.5 The graph in Figure 3.2 has period 2, and the equivalence classes of the congruence relation are given by $C_0 = \{0, 2\}$ and $C_1 = \{1\}$. □

The definitions of period and primitivity extend to irreducible matrices through their support graphs as follows. Let A be a nontrivial irreducible matrix. The *period* $\mathsf{p} = \mathsf{p}(A)$ of A is the period of the support graph of A. A nontrivial irreducible matrix A is called *primitive* if the support graph of A is primitive.

Theorem 3.10 *Let A be a nontrivial irreducible matrix with period p and let*

$$C_0, C_1, \ldots, C_{\mathsf{p}-1}$$

be the equivalence classes of the congruence relation defined on the states of the support graph of A, where edges that start in C_r terminate in C_{r+1} (C_0 if $r = \mathsf{p}-1$).

(a) The nonzero entries of A all belong to p submatrices $B_0, B_1, \ldots, B_{\mathsf{p}-1}$ of A, where each B_r has order $|C_r| \times |C_{r+1}|$ ($|C_{\mathsf{p}-1}| \times |C_0|$ if $r = \mathsf{p}-1$).

(b) A^{p} decomposes into p irreducible components $A_0, A_1, \ldots, A_{\mathsf{p}-1}$, where

$$A_r = B_r B_{r+1} \cdots B_{\mathsf{p}-1} B_0 \cdots B_{r-1} \ .$$

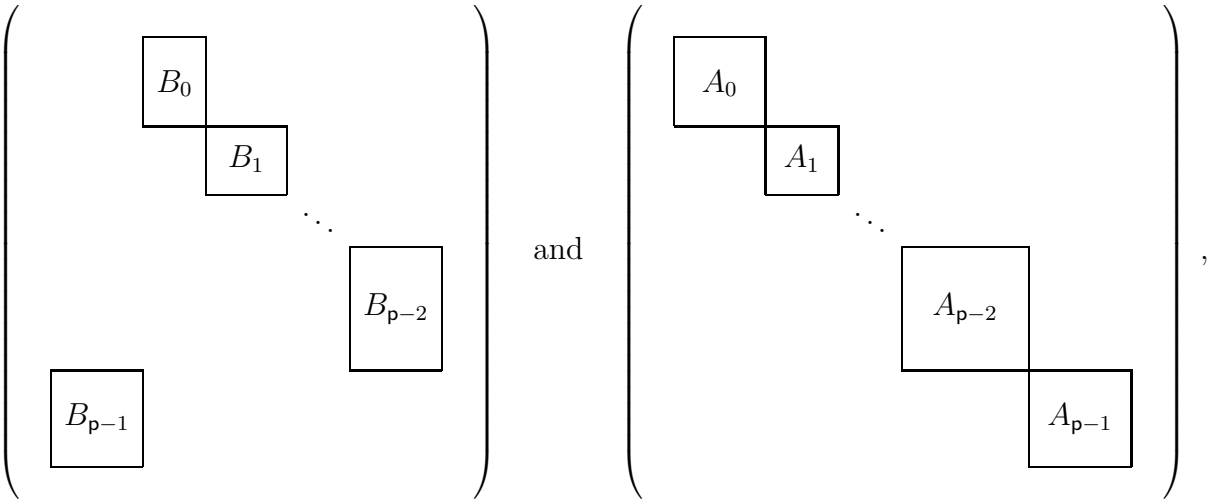
Furthermore, the entries of A^{p} that do not belong to any of the irreducible components are all zero (i.e., the irreducible components of the support graph of A^{p} are isolated).

(c) Each irreducible component A_r of A^{p} is primitive.

(d) The irreducible components of A^{p} all have the same set of nonzero eigenvalues, with the same multiplicity.

We present below a (partial) proof of the theorem. The statement of the theorem can be seen more clearly if we apply the same permutation on the rows and columns of A so that for $r = 1, 2, \ldots, \mathsf{p}-1$, the states of C_r follow those of C_{r-1}. In such a case, A and A^{p} take

the form



respectively. Note that the classes C_r need not necessarily be of the same size and the irreducible components A_r thus do not necessarily have the same order: different orders indicate different multiplicity of the zero eigenvalue.

Example 3.6 Continuing Example 3.5, consider again the graph G in Figure 3.2, which has period 2 and the equivalence classes of the congruence relation are $C_0 = \{0, 2\}$ and $C_1 = \{1\}$. As mentioned in Example 3.4, the adjacency matrix of G is given by

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

and after permuting the rows and columns of A_G so that the element(s) of C_1 follow those of C_0, we obtain the matrix

$$A = \left(\begin{array}{cc|c} 0 & 0 & 1 \\ 0 & 0 & 1 \\ \hline 1 & 1 & 0 \end{array}\right) = \left(\begin{array}{c|c} 0 & B_0 \\ \hline B_1 & 0 \end{array}\right) ,$$

where

$$B_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \text{and} \quad B_1 = \begin{pmatrix} 1 & 1 \end{pmatrix} .$$

The second power of A is given by

$$A^2 = \left(\begin{array}{c|c} B_0B_1 & 0 \\ \hline 0 & B_1B_0 \end{array}\right) = \left(\begin{array}{cc|c} 1 & 1 & 0 \\ 1 & 1 & 0 \\ \hline 0 & 0 & 2 \end{array}\right) .$$

The irreducible components G_0 and G_1 of G^2 are shown in Figure 2.16, and their adjacency matrices are

$$A_0 = B_0B_1 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \quad \text{and} \quad A_1 = B_1B_0 = \begin{pmatrix} 2 \end{pmatrix} ,$$

respectively. The eigenvalues of A_0 are 2 and 0, out of which only 2 is an eigenvalue of A_1.□

Proof of Theorem 3.10. (a) follows from the definition of C_r, and the expression for A_r in (b) follows from the rules of matrix multiplication. It is left as an exercise (see Problem 3.10) to show that each A_r is irreducible and primitive and that the nonzero entries in A^p all belong to the submatrices A_r.

As for (d), let μ be a nonzero eigenvalue of A_r; that is, there exists a nonzero vector $\mathbf{x}$ such that

$$A_r\mathbf{x} = \mu\mathbf{x} .$$

Multiplying both sides by B_{r-1} we obtain

$$B_{r-1}A_r\mathbf{x} = \mu B_{r-1}\mathbf{x} .$$

Now, $B_{r-1}A_r = A_{r-1}B_{r-1}$; so,

$$A_{r-1}(B_{r-1}\mathbf{x}) = \mu(B_{r-1}\mathbf{x}) .$$

Furthermore, the vector $B_{r-1}\mathbf{x}$ is nonzero, or else we would have $A_r\mathbf{x} = B_rB_{r+1}\ldots B_{r-1}\mathbf{x} = \mathbf{0}$, contrary to our assumption that $\mu \neq 0$. Hence, it follows that μ is an eigenvalue of A_{r-1}. By perturbation it can be shown that μ has the same algebraic multiplicity as an eigenvalue of A_r and A_{r-1}. □

3.3.3 Perron-Frobenius Theorem

Theorem 3.11 (Perron-Frobenius Theorem for irreducible matrices) [Gant60, Ch. XIII], [Minc88, Ch. 1], [Sen80, Ch. 1], [Var62, Ch. 2]) *Let A be a nontrivial irreducible matrix. Then there exists an eigenvalue λ of A such that the following holds.*

(a) λ is real and $\lambda > 0$.

(b) There are right and left eigenvectors associated with λ that are strictly positive; that is, each of their components is strictly positive.

(c) $\lambda \geq |\mu|$ for any other eigenvalue μ of A.

(d) The geometric multiplicity of λ is 1; that is, the right and left eigenvectors associated with λ are unique up to scaling.

Proof. *Parts (a) and (b).* Let A be of order $m \times m$ and define the set

$$\mathcal{B} = \{\mathbf{y} \in \mathbb{R}^m \,:\, \mathbf{y} \geq \mathbf{0}\} \ .$$

For $\mathbf{y} = (y_u)_u \in \mathcal{B}$, let $\rho(\mathbf{y})$ be defined by

$$\rho(\mathbf{y}) = \min_{u\,:\,y_u>0} \frac{(A\mathbf{y})_u}{y_u} \ .$$

Denoting by $u_{\max}$ the index u for which y_u is maximal, we have

$$0 \leq \rho(\mathbf{y}) \leq \frac{(A\mathbf{y})_{u_{\max}}}{y_{u_{\max}}} \leq \sum_v A_{u_{\max},v} \leq \max_u \sum_v A_{u,v} \ .$$

Therefore, the values $\rho(\mathbf{y})$ are uniformly bounded for every $\mathbf{y} \in \mathcal{B}$. Define

$$\lambda = \sup_{\mathbf{y}\in\mathcal{B}^*} \rho(\mathbf{y}) = \sup_{\mathbf{y}\in\mathcal{B}} \rho(\mathbf{y}) \ ,$$

where $\mathcal{B}^* = \{(y_u)_u \in \mathcal{B} \,:\, \sum_u y_u = 1\}$. Since the function $\mathbf{y} \mapsto \rho(\mathbf{y})$ is continuous over the compact set $\mathcal{B}^*$, there is some $\mathbf{x} \in \mathcal{B}^*$ for which $\rho(\mathbf{x}) = \lambda$. Observing that $A\mathbf{y} \geq \rho(\mathbf{y}) \cdot \mathbf{y}$ for every $\mathbf{y} \in \mathcal{B}$, it follows that

$$A\mathbf{x} \geq \lambda\mathbf{x} \ . \tag{3.2}$$

Next we show that the latter inequality holds with equality.

Suppose to the contrary that $A\mathbf{x} - \lambda\mathbf{x}$ is nonzero (and nonnegative). Define $B = (A + I)^{m-1}$, where I is the identity matrix; the matrix B is strictly positive (see Problem 3.14) and, therefore,

$$B(A\mathbf{x} - \lambda\mathbf{x}) > \mathbf{0} \ .$$

Letting $\mathbf{z} = (z_u)_u$ denote the vector $B\mathbf{x}$ and noting that B commutes with A, we have

$$A\mathbf{z} > \lambda\mathbf{z} \ ,$$

and, so,

$$\rho(\mathbf{z}) = \min_u \frac{(A\mathbf{z})_u}{z_u} > \lambda = \sup_{\mathbf{y}\in\mathcal{B}} \rho(\mathbf{y}) \ ,$$

thereby reaching a contradiction. We thus conclude that $A\mathbf{x} = \lambda\mathbf{x}$, i.e., λ is an eigenvalue of A with an associated nonnegative right eigenvector $\mathbf{x} = (x_u)_u$.

Next we show that both λ and $\mathbf{x}$ are strictly positive. Let the index v be such that $x_v > 0$, and for any index $u \neq v$, let $\ell_{u,v}$ be a positive integer for which $(A^{\ell_{u,v}})_{u,v} > 0$. Then, from $A^{\ell_{u,v}}\mathbf{x} = \lambda^{\ell_{u,v}}\mathbf{x}$ we obtain

$$\lambda^{\ell_{u,v}} x_u = (\lambda^{\ell_{u,v}}\mathbf{x})_u = (A^{\ell_{u,v}}\mathbf{x})_u \geq (A^{\ell_{u,v}})_{u,v} x_v > 0 \ .$$

Hence, $\lambda > 0$ and $\mathbf{x} > \mathbf{0}$. This completes the proof of part (a) and the first half of part (b): we still need to show that there is a strictly positive *left* eigenvector associated with λ. However, the existence of such a vector will follow from having, by (c), the same value of λ for the transpose of A.

Part (c). Let μ be a complex eigenvalue of A with an associated complex right eigenvector $\mathbf{y} = (y_u)_u$ and define the vector $\boldsymbol{\xi} = (\xi_u)_u$ by $\xi_u = |y_u|$. Taking the absolute value of both sides of

$$\sum_v (A)_{u,v} y_v = \mu\, y_u \ ,$$

we obtain, by the triangle inequality,

$$(A\boldsymbol{\xi})_u = \sum_v (A)_{u,v} |y_v| \geq \Big|\sum_v (A)_{u,v} y_v\Big| = |\mu| \xi_u \ ,$$

i.e.,

$$A\boldsymbol{\xi} \geq |\mu| \boldsymbol{\xi} \ .$$

Therefore,

$$|\mu| \leq \rho(\boldsymbol{\xi}) \leq \lambda \ . \tag{3.3}$$

Part (d). Let $\mathbf{x} = (x_u)_u$ be a strictly positive right eigenvector associated with the eigenvalue λ. Since λ is real, the linear space of the eigenvectors associated with λ is spanned by *real* eigenvectors. Let $\mathbf{y} = (y_u)_u$ be a real right eigenvector associated with λ and suppose to the contrary that $\mathbf{y}$ is linearly independent of $\mathbf{x}$. Then, for $\alpha = \max_u\{y_u/x_u\}$, the vector $\mathbf{z} = (z_u)_u = \alpha\mathbf{x} - \mathbf{y}$ is a nonnegative (nonzero) right eigenvector associated with λ and $z_u = 0$ for some index u. From $A\mathbf{z} = \lambda\mathbf{z}$ we obtain that $z_v = 0$ for every index v such that $(A)_{u,v} > 0$. Iterating inductively with each such v, we reach by the irreducibility of A the contradiction $\mathbf{z} = \mathbf{0}$. The respective proof for left eigenvectors is similar. □

Hereafter, we denote the transpose of a vector $\mathbf{y}$ by $\mathbf{y}^\top$.

Proposition 3.12 *Let A and B be nonnegative real square submatrices of the same order such that $A \geq B$ and A is irreducible. Then, $\lambda(A) \geq \lambda(B)$, with equality if and only if $A = B$.*

Proof. Let $\mathbf{z} = (z_u)_u$ be a right eigenvector of B associated with an eigenvalue μ such that $|\mu| = \lambda_B = \lambda(B)$ and let $\mathbf{x} = (x_u)_u$ be defined by $x_u = |z_u|$; from $B\mathbf{z} = \mu\mathbf{z}$ and the triangle inequality we have

$$A\mathbf{x} \geq B\mathbf{x} \geq \lambda_B \mathbf{x} \ , \tag{3.4}$$

where the first inequality follows from $A \geq B$. Let $\mathbf{y}^\top$ be a strictly positive left eigenvector of A associated with $\lambda_A = \lambda(A)$. Multiplying by $\mathbf{y}^\top$ yields

$$\lambda_A \mathbf{y}^\top \mathbf{x} = \mathbf{y}^\top A\mathbf{x} \geq \mathbf{y}^\top B\mathbf{x} \geq \lambda_B \mathbf{y}^\top \ , \tag{3.5}$$

and dividing by the positive constant $\mathbf{y}^\top\mathbf{x}$, we obtain $\lambda_A \geq \lambda_B$.

Now, if $\lambda_A = \lambda_B$, then the inequalities in (3.5) must hold with equality. In fact, this is also true for the inequalities in (3.4), since $\mathbf{y}^\top$ is strictly positive; that is,

$$A\mathbf{x} = B\mathbf{x} = \lambda_B\mathbf{x} = \lambda_A\mathbf{x} \; .$$

It follows that $\mathbf{x}$ is a nonnegative right eigenvector of A associated with λ_A; as such, it must be strictly positive. Combining this with $B\mathbf{x} = A\mathbf{x}$ and $A \geq B$ yields $A = B$. □

Proposition 3.13 *Let A be an irreducible matrix. Then the algebraic multiplicity of the eigenvalue $\lambda = \lambda(A)$ is 1; that is, λ is a simple root of the characteristic polynomial of A.*

Proof. The result is obvious for 1×1 matrices, so we exclude this case hereafter in the proof.

It is known that for every square matrix M,

$$M \cdot \mathrm{Adj}(M) = \det(M) \cdot I \; ,$$

where $\mathrm{Adj}(M)$ is the adjoint of M and I is the identity matrix. In particular,

$$(zI - A) \cdot \mathrm{Adj}(zI - A) = \chi_A(z) \cdot I \; ,$$

where $\chi_A(z) = \det(zI - A)$ is the characteristic polynomial of A. Differentiating with respect to z we obtain

$$\mathrm{Adj}(zI - A) + (zI - A) \cdot \frac{d}{dz}\Big(\mathrm{Adj}(zI - A)\Big) = \chi'_A(z) \cdot I \; .$$

We now substitute $z = \lambda$ and multiply each term by a strictly positive left eigenvector $\mathbf{y}^\top$ associated with λ; since $\mathbf{y}^\top(\lambda I - A) = \mathbf{0}^\top$, we end up with

$$\mathbf{y}^\top \mathrm{Adj}(\lambda I - A) = \chi'_A(\lambda)\, \mathbf{y}^\top \; .$$

Now, λ is a simple root of $\chi_A(z)$ if and only if $\chi'_A(\lambda) \neq 0$. Hence, to complete the proof, it suffices to show that the matrix $\mathrm{Adj}(\lambda I - A)$ is not all-zero. We do this next.

Let the matrix B be obtained from A by replacing the first row with the all-zero row. Denoting by $\chi_B(z)$ the characteristic polynomial of B, it is easy to see that the upper-left entry in $\mathrm{Adj}(\lambda I - A)$ is given by

$$\lambda^{-1}\chi_B(\lambda) \; .$$

However, from Proposition 3.12 it follows that λ is not an eigenvalue of B and, thus, cannot be a root of $\chi_B(z)$. □

Proposition 3.14 *Let A be an irreducible matrix. Then,*

$$\min_u \sum_v (A)_{u,v} \leq \lambda(A) \leq \max_u \sum_v (A)_{u,v} \ ,$$

where equality in one side implies equality in the other.

Proof. Let $\mathbf{y}^\top = (y_v)_v$ be a strictly positive left eigenvector associated with $\lambda = \lambda(A)$. Then $\sum_u y_u (A)_{u,v} = \lambda y_v$ for every index v. Summing over v, we obtain,

$$\sum_u y_u \sum_v (A)_{u,v} = \lambda \sum_v y_v \ ,$$

or

$$\lambda = \frac{\sum_u y_u \sum_v (A)_{u,v}}{\sum_v y_v} \ .$$

That is, λ is a weighted average (over v) of the values $\sum_v (A)_{u,v}$. □

Theorem 3.15 (Perron-Frobenius Theorem for nonnegative matrices.) *Let A be a nonnegative real square matrix. Then, the following holds.*

(a) The set of eigenvalues of A is the union (with multiplicity) of the sets of eigenvalues of the irreducible components of A.

(b) $\lambda(A)$ is an eigenvalue of A and there are nonnegative right and left eigenvectors associated with $\lambda(A)$.

Proof. Part (a) follows from the block-triangular form of Figure 3.3 (see the proof of Theorem 3.7). Part (b) is left as an exercise (see Problem 3.21). □

Since $\lambda(A)$ is actually an eigenvalue of A, we will refer to $\lambda(A)$ as the *largest eigenvalue of A* or the *Perron eigenvalue of A*.

When all the irreducible components of a nonnegative real $m \times m$ matrix A are trivial, then all the eigenvalues of A are zero. In this case, the characteristic polynomial of A is given by $\chi_A(z) = z^m$. Such a matrix is called *nilpotent*. Notice that the support graph of a nilpotent matrix A does not contain cycles and, so, there is no path in that graph of length m. Therefore, $A^m = 0$, consistently with Caley-Hamilton Theorem that states that the all-zero matrix is obtained when a square matrix is substituted in its characteristic polynomial.

3.3.4 Stronger properties in the primitive case

The following proposition says that in the primitive case, the inequality in Theorem 3.11(c) is strict.

Proposition 3.16 *Let A be a primitive matrix with $\lambda(A) = \lambda$. Then $|\mu| < \lambda$ for every eigenvalue $\mu \neq \lambda$ of A.*

Proof. We use the notations $\mathbf{y} = (y_u)_u$, $\boldsymbol{\xi} = (|y_u|)_u$, and $\rho(\cdot)$ as in the proof of Theorem 3.11(c). If $|\mu| = \lambda$ then it follows from (3.3) that $\rho(\boldsymbol{\xi}) = \lambda$, i.e.,

$$A\boldsymbol{\xi} \geq \lambda\boldsymbol{\xi} \ .$$

Re-iterating the arguments in the proof of parts (a) and (b) of Theorem 3.11 (see (3.2)), we conclude that $\boldsymbol{\xi}$ is a right eigenvector associated with the eigenvalue λ. Therefore, for every positive integer ℓ and every index u,

$$\Big|\sum_v (A^\ell)_{u,v} y_v\Big| = |\mu|^\ell |y_u| = \sum_v (A^\ell)_{u,v} |y_v| \ ,$$

i.e., the triangle inequality holds with equality. In such a case we have for every v,

$$(A^\ell)_{u,v} y_v = (A^\ell)_{u,v} |y_v| \cdot \beta \ ,$$

where $\beta = \beta(u,\ell)$ is such that $|\beta| = 1$. Taking ℓ so that $A^\ell > 0$, we obtain that $\mathbf{y}$ is a scalar multiple of $\boldsymbol{\xi}$ and $\mu = \lambda$. □

Theorem 3.17 *Let A be a primitive matrix and $\mathbf{x}$ and $\mathbf{y}^\top$ be strictly positive right and left eigenvectors of A associated with the eigenvalue $\lambda = \lambda(A)$, normalized so that $\mathbf{y}^\top \mathbf{x} = 1$. Then,*

$$\lim_{\ell \to \infty} (\lambda^{-1} A)^\ell = \mathbf{x}\,\mathbf{y}^\top \ .$$

Proof. The 1×1 case is immediate, so we exclude it from now on. Let μ be the largest absolute value of any eigenvalue of A other than λ; by Proposition 3.16 we have $\mu < \lambda$. Also, let h be the algebraic multiplicity of any eigenvalue of A whose absolute value equals μ. We show that

$$A^\ell = \lambda^\ell \mathbf{x}\mathbf{y}^\top + E^{(\ell)} \ ,$$

where $E^{(\ell)}$ is a matrix of the same order of A whose entries satisfy

$$|E^{(\ell)}_{u,v}| = O(\ell^{h-1}\mu^\ell) \tag{3.6}$$

for every u and v.

Write $A = P\Lambda P^{-1}$, where Λ is a matrix in *Jordan canonical form*; that is, Λ is a block-diagonal matrix where each block, Λ_i, is a square matrix that corresponds to an eigenvalue

λ_i and takes the *elementary Jordan form*: the entries on the main diagonal equal λ_i, and the other nonzero entries in the matrix are 1's below the main diagonal, as follows:

$$\Lambda_i = \begin{pmatrix} \lambda_i & & & & \\ 1 & \lambda_i & & & \\ & 1 & \lambda_i & & \\ & & \ddots & \ddots & \\ & & & 1 & \lambda_i \end{pmatrix} .$$

Each eigenvalue of A appears in the main diagonal of Λ a number of times which equals its algebraic multiplicity. We assume that the upper-left block Λ_1 corresponds to the largest eigenvalue λ; that is, $\Lambda_1 = \begin{pmatrix} \lambda \end{pmatrix}$ (by Proposition 3.13, this block has order 1×1). The first column of P and the first row of P^{-1} are, respectively, a right eigenvector $\mathbf{x}$ and a left eigenvector $\mathbf{y}^\top$ associated with λ, and $P \cdot P^{-1} = I$ implies that $\mathbf{x}$ and $\mathbf{y}^\top$ are normalized so that $\mathbf{y}^\top \mathbf{x} = 1$.

Now, $A^\ell = P\Lambda^\ell P^{-1}$, where Λ^ℓ is a block-diagonal matrix with blocks Λ_i^ℓ. It is easy to see that a each block Λ_i^ℓ is a lower-triangular matrix of the form

$$\Lambda_i^\ell = \begin{pmatrix} a_\ell & & & & \\ a_{\ell-1} & a_\ell & & & \\ a_{\ell-2} & a_{\ell-1} & a_\ell & & \\ \vdots & \ddots & \ddots & \ddots & \\ a_{\ell-s+1} & \dots & a_{\ell-2} & a_{\ell-1} & a_\ell \end{pmatrix} ,$$

where s in the order of Λ_i and $a_j = \binom{\ell}{j}\lambda_i^j$. Hence, the absolute value of each entry in Λ_i^ℓ is bounded from above by $\ell^{s-1}|\lambda_i|^\ell$. Noting that $s \le h$, it follows that the upper-left entry of Λ^ℓ equals λ^ℓ, whereas the absolute values of the other entries of Λ^ℓ are bounded from above by $\ell^{h-1}\mu^\ell$ for sufficiently large ℓ. Therefore,

$$A^\ell = P\Lambda^\ell P^{-1} = \lambda^\ell \mathbf{x}\mathbf{y}^\top + E^{(\ell)} ,$$

where $E^{(\ell)}$ satisfies (3.6). □

When A is not primitive, there are eigenvalues of A other than λ for which Theorem 3.11(c) holds with equality. Those eigenvalues are identified in the next theorem, which is quoted here without proof.

Theorem 3.18 *Let A be a nontrivial irreducible matrix with period p and let $\lambda = \lambda(A)$. Then there are exactly p eigenvalues μ of A for which $|\mu| = \lambda$: those eigenvalues have the form $\lambda\omega^i$, where ω is a root of order p of unity, and each of those eigenvalues has algebraic multiplicity 1.*

3.4 Markov chains

In Section 3.1, we defined the capacity of a constrained system S combinatorially as the growth rate of the number of words in S. Then, in Section 3.2, we showed how the capacity of S was related to the largest eigenvalue of a lossless presentation of S. In Section 3.5 below, we present yet another characterization of capacity, now through probabilistic means.

The following concept plays a major role in our discussion.

Let $G = (V, E)$ be a graph. A *Markov chain on* G is a probability distribution $\mathcal{P}$ on the edges of G; namely, the mapping

$$e \mapsto \mathcal{P}(e)$$

takes nonnegative values and $\sum_{e \in E} \mathcal{P}(e) = 1$.

For a state $u \in V$, let E_u denote the set of outgoing edges from u in G, i.e.,

$$E_u = \{e \in E \,:\, \sigma(e) = u\} \,,$$

where $\sigma(e) = \sigma_G(e)$ is the initial state of e in G. The *state probability vector* $\boldsymbol{\pi}^\top = (\pi_u)_{u \in V}$ of a Markov chain $\mathcal{P}$ on G is defined by

$$\pi_u = \sum_{e \in E_u} \mathcal{P}(e) \,.$$

The *conditional probability* of an edge $e \in E$ is defined by

$$q_e = \begin{cases} \mathcal{P}(e)/\pi_{\sigma(e)} & \text{if } \pi_{\sigma(e)} > 0 \\ 0 & \text{otherwise} \end{cases} \,.$$

A Markov chain $\mathcal{P}$ on G induces the following probability distribution on paths of G: given a path $\gamma = e_1 e_2 \ldots e_\ell$ in G, its probability is given by

$$\mathcal{P}(\gamma) = \pi_{\sigma(e_1)} q_{e_1} q_{e_2} \cdots q_{e_\ell} \,. \tag{3.7}$$

The *transition matrix* associated with $\mathcal{P}$ is a nonnegative real $|V| \times |V|$ matrix Q where for every $u, v \in V$,

$$(Q)_{u,v} = \sum_{e \in E_u \,:\, \tau(e) = v} q_e \,;$$

that is, $(Q)_{u,v}$ is the sum of the conditional probabilities of all edges from u to v in G. Note that Q is *stochastic*: the sum of entries in each row is 1.

A Markov chain $\mathcal{P}$ on G is called *stationary* if for every $u \in V$,

$$\sum_{e \in E \,:\, \tau(e) = u} \mathcal{P}(e) = \pi_u \,;$$

that is, the sum of the probabilities of the incoming edges to state u equals the respective sum of the outgoing edges from u. Equivalently,

$$\boldsymbol{\pi}^\top Q = \boldsymbol{\pi}^\top \ .$$

A stationary Markov chain $\mathcal{P}$ on G is called *irreducible* (or *ergodic*) if the associated transition matrix Q is irreducible. Similarly, $\mathcal{P}$ is called *primitive* (or *mixing*) if Q is a primitive matrix. Clearly, Q is irreducible (respectively, primitive) only if G is. Hereafter, when we say an irreducible (respectively, primitive) Markov chain, we mean an irreducible (respectively, primitive) *stationary* Markov chain.

Proposition 3.19 *Let Q be an irreducible stochastic $|V| \times |V|$ matrix. Then there is a unique positive vector $\boldsymbol{\pi}^\top = (\pi_u)_{u \in V}$ such that $\sum_u \pi_u = 1$ and*

$$\boldsymbol{\pi}^\top Q = \boldsymbol{\pi}^\top \ .$$

Proof. The matrix Q is irreducible and the sum of elements in each row is 1. By Proposition 3.14 we thus have $\lambda(Q) = 1$. The existence and uniqueness of $\boldsymbol{\pi}^\top$ now follow from parts (b) and (d) of Theorem 3.11. □

It follows from Proposition 3.19 that an irreducible Markov chain on $G = (V, E)$ is uniquely determined by its conditional edge probabilities $(q_e)_{e \in E}$. That is, these conditional probabilities determine the state probability vector. We refer to the state probability vector of an irreducible Markov chain as the *stationary probability vector*.

The *entropy* (or *entropy rate*) of a Markov chain $\mathcal{P}$ on $G = (V, E)$ is defined as the expected value—with respect to the probability measure $\mathcal{P}$ on the edges of G—of the random variable $\log(1/q_e)$; i.e.,

$$\mathsf{H}(\mathcal{P}) = \mathsf{E}_{\mathcal{P}} \left\{ \log(1/q_e) \right\} = -\sum_{u \in V} \pi_u \sum_{e \in E_u} q_e \log q_e \ .$$

Example 3.7 Let G be the Shannon cover of the $(0, 1)$-RLL constrained system as shown in Figure 3.1, and consider the following stochastic matrix (whose support graph is G):

$$Q = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 1 & 0 \end{pmatrix} \ .$$

One can verify that

$$\boldsymbol{\pi}^\top = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} \end{pmatrix}$$

is a left eigenvector of Q associated with the Perron eigenvalue 1. The vector $\boldsymbol{\pi}^\top$ is the stationary probability vector of the (unique) stationary Markov chain on G whose transition matrix is Q. The entropy of this Markov chain is $-\sum_u \pi_u \sum_{e \in E_u} q_e \log q_e = 2/3$. □

Proposition 3.20 *Let Q be a primitive stochastic $|V| \times |V|$ matrix and let $\boldsymbol{\xi}^\top = (\xi_u)_{u \in V}$ be such that $\sum_{u \in V} \xi_u = 1$. Then,*

$$\lim_{\ell \to \infty} \boldsymbol{\xi}^\top Q^\ell = \boldsymbol{\pi}^\top \ ,$$

where $\boldsymbol{\pi}^\top$ is the vector as in Proposition 3.19.

Proof. Since Q is stochastic we have $Q\mathbf{1} = \mathbf{1}$, where $\mathbf{1}$ is the all-one column vector; that is, $\mathbf{1}$ is a right eigenvector associated with the Perron eigenvalue $\lambda(Q) = 1$. Hence, by Theorem 3.17 we have

$$\lim_{\ell \to \infty} \boldsymbol{\xi}^\top Q^\ell = \boldsymbol{\xi}^\top \mathbf{1}\, \boldsymbol{\pi}^\top = \boldsymbol{\pi}^\top \ ,$$

as claimed. □

Suppose that $\mathcal{P}$ is a (not necessarily stationary) Markov chain on G with an associated primitive transition matrix Q and a state probability vector $\boldsymbol{\xi}^\top$. Also, let $\boldsymbol{\pi}^\top = (\pi_u)_u$ be a left positive eigenvector of Q associated with the Perron eigenvalue 1, normalized so that $\sum_u \pi_u = 1$. It follows from Proposition 3.20 that as the lengths of paths go to infinity, the probability of terminating in state u of G converges to π_u.

Theorem 3.21 (Law of large numbers for irreducible Markov chains) *Let $\mathcal{P}$ be an irreducible Markov chain on a labeled graph $G = (V, E, L)$ where $L : E \to \mathbb{R}$ (i.e., the labels are over the real field). For a positive integer ℓ, define the random variable Z_ℓ on paths $\gamma = e_1 e_2 \ldots e_\ell$ of length ℓ in G by*

$$Z_\ell = Z_\ell(\gamma) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(e_i) \ .$$

Then, for every $\epsilon > 0$,

$$\lim_{\ell \to \infty} \mathsf{Prob}\Big\{ \, |Z_\ell - \overline{L}\,| \leq \epsilon \, \Big\} = 1 \ ,$$

where $\overline{L} = \mathsf{E}_{\mathcal{P}} \{L(e)\} = \sum_{e \in E} \mathcal{P}(e) L(e)$.

The proof of Theorem 3.21 is left as a guided exercise (see Problems 3.35 and 3.36). Observe that since $\mathcal{P}$ is stationary, we have in fact $\overline{L} = \mathsf{E}_{\mathcal{P}} \{Z_\ell\}$.

Let $\mathcal{P}$ be a Markov chain on G. A path γ in G of length ℓ is called $(\mathcal{P}, \epsilon)$-*typical* if the probability $\mathcal{P}(\gamma)$, as defined by (3.7), satisfies

$$\left| \mathsf{H}(\mathcal{P}) + \frac{1}{\ell} \log \mathcal{P}(\gamma) \right| \leq \epsilon \ ,$$

or, equivalently,

$$2^{-\ell(\mathsf{H}(\mathcal{P}) + \epsilon)} \leq \mathcal{P}(\gamma) \leq 2^{-\ell(\mathsf{H}(\mathcal{P}) - \epsilon)} \ .$$

The set of $(\mathcal{P}, \epsilon)$-typical paths of length ℓ in G will be denoted by $\mathcal{T}_\ell(\mathcal{P}, \epsilon)$ (the dependency of this set on G is implied by the dependency on $\mathcal{P}$).

Theorem 3.22 (Asymptotic Equipartition Property, in short AEP) *Let $\mathcal{P}$ be an irreducible Markov chain on a graph G. Then, for every $\epsilon > 0$,*

$$\lim_{\ell\to\infty} \sum_{\gamma\in\mathcal{T}_\ell(\mathcal{P},\epsilon)} \mathcal{P}(\gamma) = 1 \ .$$

Proof. We apply Theorem 3.21 to the graph G and the labeling $L(e) = \log(1/q_e)$, where q_e is the conditional probability of an edge e; we assume that $q_e > 0$, or else we delete the edge e from G. Here,

$$\overline{L} = \mathsf{E}_{\mathcal{P}}\left\{\log(1/q_e)\right\} = \mathsf{H}(\mathcal{P})$$

and, therefore,

$$\lim_{\ell\to\infty} \mathsf{Prob}\left\{\, |Z_\ell - \mathsf{H}(\mathcal{P})| \le \epsilon \,\right\} = 1 \ . \tag{3.8}$$

On the other hand, letting $(\pi_u)_u$ be the stationary probability vector of $\mathcal{P}$, we have for every path $\gamma = e_1 e_2 \ldots e_\ell$ in G,

$$-\frac{1}{\ell}\log\mathcal{P}(\gamma) = \frac{\sum_{i=1}^{\ell}\log(1/q_{e_i})}{\ell} + \frac{\log(1/\pi_{\sigma(e_1)})}{\ell} = Z_\ell(\gamma) + o(1) \ ,$$

where $o(1)$ stands for an expression that goes to zero as ℓ goes to infinity. Hence, by (3.8) we obtain

$$\lim_{\ell\to\infty} \sum_{\gamma\in\mathcal{T}_\ell(\mathcal{P},\epsilon)} \mathcal{P}(\gamma) = \lim_{\ell\to\infty} \mathsf{Prob}\left\{ \left| \mathsf{H}(\mathcal{P}) + \frac{1}{\ell}\log\mathcal{P}(\gamma) \right| \le \epsilon \right\} = 1 \ ,$$

as claimed. □

The AEP thus states that for large ℓ, 'most' paths of length ℓ in G have probability roughly $2^{-\ell\mathsf{H}(\mathcal{P})}$; here, the quantifier 'most' does not refer to the actual count of the paths, but rather to their measure as induced by the probability distribution $\mathcal{P}$.

We remark that Theorem 3.22 holds also for the following stronger property of paths. Let $\mathcal{P}$ be a Markov chain on G. A path γ in G of length ℓ is called $(\mathcal{P},\epsilon)$-*strongly-typical* if for every edge e in G, the number of times that e is traversed in γ is bounded from below by $\ell(\mathcal{P}(e)-\epsilon)$ and from above by $\ell(\mathcal{P}(e)+\epsilon)$. It can shown that if $\mathcal{P}$ is irreducible, then this property implies that γ is typical (see Problem 3.37). The re-statement of Theorem 3.22 for strongly-typical paths is left as an exercise (Problem 3.38).

3.5 Probabilistic characterization of capacity

Theorem 3.23 *Let S be a constrained system which is presented by an irreducible lossless graph G. Then,*

$$\sup_{\mathcal{P}} \mathsf{H}(\mathcal{P}) = \log\lambda(A_G) = \mathsf{cap}(S) \ ,$$

where the supremum is taken over all stationary Markov chains on G.

Proof. By continuity, every stationary Markov chain $\mathcal{P}$ on G can be expressed as a limit of irreducible Markov chains $\mathcal{P}_1, \mathcal{P}_2, \cdots$ on G, and $\lim_{i\to\infty} \mathsf{H}(\mathcal{P}_i) = \mathsf{H}(\mathcal{P})$. Hence, it suffices to prove the theorem for irreducible Markov chains on G.

By Theorem 3.22, for every $\epsilon, \delta > 0$ there is a positive integer N such that for every $\ell \geq N$,

$$\sum_{\gamma \in \mathcal{T}_\ell(\mathcal{P},\epsilon)} \mathcal{P}(\gamma) \geq 1 - \delta \; .$$

On the other hand, for every $\gamma \in \mathcal{T}_\ell(\mathcal{P}, \epsilon)$ we have

$$\mathcal{P}(\gamma) \leq 2^{-\ell(\mathsf{H}(\mathcal{P})-\epsilon)} \; .$$

Summing over $\gamma \in \mathcal{T}_\ell(\mathcal{P}, \epsilon)$ yields

$$1 - \delta \leq \sum_{\gamma \in \mathcal{T}_\ell(\mathcal{P},\epsilon)} \mathcal{P}(\gamma) \leq |\mathcal{T}_\ell(\mathcal{P}, \epsilon)| \cdot 2^{-\ell(\mathsf{H}(\mathcal{P})-\epsilon)} \; ,$$

or

$$|\mathcal{T}_\ell(\mathcal{P}, \epsilon)| \geq (1 - \delta) \cdot 2^{\ell(\mathsf{H}(\mathcal{P})-\epsilon)} \; .$$

Assuming that $\delta < 1$, by Lemma 3.5 we obtain

$$\mathsf{H}(\mathcal{P}) - \epsilon \leq \log \lambda(A_G)$$

for every $\epsilon > 0$. Hence, $\mathsf{H}(\mathcal{P}) \leq \log \lambda(A_G)$. To complete the proof, it suffices to exhibit a stationary Markov chain $\mathcal{P}$ on $G = (V, E)$ for which $\mathsf{H}(\mathcal{P}) = \log \lambda(A_G)$.

Let $\mathbf{x} = (x_u)_{u\in V}$ and $\mathbf{y}^\top = (y_u)_{u\in V}$ be positive right and left eigenvectors of A_G associated with the eigenvalue $\lambda = \lambda(A_G)$ and normalized so that $\mathbf{y}^\top \mathbf{x} = 1$. Define a stationary Markov chain $\mathcal{P}$ through the conditional probabilities

$$q_e = \frac{x_{\tau(e)}}{\lambda x_{\sigma(e)}} \; .$$

The entries of the transition matrix Q are given by

$$(Q)_{u,v} = \sum_{e \in E_u \,:\, \tau(e)=v} q_e = \frac{(A_G)_{u,v} x_v}{\lambda x_u} \; , \tag{3.9}$$

and it is easy to verify that Q is, indeed, a stochastic matrix on G. A simple computation shows that the vector $\boldsymbol{\pi}^\top = (\, x_u y_u \,)_{u\in V}$ satisfies $\boldsymbol{\pi}^\top Q = \boldsymbol{\pi}^\top$ and, so, it is the stationary probability vector of $\mathcal{P}$.

Now,

$$\mathsf{H}(\mathcal{P}) \;\; = \;\; -\sum_{u\in V} \pi_u \sum_{e\in E_u} q_e \log q_e$$

$$
\begin{aligned}
&= \sum_{u\in V} \pi_u \sum_{e\in E_u} q_e\Big((\log\lambda) + (\log x_u) - (\log x_{\tau(e)})\Big) \\
&= (\log\lambda) \underbrace{\sum_{u\in V} \pi_u \underbrace{\sum_{e\in E_u} q_e}_{1}}_{1} + \Big(\sum_{u\in V} (\pi_u \log x_u) \underbrace{\sum_{e\in E_u} q_e}_{1}\Big) - \Big(\sum_{u\in V} \pi_u \sum_{v\in V} (\log x_v) \underbrace{\sum_{e\in E_u\,:\,\tau(e)=v} q_e}_{(Q)_{u,v}}\Big) \\
&= (\log\lambda) + \sum_{u\in V} (\pi_u \log x_u) - \sum_{v\in V} (\log x_v) \underbrace{\sum_{u\in V} \pi_u (Q)_{u,v}}_{\pi_v} \\
&= \log\lambda\,.
\end{aligned}
$$

See also [Imm91, p. 48]. □

A stationary Markov chain $\mathcal{P}$ on G for which $\mathsf{H}(\mathcal{P}) = \log\lambda(A_G)$ is called a *maxentropic Markov chain on* G. We have shown in the proof of Theorem 3.23 that a maxentropic Markov chain exists for every irreducible graph. In fact, it can be shown that such a stationary Markov chain is unique [Par64] (see also [PT82]). It follows from Theorem 3.23 that if $\mathcal{P}$ is a maxentropic Markov chain on an irreducible graph G, then for every $\epsilon > 0$, the growth rate of the $(\mathcal{P},\epsilon)$-typical paths in G is essentially the same as the number of paths in G. We can thus say that the $(\mathcal{P},\epsilon)$-typical paths form a 'substantial subset' within the set of all paths in G. The same can be said about the $(\mathcal{P},\epsilon)$-strongly-typical paths in G and, as such, a maxentropic Markov chain defines the frequency with which a symbol appears in a 'substantial subset' of words of $S(G)$. For the analysis of such frequency in (d,k)-RLL constrained systems, see [How89].

Example 3.8 Let G be the Shannon cover of the $(0,1)$-RLL constrained system, as shown in Figure 3.1. The adjacency matrix of G is

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

with Perron eigenvalue $\lambda = (1+\sqrt{5})/2 \approx 1.618$ and an associated right eigenvector

$$\mathbf{x} = \begin{pmatrix} \lambda \\ 1 \end{pmatrix}.$$

By (3.9), the transition matrix Q of the maxentropic Markov chain on G is given by

$$Q = \begin{pmatrix} 1/\lambda & 1/\lambda^2 \\ 1 & 0 \end{pmatrix} \approx \begin{pmatrix} .618 & .382 \\ 1 & 0 \end{pmatrix}$$

and the respective stationary probability vector is

$$\boldsymbol{\pi}^\top = \left(\, \tfrac{\lambda+1}{\lambda+2} \;\; \tfrac{1}{\lambda+2} \,\right) \approx (\, .724 \;\; .276\,).$$

This means that in a 'substantial subset' of words of the $(0,1)$-RLL constrained system, approximately 27.6% of the symbols are 0.

The entropy of the maxentropic Markov chain is $\log \lambda \approx .6942$. Note that the stationary Markov chain in Example 3.7 has smaller entropy. □

3.6 Approaching capacity by finite-type constraints

The next two propositions exhibit an important feature of finite-type constrained systems.

Proposition 3.24 *Let S be a constrained system. Then, there is a sequence of finite-type constrained systems $\{S_m\}_{m=1}^{\infty}$ such that $S \subseteq S_m$ for every m and $\lim_{m\to\infty} \mathsf{cap}(S_m) = \mathsf{cap}(S)$.*

Proof. Given a positive integer m, we let S_m be the constrained system which is presented by the following deterministic graph G_m: For each word $\mathbf{w}$ of length m in S, we associate a state $u_{\mathbf{w}}$ in G_m. Given two words of length m in S, $\mathbf{w} = w_1 w_2 \ldots w_m$ and $\mathbf{z} = z_1 z_2 \ldots z_m$, we draw an edge $u_{\mathbf{w}} \xrightarrow{b} u_{\mathbf{z}}$ in G_m if and only if $b = z_m$ and $z_j = w_{j+1}$ for $j = 1, 2, \ldots, m-1$.

It is easy to verify that all paths in G_m that generate a word $\mathbf{w}$ terminate in $u_{\mathbf{w}}$. Hence, G_m has memory $\leq m$. To show that $S \subseteq S_m$, let $\mathbf{z} = z_1 z_2 \ldots z_\ell$ be a word of length $\ell \geq m$ in S. Then, by construction, the word $\mathbf{z}$ is generated in G_m by a path

$$u_{w_1 w_2 \ldots w_m} \xrightarrow{z_1} u_{w_2 w_3 \ldots w_m z_1} \xrightarrow{z_2} u_{w_3 w_4 \ldots z_m z_1 z_2} \xrightarrow{z_3} \cdots \xrightarrow{z_\ell} u_{z_{\ell-m+1} z_{\ell-m+2} \ldots z_\ell} \ .$$

Hence, $\mathbf{z}$ is a word in S_m. Furthermore, we also have $S_{m+1} \subseteq S_m$ and, so, the values of $\mathsf{cap}(S_m)$ are nonincreasing with m. Therefore, the limit $\lim_{m\to\infty} \mathsf{cap}(S_m)$ exists and it is at least $\mathsf{cap}(S)$. It remains to show that it is actually equal to $\mathsf{cap}(S)$.

Let $N(m; S)$ be the number of words of length m in S. Every word of length tm in S_m can be written as a concatenation of t words of length m in S. Therefore, the number of words of length tm in S_m is at most $(N(m; S))^t$. Thus, $\mathsf{cap}(S_m) \leq (\log N(m; S))/m$ and, so,

$$\mathsf{cap}(S) \leq \lim_{m\to\infty} \mathsf{cap}(S_m) \leq \lim_{m\to\infty} (\log N(m; S))/m = \mathsf{cap}(S) \ ,$$

as desired. □

The following dual result is proved in [Mar85].

Proposition 3.25 *Let S be a constrained system. Then, there is a sequence of finite-type constrained systems $\{S_m\}_{m=1}^{\infty}$ such that $S_m \subseteq S$ for every m and $\lim_{m\to\infty} \mathsf{cap}(S_m) = \mathsf{cap}(S)$.*

Sketch of proof (in the irreducible case): Let S_m be the constrained system which is presented by the following deterministic graph G_m: For every magic word $\mathbf{w}$ of length m in S (see Section 2.6.4), we associate a state $u_{\mathbf{w}}$ in G_m. Given two magic words of length m in S, $\mathbf{w} = w_1 w_2 \ldots w_m$ and $\mathbf{z} = z_1 z_2 \ldots z_m$, we draw an edge $u_{\mathbf{w}} \stackrel{b}{\rightarrow} u_{\mathbf{z}}$ in G_m if and only if the following three conditions hold:

(a) $z_j = w_{j+1}$ for $j = 1, 2, \ldots, m-1$;

(b) $b = z_m$;

(c) $\mathbf{w}b \in S$.

It is easy to verify that G_m has memory $\leq m$ and that $S_m \subseteq S$. The approach of $\mathsf{cap}(S_m)$ to $\mathsf{cap}(S)$ follows from the fact that most long enough words in S are magic words. The precise proof is given in [Mar85]. □

Problems

Problem 3.1 Let S be a constrained system and ℓ a positive integer. Based on the definition of capacity, show that

$$\mathsf{cap}(S^\ell) = \ell \cdot \mathsf{cap}(S) \ .$$

The following is a skeleton of a proof for the inequality $\mathsf{cap}(S^\ell) \geq \ell \cdot \mathsf{cap}(S)$. Justify each step.

Denote by Σ the alphabet of S and let $\ell_1 < \ell_2 < \cdots < \ell_i < \cdots$ be such that

$$\lim_{i\to\infty} \frac{1}{\ell_i} \log N(\ell_i; S) = \mathsf{cap}(S)$$

(why do such ℓ_i's exist?). Define $m_i = \lfloor \ell_i/\ell \rfloor$. Then

$$\begin{aligned} \mathsf{cap}(S^\ell) &\geq \limsup_{i\to\infty} \frac{1}{m_i} \log N(m_i\ell; S) \\ &\geq \limsup_{i\to\infty} \frac{1}{m_i} \log N(\ell_i; S) \\ &\geq \ell \cdot \lim_{i\to\infty} \frac{\ell_i}{m_i\ell} \lim_{i\to\infty} \frac{1}{\ell_i} \log N(\ell_i, S) \\ &= \ell \cdot \mathsf{cap}(S) \ . \end{aligned}$$

Problem 3.2 Let $\varepsilon : \mathbb{N} \to \mathbb{R}^+$ be a function such that $\lim_{m\to\infty} \varepsilon(m) = 0$ and for a positive integer h define

$$\delta(\ell) = \frac{1}{\ell} \max_{(\ell_1,\ell_2,\ldots,\ell_h)} \sum_{i=1}^{h} \ell_i \varepsilon(\ell_i) \ ,$$

where $(\ell_1, \ell_2, \ldots, \ell_h)$ ranges over all nonnegative integer h-tuples such that $\ell_1 + \ell_2 + \ldots + \ell_h \leq \ell$. Complete the proof of Theorem 3.1 by showing that $\lim_{\ell\to\infty} \delta(\ell) = 0$.

Hint: Let β be a finite upper bound on the values of $\varepsilon(m)$. Justify each of the following steps:

$$\begin{aligned}
\delta(\ell) &= \max_{(\ell_1,\ell_2,\ldots,\ell_h)} \sum_{i=1}^{h} \frac{\ell_i}{\ell}\,\varepsilon(\ell_i) \\
&= \max_{(\ell_1,\ell_2,\ldots,\ell_h)} \left(\left(\sum_{i\,:\,\ell_i \le \sqrt{\ell}} \frac{\ell_i}{\ell}\,\varepsilon(\ell_i) \right) + \left(\sum_{i\,:\,\ell_i > \sqrt{\ell}} \frac{\ell_i}{\ell}\,\varepsilon(\ell_i) \right) \right) \\
&\le \frac{\beta \cdot h}{\sqrt{\ell}} + \max_{(\ell_1,\ell_2,\ldots,\ell_h)} \left(\sum_{i\,:\,\ell_i > \sqrt{\ell}} \varepsilon(\ell_i) \right) ,
\end{aligned}$$

and then deduce that $\lim_{\ell\to\infty} \delta(\ell) = 0$.

Problem 3.3 Let $S_{d,\infty}$ denote the (d,∞)-RLL constrained system.

1. Show that
$$\mathsf{cap}(S_{d,\infty}) \le \frac{\log\,(d+2)}{d+1} .$$
Hint: Show that when a word in $S_{d,\infty}$ is divided into nonoverlapping blocks of length $d+1$, then each such block may contain at most one 1.

2. Show that for every positive integer m,
$$\mathsf{cap}(S_{d,\infty}) \ge \frac{\log\,(m+1)}{d+m} .$$
Hint: Consider the concatenation of binary blocks of length $m+d$, each containing at most one 1 which is located in one of the first m positions of the block.

3. Show that
$$\lim_{d\to\infty} \mathsf{cap}(S_{d,\infty}) \cdot \frac{d}{\log d} = 1 .$$
Hint: Substitute $m = \varepsilon d$ in 2 and let $d \to \infty$ for every fixed small ε.

Problem 3.4 Compute the capacity of the $(d,\infty,2)$-RLL constraint for $d = 0, 1$; recall that this constraint consist of all binary words in which the runlengths of 0's between consecutive 1's are even when $d = 0$ and odd when $d = 1$.

Problem 3.5 Identify the values of d and k for which the Shannon cover in Figure 1.3 is periodic.

Problem 3.6 Let G be a nontrivial irreducible graph and let G' and G'' be the Moore form and Moore co-form of G, respectively. Show that the periods of G, G', and G'' are equal.

Problem 3.7 Let G_1 and G_2 be nontrivial irreducible graphs with periods p_1 and p_2, respectively. Show that the period of each *nontrivial* irreducible component of $G_1 * G_2$ is divisible by the least common multiplier (l.c.m.) of p_1 and p_2.

Problem 3.8 A path

$$u_0 \to u_1 \to u_2 \to \cdots \to u_\ell$$

in a graph is called a *simple cycle* if $u_0 = u_\ell$ and $u_i \neq u_j$ for $0 \leq i < j < \ell$. Let G be a nontrivial irreducible graph with period p. Show that p is the greatest common divisor of the lengths of all the simple cycles in G.

Problem 3.9 Let G be a nontrivial irreducible graph with period p and let v be a state in G. Show that p is the greatest common divisor of the lengths of all cycles in G that pass through v.

Problem 3.10 Let G be a nontrivial irreducible graph with period p.

1. Show that for every pair of states u and v in G there exist nonnegative integers $m_{u,v}$ and $r_{u,v}$, such that for every integer $m \geq m_{u,v}$ there is a path in G of length $m \cdot \mathsf{p} + r_{u,v}$ originating in state u and terminating in v.
2. Show that G^{p} has p irreducible components which are primitive and isolated from each other.
3. Let ℓ be a positive integer relatively prime to p (i.e., $\gcd(\mathsf{p}, \ell) = 1$). Show that G^ℓ is irreducible.

   Hint: Make use of 1 and the fact that there is a positive integer b such that $b \cdot \mathsf{p} \equiv 1 \pmod{\ell}$.
4. Generalize 2 and 3 as follows. Let ℓ be a positive integer and $d = \gcd(\mathsf{p}, \ell)$. Show that G^ℓ composes of d isolated irreducible components, each with period p/d.

Problem 3.11 Let G be a nontrivial irreducible (not necessarily lossless) graph with period p and let $G_0, G_1, \ldots, G_{\mathsf{p}-1}$ be the irreducible components of G^{p}. Show from the definition of capacity that

$$\mathsf{cap}(S(G_i)) = \mathsf{p} \cdot \mathsf{cap}(S(G))$$

for every irreducible component G_i.

Problem 3.12 Let A be a nonnegative real square matrix and let ℓ be a positive integer. Show that the irreducible components of A^ℓ all have the same set of nonzero eigenvalues, with the same multiplicity.

Problem 3.13 Let A be the matrix given by

$$A = \begin{pmatrix} 2 & 5 & 0 \\ 0 & 1 & 6 \\ 1 & 3 & 3 \end{pmatrix}.$$

1. Compute the eigenvalues of A and respective left and right *integer* eigenvectors.
2. Find a diagonal matrix Λ and an invertible matrix P such that $A = P\Lambda P^{-1}$.
3. Compute the limit
$$B = \lim_{\ell \to \infty} \frac{1}{7^\ell} \cdot A^\ell \, .$$
4. Find integer vectors that span the row space and column space, respectively, of B.

Problem 3.14 Let A be a nonnegative real square matrix of order $m \times m$. Show that A is irreducible if and only if $(A + I)^{m-1} > 0$.

Problem 3.15 Let G be a graph with an adjacency matrix
$$A_G = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} ,$$

1. What is the period p of A_G?
2. Write the matrix A_G^{p}.
3. Compute the absolute values of the eigenvalues of A_G.

Problem 3.16 Let G be the graph in Figure 2.24.

1. What is the period p of G?
2. Find the irreducible components G_i of G^{p}.
3. For every component G_i found in 2, compute the value of $\lambda(A_{G_i})$.
4. Compute $\lambda(A_G)$.
5. For every component G_i found in 2, compute a positive *integer* eigenvector associated with $\lambda(A_{G_i})$, such that the largest entry in the vector is the smallest possible.
6. Compute an eigenvector $\mathbf{x} = (x_{u_i})_{i=1}^8$ of A_G associated with $\lambda(A_G)$ such that $x_{u_1} = 1$. How many such vectors exist?
7. Is there an *integer* eigenvector of A_G associated with $\lambda(A_G)$?
8. Repeat 6 except that now $\mathbf{x}$ is an eigenvector of A_{G^2} associated with $\lambda(A_{G^2})$.
9. Compute a nonnegative integer eigenvector of A_{G^2} associated with $\lambda(A_{G^2})$ where the largest entry in that eigenvector is the smallest possible.

Problem 3.17 Let $G_1 = (V_1, E_1, L_1)$ and $G_2 = (V_2, E_2, L_2)$ be two labeled graphs with labeling $L_1 : E_1 \to \Sigma_1$ and $L_2 : E_2 \to \Sigma_2$. The *Kronecker product* of G_1 and G_2 is the labeled graph

$$G = G_1 \otimes G_2 = (V_1 \times V_2, E, L) \ ,$$

where the set of edges E and the labeling $L : E \to \Sigma_1 \times \Sigma_2$ are defined as follows:

$$(u_1, u_2) \stackrel{(a_1,a_2)}{\longrightarrow} (v_1, v_2)$$

is an edge in E if and only if

$$u_1 \stackrel{a_1}{\to} v_1 \quad \text{and} \quad u_2 \stackrel{a_2}{\to} v_2$$

are edges in G_1 and G_2, respectively.

Let S_1, S_2, and S be the constrained systems that are generated by G_1, G_2, and G, respectively.

Hereafter assume that G_1 and G_2 are nontrivial irreducible graphs and let p_1 and p_2 denote their periods, respectively.

1. Show that
$$\mathsf{cap}(S) = \mathsf{cap}(S_1) + \mathsf{cap}(S_2)$$
(do not assume that G_1 and G_2 are lossless).

2. Show that the anticipation of G is given by
$$\mathcal{A}(G) = \max\Big\{\mathcal{A}(G_1), \mathcal{A}(G_2)\Big\} \ .$$

3. Show that if G_1 and G_2 are deterministic and have finite memory, then the memory of G is given by
$$\mathcal{M}(G) = \max\{\mathcal{M}(G_1), \mathcal{M}(G_2)\} \ .$$

4. Show that G has $\gcd(\mathsf{p}_1, \mathsf{p}_2)$ irreducible components, which are isolated from each other.

5. Show that the period of each irreducible component of G is the least common multiplier (lcm) of p_1 and p_2.

6. Show that the adjacency matrix A_G is the Kronecker (or direct) product of A_{G_1} and A_{G_2}, namely, for every (u_1, u_2) and (v_1, v_2) in V,
$$(A_G)_{(u_1,u_2),(v_1,v_2)} = (A_{G_1})_{u_1,v_1} \cdot (A_{G_2})_{u_2,v_2} \ .$$

7. Show that
$$\lambda(A_G) = \lambda(A_{G_1}) \cdot \lambda(A_{G_2})$$
(this is a known property of Kronecker product of matrices, but it can be proved also by counting paths in G_1, G_2, and G).

Problem 3.18 Let $G = (V, E, L)$ be a primitive graph and let $\lambda = \lambda(A_G)$. Denote by μ the largest absolute value of any eigenvalue of A_G other than λ (let $\mu = 0$ if $|V| = 1$). Show that the number, $\Phi_G(\ell)$, of cycles of length ℓ in G satisfies

$$|\Phi_G(\ell) - \lambda^\ell| \leq (|V| - 1)\mu^\ell \ .$$

Hint: Recall that the trace of a square matrix B, which is the sum of the entries along the main diagonal of B, is preserved in the Jordan form of B (as well as in any other matrix that is similar to B).

Problem 3.19 For $0 \leq d \leq k < \infty$, let $G(d, k)$ be the Shannon cover in Figure 1.3 of the (d, k)-RLL constrained system and let $\chi_{d,k}(z) = \det(zI - A_{G(d,k)})$ be the characteristic polynomial of $A_{G(d,k)}$.

1. Show that
$$\chi_{d,k}(z) = z^{k+1} - \sum_{j=0}^{k-d} z^j \ .$$

2. Show that $\lambda(A_{G(d,k)})$ is the largest positive solution of the equation
$$z^{k+2} - z^{k+1} - z^{k-d+1} + 1 = 0 \ .$$

3. Show that for $d = 0$,
$$\lambda(A_{G(0,k)}) \leq 2 - \frac{1}{2^{k+1}} \ .$$

4. Extending the definition of $G(d, k)$ to $k = \infty$, show that $\lambda(A_{G(d,\infty)})$ is the largest positive solution of the equation
$$z^{d+1} - z^d - 1 = 0 \ .$$

5. [AS87] Show that for $d \geq 1$,
$$\lambda(A_{G(d,\infty)}) = \lambda(A_{G(d-1,2d-1)}) \ .$$

6. [ForsB88] Show that for $d \geq 0$,
$$\lambda(A_{G(d,2d)}) = \lambda(A_{G(d+1,3d+1)}) \ .$$

Problem 3.20 [C70] For a nonnegative integer B, let G_B be the Shannon cover in Figure 1.14 of the B-charge constraint and let $\chi_B(z) = \det(zI - A_{G_B})$ be the characteristic polynomial of A_{G_B}.

1. Show that $\chi_0(z) = z$ and $\chi_1(z) = z^2 - 1$.

2. Show that for $B \geq 2$,
$$\chi_B(z) = z \cdot \chi_{B-1}(z) - \chi_{B-2}(z) \ .$$

3. Show that
$$\chi_B(2\cos x) = \frac{\sin(B+2)x}{\sin x}$$
and, so, for $|z| \leq 2$,
$$\chi_B(z) = \frac{\sin((B+2)\cos^{-1}(z/2))}{\sin(\cos^{-1}(z/2))} .$$
Hint: Make use of the trigonometric identity
$$\sin(Bx) + \sin(B+2)x = 2\cos x \cdot \sin(B+1)x .$$
The polynomials $\chi_B(2z)$ are known as *Chebyshev polynomials of the second kind*. See [AbS65, pp. 774–776].

4. Show that the eigenvalues of A_{G_B} are given by
$$\lambda_i = 2\cos\left(\frac{\pi i}{B+2}\right) , \quad i = 1, 2, \ldots, B+1 .$$

5. Let the graph H_B be obtained from G_B by adding an edge from state 0 (the leftmost state in Figure 1.14) to state B (the rightmost state), and another edge from state B to state 0. Denote by $\overline{\chi}_B(z)$ the characteristic polynomial of H_B. Show that
$$\overline{\chi}_B(2\cos x) = 2\cos(B+1)x \ - \ 2$$
and, so, for $|z| \leq 2$,
$$\overline{\chi}_B(z) = 2\cos((B+1)\cos^{-1}(z/2)) \ - \ 2 .$$
The polynomials $\overline{\chi}_B(2z) + 2$ are known as *Chebyshev polynomials of the first kind*.

6. Show that the eigenvalues of A_{H_B} are given by
$$\lambda_i = 2\cos\left(\frac{\pi i}{B+1}\right) , \quad i = 1, 2, \ldots, B+1 .$$

Problem 3.21 [Sen80] Let A be a nonnegative square matrix, not necessarily irreducible.

1. Show that there always exists a *nonnegative* real eigenvector associated with the largest eigenvalue $\lambda(A)$.
Hint: Present A as a limit of an infinite sequence of irreducible matrices A_i. Show that the largest eigenvalues $\lambda(A_i)$ converge to $\lambda(A)$, and a respective eigenvector of A can be presented as a limit of eigenvectors of (a subsequence of) the A_i's.

2. Does there always exist such an eigenvector that is *strictly positive*?

Problem 3.22 Let A be a nonnegative irreducible matrix and let μ be an eigenvalue of A. Show that there exists a nonnegative real eigenvector associated with μ only if $\mu = \lambda(A)$.

Problem 3.23 Let H be the graph in Figure 2.26. Show that $\lambda(A_H) = 2$ by finding a strictly positive vector $\mathbf{x}$ such that $A_H\mathbf{x} = 2\mathbf{x}$.

Problem 3.24 Let A be a nonnegative irreducible matrix.

1. Show that $(A)_{u,u} \leq \lambda(A)$ for every row index u. When does the inequality hold with equality?
2. Two rows in A indexed by u and u' are called *twin rows* if the following two conditions hold:
   (a) $(A)_{u,v} = (A)_{u',v}$ for every column index $v \notin \{u, u'\}$;
   (b) $(A)_{u,u} + (A)_{u,u'} = (A)_{u',u} + (A)_{u',u'}$.

   Let $\mathbf{x} = (x_v)_v$ be an eigenvector of A associated with an eigenvalue μ. Show that if u and u' index twin rows and $(A)_{u,u} - (A)_{u',u} \neq \mu$, then $x_u = x_{u'}$. Provide an example where $(A)_{u,u} - (A)_{u',u} = \mu$ and $x_u \neq x_{u'}$.
3. Show that if $\mathbf{x} = (x_v)_v$ is an eigenvector of A associated with $\lambda(A)$ and u and u' index twin rows, then $x_u = x_{u'}$.
4. Suppose that A is the adjacency matrix of a deterministic graph G and let the states u and u' index twin rows in A (note that the sets of outgoing edges of u and u' in G do not necessarily have the same sets of labels). Let the graph H be obtained from G by redirecting the incoming edges of u in G into u' and deleting state u with its outgoing edges. Show that H is irreducible and that $\mathsf{cap}(S(G)) = \mathsf{cap}(S(H))$.

Problem 3.25 Let A be a nonnegative integer irreducible matrix of order m and let $\mathbf{x} = (x_v)_v$ be a positive real eigenvector associated with $\lambda(A)$. Denote by $x_{\max}$ and $x_{\min}$ the largest and smallest entries in $\mathbf{x}$, respectively. Show that

$$\frac{x_{\max}}{x_{\min}} \leq (\lambda(A))^{m-1} .$$

Hint: Think of A as the adjacency matrix of a graph G, and show that if there is an edge from u to v in G, then $x_v = \lambda(A)x_u$.

Problem 3.26 Let G be a nontrivial irreducible lossless graph and let G' be obtained from G by deleting an edge from G. Show that

$$\mathsf{cap}(S(G')) < \mathsf{cap}(S(G)) .$$

Hint: Use Proposition 3.12.

Problem 3.27 Let G be a nontrivial irreducible graph. It follows from Proposition 3.12 that if G' is obtained from G by deleting *any* edge from G, then $\lambda(A_{G'}) < \lambda(A_G)$. Show that there exists at least one edge in G, the deletion of which produces a graph G' for which

$$\lambda(A_G) - 1 \leq \lambda(A_{G'}) \leq \lambda(A_G) .$$

Problem 3.28 Let S_1 and S_2 be irreducible constrained systems with the same capacity. Show that if $S_1 \subseteq S_2$ then $S_1 = S_2$.

Hint: Assume to the contrary that there is a word in $S_2 \setminus S_1$ whose length, ℓ, is relatively prime to the period of the Shannon cover of S_2; then consider the constrained systems S_1^ℓ and S_2^ℓ.

Problem 3.29 Let S_0 be an irreducible constrained system and let G be a graph such that $S_0 \subseteq S(G)$ and $\mathsf{cap}(S_0) = \mathsf{cap}(S(G))$. Show that there is an irreducible component H of G such that $S_0 = S(H)$.

Problem 3.30 Let S be the constrained system over the alphabet $\Sigma = \{a, b, c, d, e, f, g\}$ generated by the graph G in Figure 2.22.

1. Obtain the matrices A_G and A_{G^2}.
2. Show that $\lambda(A_G) = 3$ and find a nonnegative integer right eigenvector and a nonnegative integer left eigenvector associated with the eigenvalue 3.
3. Compute the other eigenvalues of A_G.
4. Compute the eigenvalues of A_{G^2}.
5. Compute the capacity of S.

Problem 3.31 Let G be an irreducible *lossy* graph. Show that $\mathsf{cap}(S(G)) < \log \lambda(A_G)$.

Problem 3.32 (Graphs with extremal eigenvalues [LM95])

1. Among all irreducible graphs with m states, find an irreducible graph G_m for which $\lambda(A_{G_m})$ is minimal.
2. Among all *primitive* graphs with three states, find a primitive graph H_m for which $\lambda(A_{H_3})$ is minimal.
3. Find a primitive graph H_m with m states for which $\lambda(A_{H_m})$ is minimal.

Problem 3.33 Show by example that there are *nonstationary* Markov chains $\mathcal{P}$ on irreducible graphs G such that $\mathsf{H}(\mathcal{P}) > \log \lambda(A_G)$.

Problem 3.34 Let Q be an irreducible stochastic $|V| \times V|$ matrix and let $\boldsymbol{\pi}^\top = (\pi_u)_{u \in V}$ be the vector as in Proposition 3.19. Denote by p the period of Q and let $C_0, C_1, \ldots, C_{\mathsf{p}-1}$ be the equivalence classes of the congruence relation defined on the states of the support graph of Q. Assume that rows in Q that are indexed by C_r precede those that are indexed by C_{r+1}.

1. Show that for $r = 0, 1, \ldots, \mathsf{p}-1$,
$$\sum_{u \in C_r} \pi_u = 1/\mathsf{p} \ .$$

2. Denote by $\boldsymbol{\pi}_r$ the subvector of $\boldsymbol{\pi}$ that is indexed by C_r; that is, $\boldsymbol{\pi}_r = (\pi_u)_{u \in C_r}$. Also, let $\mathbf{1}_r$ be an all-one vector of length $|C_r|$. Define the $|V| \times |V|$ matrix Π_Q by
$$\Pi_Q = \frac{1}{\mathsf{p}} \begin{pmatrix} \mathbf{1}_0 \boldsymbol{\pi}_0^\top & & & \\ & \mathbf{1}_1 \boldsymbol{\pi}_1^\top & & 0 \\ & & \ddots & \\ 0 & & & \mathbf{1}_{\mathsf{p}-1} \boldsymbol{\pi}_{\mathsf{p}-1}^\top \end{pmatrix} .$$
Show that for every $t \geq 0$,
$$Q^{\mathsf{p}t} = \Pi_Q + E^{(t)} \ ,$$
where $E^{(t)}$ satisfies
$$\sum_{u,v \in V} |(E^{(t)})_{u,v}| \leq \beta \cdot \alpha^t$$
for some $0 \leq \alpha < 1$ and $\beta > 0$ and every $t \geq 0$.
Hint: Apply Theorems 3.10 and 3.17 and make use of the rate of convergence of the limit in Theorem 3.17, as stated in (3.6).

Problem 3.35 (Autocorrelation and power spectral density) Let $\mathcal{P}$ be an irreducible Markov chain with period p on a labeled graph $G = (V, E, L)$ where $L : E \to \mathbb{R}$. Denote by $C_0, C_1, \ldots, C_{\mathsf{p}-1}$ the equivalence classes of the congruence relation defined on the states of G, and for $r = 0, 1, \ldots, \mathsf{p}-1$, let $\overline{L}_r$ be the conditional expectation
$$\overline{L}_r = \mathsf{E}_{\mathcal{P}} \left\{ L(e) \,|\, \sigma(e) \in C_r \right\} \ ,$$
where $\sigma(e) = \sigma_G(e)$ is the initial state of the edge e in G. Define the random sequence
$$\mathbf{X} = X_{-\ell} X_{-\ell+1} \ldots X_0 X_1 \ldots X_\ell$$
on paths
$$e_{-\ell} e_{-\ell+1} e_0 e_1 \ldots e_\ell$$
of length $2\ell+1$ in G by
$$X_i = L(e_i) - \overline{L}_{r(e_i)} \ ,$$
where $r(e_i)$ is the index r such that $\sigma(e_i) \in C_r$. The *autocorrelation* of $\mathbf{X}$ is defined by
$$R_{\mathbf{X}}(t, i) = \mathsf{E}_{\mathcal{P}} \left\{ X_i X_{i+t} \right\}$$
for every $-\ell \leq i, i+t \leq \ell$.

Denote by Q the transition matrix associated with $\mathcal{P}$ and by $\boldsymbol{\pi}^\top$ the state probability vector of $\mathcal{P}$.

1. Let e_i and e_{i+t} be the ith and $(i+t)$th edge, respectively, along a random path on G. Show that for every $t > 0$,

$$\mathsf{Prob}\left\{e_i = e \text{ and } e_{i+t} = e'\right\} = \mathcal{P}(e) \cdot (Q^{t-1})_{\tau(e),\sigma(e')} \cdot q_{e'} \ ,$$

where the probability is taken with respect to $\mathcal{P}$ and $q_{e'}$ is the conditional probability of the edge e'.

2. Let B be the $|V| \times |V|$ matrix whose entries are defined for every $u, v \in V$ by

$$(B)_{u,v} = \sum_{e \in E_u \,:\, \tau(e) = v} (L(e) - \overline{L}_{r(e)}) \cdot q_e \ .$$

Show that for every $t > 0$,

$$R_{\mathbf{X}}(t,i) = \boldsymbol{\pi}^\top B Q^{t-1} B \mathbf{1} \ .$$

In particular, $R_{\mathbf{X}}(t,i)$ does not depend on i (provided that $\ell \le i, i+t \le \ell$) and will therefore be denoted hereafter by $R_{\mathbf{X}}(t)$.

3. Let Π_Q be the matrix defined in part 2 of Problem 3.34. Show that $\boldsymbol{\pi}^\top B \Pi_Q = \Pi_Q B \mathbf{1} = 0$.

4. Show that there exist $0 \le \alpha < 1$ and $\beta > 0$ such that for every $t > 0$,

$$|R_{\mathbf{X}}(t)| \le \beta \cdot \alpha^{t-1} \ .$$

Hint: Apply Problem 3.34.

5. Show that the semi-infinite series

$$\sum_{t=1}^{\infty} R_{\mathbf{X}}(t) z^{-t}$$

converges for all complex values z with $|z| \ge 1$.

6. Show that

$$\sum_{t=1}^{\infty} R_{\mathbf{X}}(t) z^{-t} = \boldsymbol{\pi}^\top B (zI - Q)^{-1} B \mathbf{1}$$

for all $|z| \ge 1$, except when $z^{\mathsf{p}} = 1$ (in particular, verify that the matrix $zI - Q$ is nonsingular in this range of convergence). Extend the result also to the limit $z \to e^{\jmath 2\pi r/\mathsf{p}}$, where $\jmath = \sqrt{-1}$ and $r = 0, 1, \ldots, \mathsf{p}-1$.

7. The *power spectral density* of the random sequence $\mathbf{X}$ is defined as the two-sided Fourier transform of $R_{\mathbf{X}}(t)$; namely, it is the function $\Psi_{\mathbf{X}}(f) : [0,1] \to \mathbb{C}$ which is given by

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

(note that $R_{\mathbf{X}}(-t) = R_{\mathbf{X}}(t)$ and that by part 5 the bi-infinite sum indeed converges). Show that for every $f \in [0,1] \setminus \{r/\mathsf{p}\}_{r=0}^{\mathsf{p}-1}$,

$$\Psi_{\mathbf{X}}(f) = R_{\mathbf{X}}(0) + 2 \cdot \mathrm{Re}\left\{\boldsymbol{\pi}^\top B (e^{\jmath 2\pi f} I - Q)^{-1} B \mathbf{1}\right\} \ ,$$

where $\mathrm{Re}\{\cdot\}$ stands for the real value and

$$R_{\mathbf{X}}(0) = \sum_{e \in E} \mathcal{P}(e)(L(e) - \overline{L}_{r(e)})^2 \ .$$

8. Let $f \mapsto \Phi_{\mathbf{X}}(f)$ be the (two-sided) Fourier transform of $\mathbf{X}$, namely,

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

Show that

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

9. Let G be the Shannon cover of the 2-charge constrained system, as shown in Figure 3.2. Consider the irreducible Markov chain $\mathcal{P}$ on G that is defined by the transition matrix

$$Q = \begin{pmatrix} 0 & 1 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 1 & 0 \end{pmatrix} \ .$$

Show that $\mathcal{P}$ is a maxentropic Markov chain and that

$$\Psi_{\mathbf{X}}(f) = 1 - \cos 2\pi f \ .$$

Problem 3.36 The purpose of this problem is proving Theorem 3.21 for every irreducible Markov chain $\mathcal{P}$ on G. As with the case of independent random variables, the proof is based upon Chebyshev inequality, which states that for every random variable Y with zero expected value,

$$\mathsf{Prob}\left\{|Y| \geq \epsilon\right\} = \mathsf{Prob}\left\{\frac{Y^2}{\epsilon^2} \geq 1\right\} \leq \mathsf{E}\left\{\frac{Y^2}{\epsilon^2}\right\} = \frac{\mathsf{E}\left\{Y^2\right\}}{\epsilon^2} \ .$$

Letting p be the period of G, we use the notations C_r and $\overline{L}_r$ as in Problem 3.35.

For a positive integer ℓ, define the random variable $Y_\ell(\gamma)$ on a path $\gamma = e_1 e_2 \dots e_\ell$ by

$$Y_\ell(\gamma) = Y_\ell = \frac{1}{\ell} \sum_{i=1}^{\ell} X_i \ ,$$

where

$$X_i = L(e_i) - \overline{L}_{r(e)}$$

and $r(e)$ is the index r such that $\sigma_G(e) \in C_r$. Hereafter, all the expectations are taken with respect to $\mathcal{P}$.

1. Show that

$$\overline{L} = \frac{1}{\mathsf{p}} \sum_{r=0}^{\mathsf{p}-1} \overline{L}_r \ .$$

Hint: Apply part 1 of Problem 3.34.

2. Show that $\mathsf{E}\{Y_\ell\} = \mathsf{E}\{X_i\} = 0$.

3. Let Z_ℓ and $\overline{L}$ be defined as in the statement of Theorem 3.21. Show that $Y_\ell = Z_\ell - \overline{L}$.

4. Show that there is a real constant η such that

$$\mathsf{E}\left\{Y_\ell^2\right\} \le \frac{1}{\ell}\left(\mathsf{E}\left\{X_1^2\right\} + \eta\right) .$$

Hint: Write

$$\mathsf{E}\left\{\left(\sum_{i=1}^{\ell} X_i\right)^2\right\} = \sum_{i=1}^{\ell}\left(\mathsf{E}\{X_i^2\} + 2\sum_{t=1}^{\ell-i}\mathsf{E}\{X_i X_{i+t}\}\right) .$$

Then use part 4 in Problem 3.35.

5. Complete the proof of Theorem 3.21 by applying Chebyshev inequality to Y_ℓ.

Problem 3.37 Let $\mathcal{P}$ be an irreducible Markov chain on G. Show that for every $\delta > 0$ there exist ϵ and N such that every $(\mathcal{P}, \epsilon)$-strongly-typical path γ of length $\ell \ge N$ in G is $(\mathcal{P}, \delta)$-typical.

Hint: Let $\boldsymbol{\pi} = (\pi_u)_u$ be the stationary probability vector of $\mathcal{P}$ and let q_e denote the conditional probability of an edge e in G. Consider a $(\mathcal{P}, \epsilon)$-strongly-typical path of length ℓ in G that starts at state u_0. First argue that

$$\pi_{u_0} \prod_{e\,:\,q_e>0} q_e^{\ell(\pi_{\sigma(e)}q_e+\epsilon)} \le \mathcal{P}(\gamma) \le \pi_{u_0} \prod_{e\,:\,q_e>0} q_e^{\ell(\pi_{\sigma(e)}q_e-\epsilon)} .$$

Then deduce that

$$\left| \mathsf{H}(\mathcal{P}) + \frac{\log \mathcal{P}(\gamma)}{\ell} - \frac{\log \pi_{u_0}}{\ell} \right| \le -\epsilon \sum_{e\,:\,q_e>0} \log q_e .$$

Finally, given δ, pick ϵ and N so that

$$\delta \ge -\frac{\log\left(\min_u \pi_u\right)}{N} - \epsilon \sum_{e\,:\,q_e>0} \log q_e .$$

Problem 3.38 Show that Theorem 3.22 holds also when γ ranges over all $(\mathcal{P}, \epsilon)$-strongly-typical paths of length ℓ.

Hint: Apply Theorem 3.21 with $L : E \to \mathbb{R}$ being the indicator function $\mathcal{I}_e$ of an edge $e \in E$; i.e., $\mathcal{I}_e(e')$ takes the value 1 if $e' = e$ and is zero for $e' \in E \setminus \{e\}$.

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 $u \stackrel{s/a}{\rightarrow} v$ 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

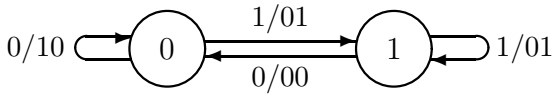


Figure 4.1: Rate $1 : 2$ two-state encoder for $(1, 3)$-RLL constrained system.

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,

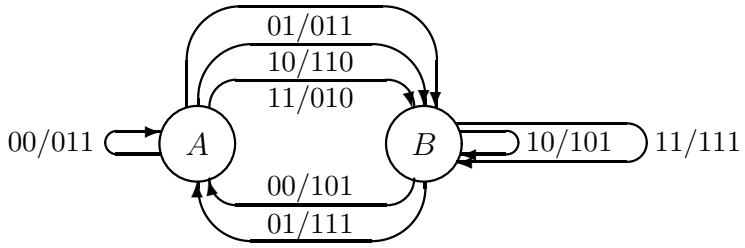


Figure 4.2: Rate $2 : 3$ two-state encoder for $(0, 1)$-RLL constrained system.

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 $p/q \leq \mathsf{cap}(S)$ (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 $p/q \leq \mathsf{cap}(S)$. □

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.

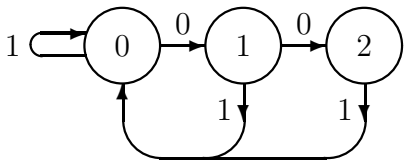


Figure 4.3: Presentation G of $(0,2)$-RLL constrained system.

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 $e_{-\mathsf{m}} e_{-\mathsf{m}+1} \dots e_0 \dots e_{\mathsf{a}}$ and $e'_{-\mathsf{m}} e'_{-\mathsf{m}+1} \dots e'_0 \dots e'_{\mathsf{a}}$ 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 $\mathbf{w} = w_0 w_1 w_2 \cdots$ is any symbol sequence generated by the encoder from the input tag sequence $\mathbf{s} = s_0 s_1 s_2 \cdots$, then, for $i \geq \mathsf{m}$,

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

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 $\mathbf{w} = w_0 w_1 w_2 \cdots$ is any sequence of q-blocks (codewords) generated by the encoder from the input tag sequence of p-blocks $\mathbf{s} = s_0 s_1 s_2 \cdots$, then, for $i \geq \mathsf{m}$,

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

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

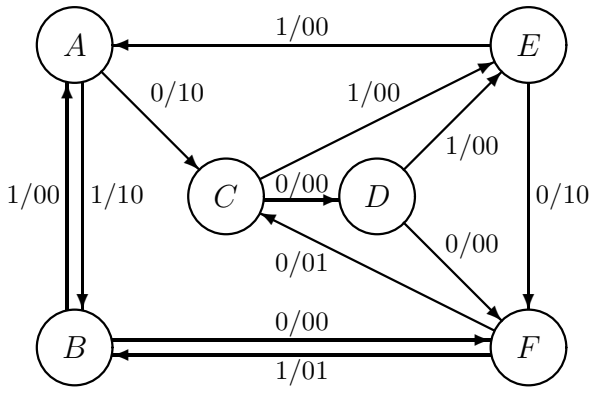


Figure 4.4: Rate $1 : 2$ six-state encoder for $(2, 7)$-RLL constrained system.

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

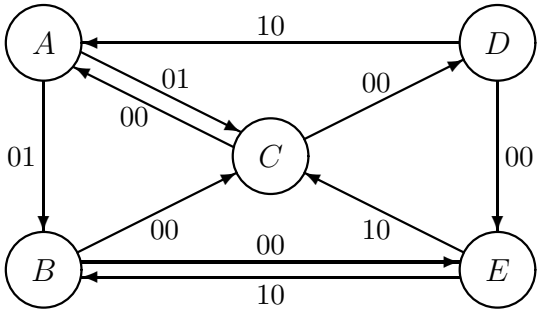


Figure 4.5: Rate 1 : 2 five-state encoder for $(2, 7)$-RLL constrained system.

is 4 (see Problem 2.4). □

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

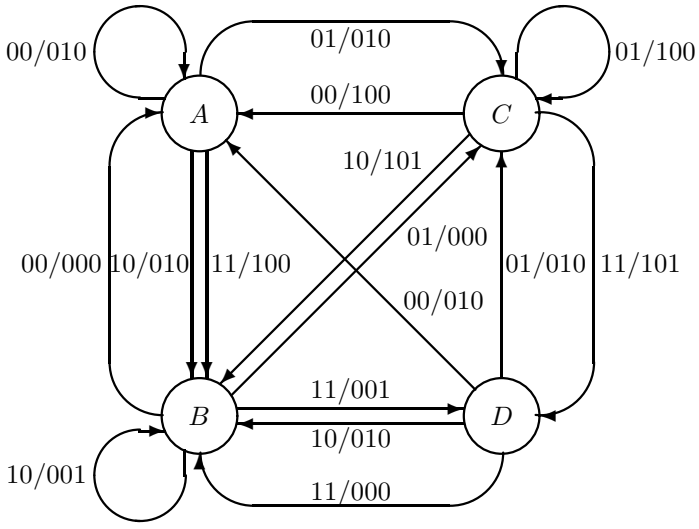


Figure 4.6: Rate 2 : 3 two-state encoder for $(1, 7)$-RLL constrained system.

is due to Weathers and Wolf [WW91] and is $(0, 2)$-sliding-block decodable (a sliding-block decoder can be found in [WW91]). □

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 $u \stackrel{s/a}{\to} v$ and $u' \stackrel{s'/a'}{\to} v'$ 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.

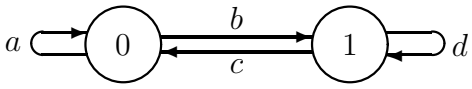


Figure 4.7: Graph presentation for Example 4.10.

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. □

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 $p/q \leq \mathsf{cap}(S)$, 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. □

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 $L(e) = w_1 w_2 \dots w_q$ becomes a path γ_e in $\mathcal{E}'$ that takes the form

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

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

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

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

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

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

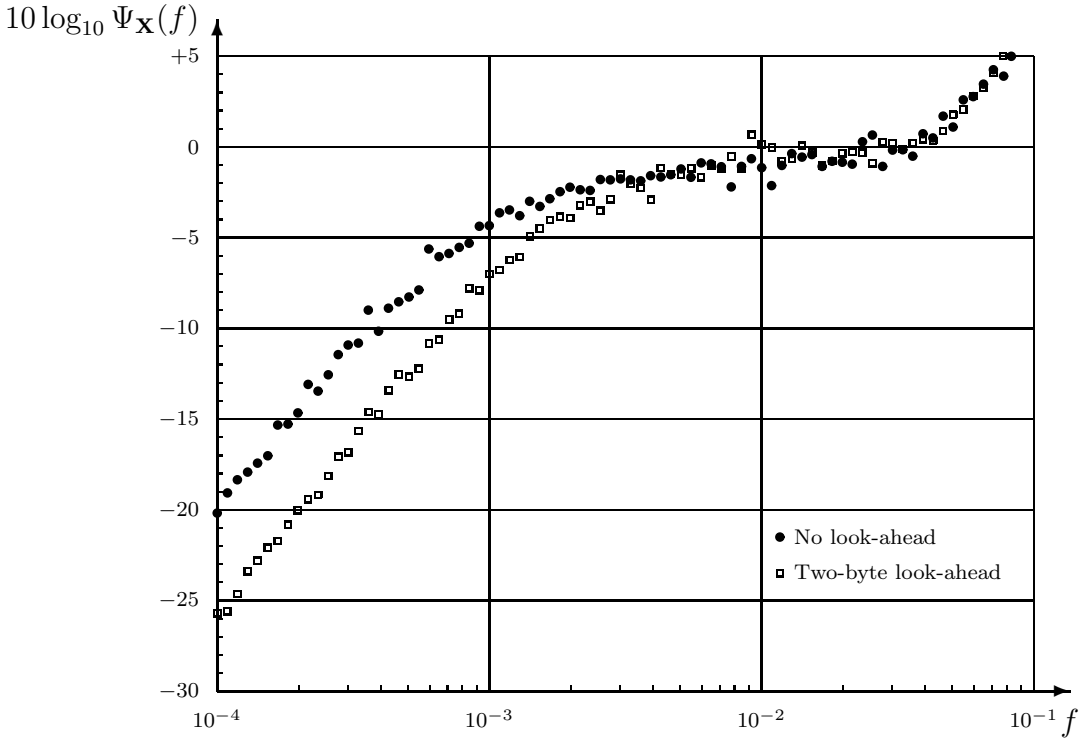


Figure 4.8: Power spectral density of the 9-EFM(16) code.

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 $\mathbf{w} = w_1 w_2 \dots w_\ell$ and $\mathbf{z} = z_1 z_2 \dots z_\ell$ in $\mathcal{L}$, we say that $\mathbf{w} < \mathbf{z}$ if there is $i \in \{1, 2, \dots, \ell\}$ such that $w_j = z_j$ for $1 \le j < i$ and $w_i < z_i$.

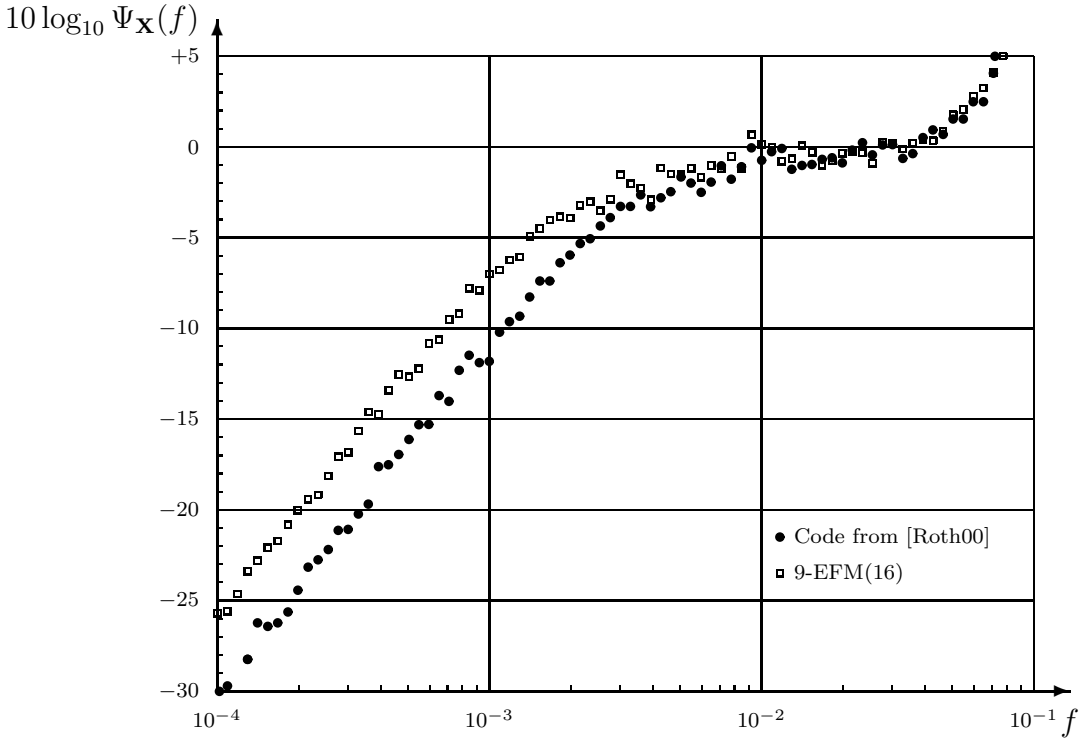


Figure 4.9: Improvement on the dc suppression compared to the 9-EFM(16) code.

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 $i \leq \ell$ 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 $0 \leq r < |\mathcal{L}|$, 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 $0 \leq r < |\mathcal{L}_G(u, v; \ell)|$, 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_C(w_1, w_2, ..., w_{i-1}, a))    {
          s ← s − N_C(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.

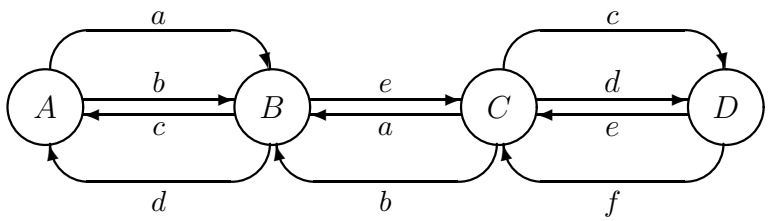


Figure 4.11: Graph G for Problem 4.6.

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

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

Problem 4.8 Prove Proposition 4.10.

Chapter 5

The State-Splitting Algorithm

In this chapter, we provide an exposition of the state-splitting algorithm, which implements the proof of Theorem 4.1, for constructing finite-state encoders. The steps in the algorithm are summarized in Figure 5.9.

The approach we will follow uses graph construction techniques, based on state splitting and approximate eigenvectors, which have their roots in symbolic dynamics, where they were introduced by R.F. Williams [Will73] and Adler, Goodwyn, and Weiss [AGW77]. The first application of state-splitting ideas in constrained coding was Patel's construction of the Zero-Modulation (ZM) code [Patel75] (see Section 4.5). The state-splitting algorithm is also related to earlier ideas of Franaszek [Fra80b], [Fra82], [Fra89].

For a given deterministic presentation G of a constrained system S and an achievable rate $p/q \leq \mathsf{cap}(S)$, we will apply a state-splitting transformation iteratively beginning with the qth power graph G^q; the procedure culminates in a new presentation of S^q with minimum out-degree at least 2^p; then, after deleting edges, we get an $(S^q, 2^p)$-encoder, which, when tagged, gives our desired rate $p : q$ finite-state encoder for S.

Although the design procedure can be made completely systematic—in the sense of having the computer automatically generate an encoder and decoder for any valid code rate—the application of the method to just about any nontrivial code design problem will benefit from the interactive involvement of the code designers. There are some practical tools that can help the designer make "good" choices during the construction process. We will discuss some of these tools in Section 5.5.

It should be stressed that the general problem of designing codes that achieve, for example, the minimum number of encoder states, minimum sliding-block decoding window, or the less precise feature of minimum hardware complexity, is not solved. This remains an active research topic, as exemplified by recent papers where lower bounds on the number of encoder states [MR91] and the minimum sliding-block decoder window are studied [Ash88],

[Kam89], [Imm92], [Holl95], [AM97], [AM00]. See Chapters 6 and 7 for more on this.

5.1 State splitting

In this section, we define state splitting of a labeled graph H and later apply it to $H = G^q$. We begin with a simplified special case.

Let $H = (V, E, L)$ be a labeled graph and denote by E_u the set of outgoing edges from state u in H. A *basic out-splitting* at state u is determined by a partition

$$E_u = E_u^{(1)} \cup E_u^{(2)}$$

of E_u into two disjoint sets. This partition is used to define a new labeled graph $H' = (V', E', L')$ that changes the local picture at state u. The set of states V' consists of all states $v \neq u$ in H, as well as two new states denoted $u^{(1)}$ and $u^{(2)}$:

$$V' = (V - \{u\}) \cup \{u^{(1)}, u^{(2)}\} .$$

The states $u^{(1)}$ and $u^{(2)}$ are called *descendant states* of state u, and state u is called the *parent state* of $u^{(1)}$ and $u^{(2)}$.

The edges in H' that do not involve states $u^{(1)}$ and $u^{(2)}$ are inherited from H. That is, if there is an edge e from state v to state v' in H, (with $v, v' \neq u$) there is a corresponding edge in H'. For edges involving state u, we consider the following three cases.

Case 1: Let edge e in H start at a state $v \neq u$ and terminate in state u. This edge is replicated in H' to produce two edges: an edge $e^{(1)}$ from v to $u^{(1)}$ and an edge $e^{(2)}$ from v to $u^{(2)}$.

Case 2: Let edge e in H start at state u and terminate in a state $v \neq u$, and suppose e belongs to the set $E_u^{(i)}$ in the partition of E_u. We draw in H' a corresponding edge from state $u^{(i)}$ to state v.

Case 3: Let edge e be a self-loop at state u in H, and suppose that e belongs to $E_u^{(i)}$. In H' there will be two edges from state $u^{(i)}$ corresponding to e: one edge to state $u^{(1)}$, the other to state $u^{(2)}$.

As with states, we refer to *descendant edges* in H' and *parent edges* in H. In all cases, the edge label of an edge in H' is the edge label of its parent edge in H.

In specifying the partitions in particular examples, we will refer to the edges by their edge labels in cases where this causes no ambiguity.

The change in the local picture at state u is shown in Figures 5.1 and 5.2. In the figures, we have partitioned the set of edges E_u into subsets, $E_u^{(1)} = \{a, b\}$ and $E_u^{(2)} = \{c\}$. The

state u splits into two states, $u^{(1)}$ and $u^{(2)}$, according to the partition. It is evident that the anticipation at states v_1 and v_2 may increase by one symbol. So, H' need not be deterministic even if H is.

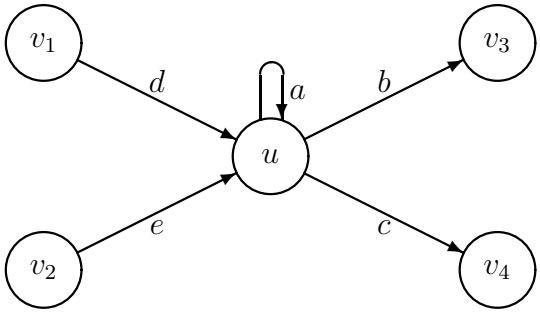


Figure 5.1: Local picture at state u before splitting.

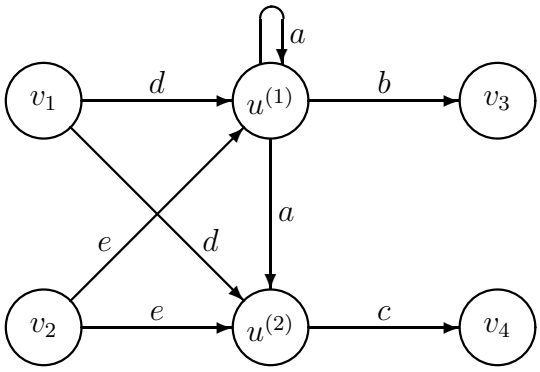


Figure 5.2: Basic out-splitting at state u for Figure 5.1.

In general, a state splitting may involve partitions into any number of subsets, and several states may be split simultaneously; so, we have the following more general notion of state splitting.

An *out-splitting* of a labeled graph H begins with a partition of the set, E_u, of outgoing edges for each state u in H into $N(u)$ disjoint subsets

$$E_u = E_u^{(1)} \cup E_u^{(2)} \cup \cdots \cup E_u^{(N(u))} \ .$$

From the partition, we derive a new labeled graph H'. The set of states $V_{H'}$ consists of the descendant states $u^{(1)}, u^{(2)}, \ldots, u^{(N(u))}$ for every $u \in V_H$. Outgoing edges from state u in H are partitioned among its descendant states and replicated in H' to each of the descendant terminal states as follows: for each edge e from u to v in H, determine the partition element $E_u^{(i)}$ to which e belongs, and endow H' with edges $e^{(r)}$ from $u^{(i)}$ to $v^{(r)}$ for $r = 1, 2, \ldots, N(v)$; the label of $e^{(r)}$ in H' is the same as the label of e in H.

Sometimes an out-splitting is called a *round* of out-splitting to indicate that several states may have been split simultaneously.

The labeled graph H' obtained from H by out-splitting is sometimes called an out-splitting of H. It has several important characteristics, relative to H, enumerated in the following proposition.

Proposition 5.1 *Let H be a labeled graph and let H' be obtained from H by out-splitting. Then*

1. *$S(H') = S(H)$.*
2. *If H has anticipation $\mathcal{A}$, then H' has anticipation at most $\mathcal{A}+1$.*
3. *If H is (m, a)-definite, then H' is $(\mathsf{m}, \mathsf{a}+1)$-definite.*
4. *If H is irreducible, so is H'.*

The key to this result is the following fact, which is a consequence of the definition of out-splitting.

Lemma 5.2 *Let H be a labeled graph and let H' be obtained from H by out-splitting. Then $e_1^{(r_1)} e_2^{(r_2)} \dots e_\ell^{(r_\ell)}$ is a path in H' if and only if $e_1 e_2 \dots e_\ell$ is a path in H and*

$$e_{i+1} \in E^{(r_i)}_{\tau(e_i)} \quad \text{for} \quad i = 1, 2, \dots, \ell-1 \, .$$

Moreover, both paths generate the same word.

We leave the proof of the lemma to the reader.

Proof of Proposition 5.1. 1. This follows immediately from Lemma 5.2.

2. Let $e_1^{(r_1)} e_2^{(r_2)} \dots e_{\mathcal{A}+2}^{(r_{\mathcal{A}+2})}$ and $\hat{e}_1^{(s_1)} \hat{e}_2^{(s_2)} \dots \hat{e}_{\mathcal{A}+2}^{(s_{\mathcal{A}+2})}$ be paths of length $\mathcal{A}+2$ in H' starting at the same state and generating the same word. Then $e_1 e_2 \dots e_{\mathcal{A}+2}$ and $\hat{e}_1 \hat{e}_2 \dots \hat{e}_{\mathcal{A}+2}$ are paths in H that start at the same initial state and generate the same word. Thus $e_1 = \hat{e}_1$ and $e_2 = \hat{e}_2$. Since e_2 belongs to the partition element $E^{(r_1)}_{\tau(e_1)}$ and $\hat{e}_2$ belongs to the partition element $E^{(s_1)}_{\tau(\hat{e}_1)} = E^{(s_1)}_{\tau(e_1)}$, it then follows that $r_1 = s_1$. So, $e_1^{(r_1)} = \hat{e}_1^{(s_1)}$. Thus, H' has anticipation at most $\mathcal{A}+1$.

3. The proof of this is similar to 2 and is left to the reader.

4. Let $u^{(s)}$ and $v^{(t)}$ be states in H' and let $\gamma = e_1 e_2 \dots e_\ell$ be any path in H from u to v such that $e_1 \in E^{(s)}_u$. Then, by Lemma 5.2, there is a path $\gamma' = e_1^{(r_1)} e_2^{(r_2)} \dots e_\ell^{(r_\ell)}$ in H' where

the r_i are determined by $e_{i+1} \in E^{(r_i)}_{\tau(e_i)}$ for $i = 1, 2, \ldots \ell{-}1$ and $r_\ell = t$. Observe that γ' is a path from $u^{(s)}$ to $v^{(t)}$ in H'. □

The *complete out-splitting* of a labeled graph H is the out-splitting obtained from the partition in which each set in the partition consists of a single, distinct edge. The resulting graph is exactly the Moore co-form of H, as was defined in Section 2.2.7.

We also have the notion of *in-splitting* obtained by reversing the roles of outgoing and incoming edges in the definition of out-splitting.

Finally, we mention that the out-splitting graph transformation can be described in terms of adjacency matrices as follows. A 0–1 matrix is called a *division matrix* if it has exactly one 1 in each column and at least one 1 in each row. Now, given an out-splitting H' of H, let D be the division matrix with rows indexed by states of H and columns indexed by states of H', defined by

$$D_{u,v^{(i)}} = \begin{cases} 1 & \text{if } u = v \\ 0 & \text{if } u \neq v \end{cases} ,$$

and let C be the matrix with rows indexed by states of H' and columns indexed by states of H, defined by

$$C_{v^{(i)},u} = \text{number of edges in } E^{(i)}_v \text{ which terminate in } u \, .$$

Then one can check that

$$A_H = DC \quad \text{and} \quad A_{H'} = CD \, .$$

Conversely, if there is a division matrix D and a matrix C with nonnegative integer entries such that $A_H = DC$ and $A_{H'} = CD$, then H' is an out-splitting of H; we leave the proof of this to the reader.

5.2 Approximate eigenvectors and consistent splitting

The state-splitting algorithm that we will present starts with a deterministic graph presentation of a given constrained system S and, through a sequence of rounds of out-splitting, ends up with an (S, n)-encoder. One key question to answer is which states to split and how to split them. Approximate eigenvectors, to be discussed next, serve as such a guide. In fact, as we show in Section 5.6 and Chapter 7, approximate eigenvectors are useful not only for the synthesis of finite-state encoders, but also for analyzing them.

5.2.1 Approximate eigenvectors

Given a nonnegative integer square matrix A and an integer n, an (A, n)-*approximate eigenvector* is a nonnegative integer vector $\mathbf{x} \not\equiv \mathbf{0}$ such that

$$A\mathbf{x} \geq n\mathbf{x} \ ,$$

where the (weak) inequality holds componentwise. We refer to this inequality as the *approximate eigenvector inequality*. The set of all (A, n)-approximate eigenvectors is denoted $\mathcal{X}(A, n)$.

When A is the adjacency matrix of a graph G, the approximate eigenvector inequality has a very simple meaning in terms of G. Think of the vector $\mathbf{x} = (x_u)_{u \in V_G}$ as assigning *state weights*: the weight of state u is x_u. Now assign *edge weights* to the edges of the graph according to their terminal states: the weight of an edge e is given by $x_{\tau_G(e)}$. Recalling that E_u denotes the set of outgoing edges from state u in G, the approximate eigenvector inequality can be written as the set of simultaneous scalar inequalities, one for each state u,

$$\sum_{e \in E_u} x_{\tau_G(e)} \geq n x_u \quad \text{for every } u \in V_G \ .$$

That is, the sum of the weights of the outgoing edges from a given state u is at least n times the weight of the state u itself.

Example 5.1 Let G be the presentation of the $(0, 1)$-RLL constrained system shown in Figure 2.2. The third power of G is shown in Figure 2.10, and the adjacency matrix $A_{G^3} = A_G^3$ of G^3 satisfies

$$A_G^3 \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 8 \\ 5 \end{pmatrix} \geq 4 \begin{pmatrix} 2 \\ 1 \end{pmatrix} \ .$$

Therefore, the vector $\mathbf{x} = (2\ 1)^\top$ is an $(A_{G^3}, 4)$-approximate eigenvector. □

The following result is straightforward.

Proposition 5.3 *For a graph G, the all-one vector $\mathbf{1}$ is an (A_G, n)-approximate eigenvector if and only if G has minimum out-degree at least n. Also, a 0–1 vector is an (A_G, n)-approximate eigenvector if and only if G has a subgraph with minimum out-degree at least n.*

The next result tells us that approximate eigenvectors exist when we need them.

Theorem 5.4 *Let A be a nonnegative integer square matrix and let n be a positive integer. Then*

$$\mathcal{X}(A,n) \neq \emptyset \quad \text{if and only if} \quad \lambda(A) \geq n \ .$$

Furthermore, if A is irreducible and $\lambda(A) = n$, then every (A,n)-approximate eigenvector is a right eigenvector associated with the eigenvalue n.

Proof. *Sufficiency:* Assume that $\lambda(A) \geq n$. We first show that there is an (A,n)-approximate eigenvector under the assumption that A is irreducible. We distinguish between the following two cases.

Case 1: $\lambda(A) > n$. By Theorem 3.11(b), A has a strictly positive right eigenvector $\mathbf{y}$ associated with $\lambda(A)$. We first perturb the entries of $\mathbf{y}$ to obtain a new vector $\hat{\mathbf{y}}$, with positive rational entries, that satisfies the inequality $A\hat{\mathbf{y}} \geq n\hat{\mathbf{y}}$. Now, let $\mathbf{x}$ be the vector obtained from $\hat{\mathbf{y}}$ by clearing denominators—i.e., by multiplying $\hat{\mathbf{y}}$ by a common multiple of the denominators of its entries. The vector $\mathbf{x}$ has positive integer entries, and it satisfies the approximate eigenvector inequality since $\hat{\mathbf{y}}$ does. Thus, $\mathbf{x}$ is an (A,n)-approximate eigenvector.

Case 2: $\lambda(A) = n$. Since $\lambda(A) = n$ is an eigenvalue of A, there is a nontrivial solution (i.e., *not* the zero vector) to the homogeneous linear system of equations

$$(A - nI)\mathbf{y} = \mathbf{0} \ .$$

Since the coefficients of this linear system are rational numbers, we can assume, by applying Gaussian elimination, that $\mathbf{y}$ has rational entries. Clearing denominators, we obtain a solution $\mathbf{x}$ with integer entries. By Theorem 3.11(d) it follows that this solution is an integer eigenvector associated with $\lambda(A) = n$, and by Theorem 3.11(b) it is strictly positive (possibly after multiplying each of its entries by -1). Hence, $\mathbf{x}$ is a positive integer eigenvector and, as such, it is an (A,n)-approximate eigenvector. This completes the proof of sufficiency in case A is irreducible.

If A is a $k \times k$ reducible matrix, then there is, by Theorem 3.15(a), an irreducible component B of A with $\lambda(B) = \lambda(A) \geq n$. Thus, by what we have just proved, there is a (B,n)-approximate eigenvector $\mathbf{x}$. We extend $\mathbf{x}$ to a vector with k entries simply by setting to zero the entries that are indexed by columns of A that do not contain columns of B. This new vector is an (A,n)-approximate eigenvector.

Necessity: Suppose first that A is irreducible and let $\mathbf{x}$ be an (A,n)-approximate eigenvector. Also, let $\mathbf{z}$ be a left eigenvector associated with the eigenvalue $\lambda = \lambda(A)$. By Theorem 3.11(b), the eigenvector $\mathbf{z}$ can be assumed to be strictly positive. Hence,

$$\lambda\mathbf{z}\mathbf{x} = \mathbf{z}\lambda\mathbf{x} = \mathbf{z}A\mathbf{x} \geq n\mathbf{z}\mathbf{x} \ .$$

Noting that $\mathbf{z}\mathbf{x} > 0$, we thus obtain $\lambda \geq n$, with equality ($\lambda = n$) if and only if $A\mathbf{x} = n\mathbf{x}$. Therefore, if $\lambda = n$, every (A,n)-approximate eigenvector is a right eigenvector associated with the eigenvalue n.

Finally, suppose that A is reducible and let $\tilde{A}$ be the matrix obtained by removing the rows and columns of A that contain the irreducible components of A whose columns all index zero components in $\mathbf{x}$. Let B be an irreducible sink in $\tilde{A}$ and let $\mathbf{y}$ be the subvector of $\mathbf{x}$ which is indexed by the columns of B in A. It can be readily verified that the vector $\mathbf{y}$ is a (B, n)-approximate eigenvector and, so, $n \leq \lambda(B) \leq \lambda(\tilde{A}) \leq \lambda(A)$. □

5.2.2 Computing approximate eigenvectors

In this section, we describe an algorithm for computing (A, n)-approximate eigenvectors. The algorithm is due to Franaszek [Fra82, Appendix] (see also [ACH83, Appendix]), and its running time is proportional to the *values* (rather than the size of the bit representations) of the computed approximate eigenvector [MR91].

The Franaszek algorithm is presented in Figure 5.3. The input to the algorithm is a

---

$\mathbf{y} \leftarrow \boldsymbol{\xi}$;
$\mathbf{x} \leftarrow \mathbf{0}$;
`while` $(\mathbf{x} \neq \mathbf{y})$ {
  $\mathbf{x} \leftarrow \mathbf{y}$;
  $\mathbf{y} \leftarrow \min\left\{\left\lfloor \frac{1}{n} A\mathbf{x} \right\rfloor, \mathbf{x}\right\}$; /* apply $\lfloor\cdot\rfloor$ and $\min\{\cdot,\cdot\}$ componentwise */
}
`return` $\mathbf{x}$;

Figure 5.3: Franaszek algorithm for computing (A, n)-approximate eigenvectors.

nonnegative integer square matrix A, a positive integer n, and a nonnegative integer vector $\boldsymbol{\xi}$. The output is a nonnegative integer vector $\mathbf{x}$, the properties of which are summarized in Proposition 5.5(b) below.

For a nonnegative integer square matrix A, a positive integer n, and a nonnegative integer vector $\boldsymbol{\xi} = (\xi_u)_u$, let $\mathcal{X}(A, n; \boldsymbol{\xi})$ denote the set of all elements $\mathbf{x} = (x_u)_u$ of $\mathcal{X}(A, n)$ that are dominated by $\boldsymbol{\xi}$ (i.e., $x_u \leq \xi_u$ for all u, or, in short, $\mathbf{x} \leq \boldsymbol{\xi}$). Also, for a vector $\mathbf{y} = (y_u)_u$, we will use the notations $\|\mathbf{y}\|_1$ and $\|\mathbf{y}\|_\infty$ for $\sum_u |y_u|$ and $\max_u |y_u|$, respectively.

Proposition 5.5 *Let A be a nonnegative integer square matrix and let n be a positive integer.*

(a) If $\mathbf{x}, \mathbf{x}' \in \mathcal{X}(A, n)$, then the vector defined by $Bigl(\max(x_u, x'_u)\Big)_u$ belongs to $\mathcal{X}(A, n)$. Thus, for any nonnegative integer vector $\boldsymbol{\xi}$ there is a largest (componentwise) element of $\mathcal{X}(A, n; \boldsymbol{\xi})$ (provided of course that $\mathcal{X}(A, n; \boldsymbol{\xi}) \neq \emptyset$).

(b) The Franaszek algorithm eventually halts for any input vector $\boldsymbol{\xi}$; and the output is either the zero vector (if $\mathcal{X}(A, n; \boldsymbol{\xi}) = \emptyset$) or the largest (componentwise) element of $\mathcal{X}(A, n; \boldsymbol{\xi})$.

Proof. Part (a) is a straightforward computation. As for part (b), let $\mathbf{x}$ be an element of $\mathcal{X}(A, n; \boldsymbol{\xi})$ and let $\mathbf{y}_m$ denote the value of $\mathbf{y}$ at the beginning of the mth iteration of the main loop of the algorithm. We show inductively on m that $\mathbf{x} \leq \mathbf{y}_m$. Clearly, this holds for $m = 1$, where we have $\mathbf{y}_1 = \boldsymbol{\xi}$. Now, assuming that $\mathbf{x} \leq \mathbf{y}_m$, we also have $\mathbf{x} \leq \frac{1}{n} A\mathbf{x} \leq \frac{1}{n} A\mathbf{y}_m$. Since $\mathbf{x}$ is an integer vector we obtain

$$\mathbf{x} \leq \min\left\{ \left\lfloor \frac{1}{n} A\mathbf{y}_m \right\rfloor, \mathbf{y}_m \right\} = \mathbf{y}_{m+1} \ .$$

It remains to show that the algorithm halts and produces the required vector $\mathbf{x}$. Assume first that $\mathbf{y}_{m+1} \neq \mathbf{y}_m$. Recalling that $\mathbf{y}_{m+1}$ is dominated by $\mathbf{y}_m$, we must have $\|\mathbf{y}_{m+1}\|_1 < \|\mathbf{y}_m\|_1$. Now, all vectors involved are nonnegative integer vectors and, therefore, the algorithm must eventually halt with $\mathbf{y}_{m+1} = \mathbf{y}_m$. At this point we have $\mathbf{y}_m = \mathbf{y}_{m+1} \leq \left\lfloor \frac{1}{n} A\mathbf{y}_m \right\rfloor$. Hence, either $\mathbf{y}_m \in \mathcal{X}(A, n; \boldsymbol{\xi})$ or $\mathbf{y}_m = \mathbf{0}$. Furthermore, if $\mathcal{X}(A, n; \boldsymbol{\xi}) \neq \emptyset$, we must have $\mathbf{y}_m \neq \mathbf{0}$, as $\mathbf{y}_m$ dominates any vector in $\mathcal{X}(A, n; \boldsymbol{\xi})$. □

By the proof of Proposition 5.5, it also follows that the number of iterations of the main loop of the algorithm is at most $\|\boldsymbol{\xi}\|_1 + 1$.

Now, suppose that A is a nonnegative integer $k \times k$ matrix and we would like to compute the vector in $\mathcal{X}(A, n)$ with the smallest norm $\|\mathbf{x}\|_\infty$. In order to find such a vector, we apply the Franaszek algorithm to vectors of the form $\boldsymbol{\xi} = \xi \cdot \mathbf{1}$ (namely, to integer multiples of the all-one vector), searching for the smallest positive integer ξ for which the algorithm produces a nonzero vector $\mathbf{x}$. By Theorem 5.4, there exists such ξ if $n \leq \lambda(A_G)$. Performing a binary search on ξ, the number of integer operations thus totals to $O\Big(k^2 \cdot \|\mathbf{x}\|_\infty \log \|\mathbf{x}\|_\infty\Big)$. We point out, however, that there are families of $k \times k$ matrices A for which the computed vectors $\mathbf{x}$ are such that $\|\mathbf{x}\|_\infty$ is *exponential* in k. Such a family is described in Example 7.1.

Example 5.2 Let G denote the third power of the Shannon cover of the $(1, 7)$-RLL constrained system. The adjacency matrix of G is given by

$$A_G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 2 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} .$$

We apply the Franaszek algorithm to A_G and $n = 4$, with vectors of the form $\boldsymbol{\xi} = \xi \cdot \mathbf{1}$. The smallest value of ξ that results in a nonzero output is $\xi = 3$, in which case the output of the algorithm is

$$\mathbf{x} = (2\ 3\ 3\ 3\ 2\ 2\ 2\ 1)^\top \ .$$

We can now try to find other $(A_G, 4)$-approximate eigenvectors whose largest entry is 3 by applying the Franaszek algorithm with $\boldsymbol{\xi} = \mathbf{x}_i$ for $i = 0, 1, 2, \ldots, 7$, where

$$(\mathbf{x}_i)_u = \begin{cases} (\mathbf{x})_u & \text{if } u \neq i \\ (\mathbf{x})_u - 1 & \text{if } u = i \end{cases} .$$

That is, in each application of the algorithm, we start with a vector $\boldsymbol{\xi}$ obtained by subtracting 1 from one of the entries of $\mathbf{x}$. When doing so, we find that $\boldsymbol{\xi} = \mathbf{x}_7$ is the only case for which the algorithm produces a nonzero output: that output is $\mathbf{x}_7$ itself, namely,

$$(2\ 3\ 3\ 3\ 2\ 2\ 2\ 0)^\top .$$

It follows that there are only two $(A_G, 4)$-approximate eigenvectors, $\mathbf{x}$ and $\mathbf{x}_7$, whose largest entry is 3. □

Example 5.3 Let G be now the second power of the Shannon cover of the $(2, 7)$-RLL constrained system. By applying the Franaszek algorithm we find that $\mathcal{X}(A_G, 2; \xi \cdot \mathbf{1})$ is nonempty if and only if $\xi \geq 4$. The output of the algorithm for $\xi = 4$ is

$$\mathbf{x} = (2\ 3\ 4\ 4\ 3\ 3\ 2\ 1)^\top .$$

There is another $(A_G, 2)$-approximate eigenvector,

$$(2\ 3\ 4\ 4\ 3\ 3\ 1\ 1)^\top ,$$

whose largest entry is 4. □

Recall from Section 4.4 that a necessary and sufficient condition for the existence of a rate $p : q$ deterministic encoder for a constrained system S is the existence of a set of principal states; the definition of such a set (given in Section 4.4) depends on p, q and a deterministic presentation G of S (recall also that this condition was necessary for the existence of a block decodable encoder, in particular the existence of a block encoder). From Proposition 5.3, it is evident that such a set exists if and only if there is an $(A_G^q, 2^p)$-approximate eigenvector with 0–1 entries. Thus, the question of existence of a rate $p : q$ deterministic encoder can be answered by applying the Franaszek Algorithm to the vector $\boldsymbol{\xi} = \mathbf{1}$: such an encoder exists if and only if the algorithm does not return the zero vector.

5.2.3 x-consistent splitting

Let H be a labeled graph and let $\mathbf{x} = (x_v)_{v \in V_H}$ be an (A_H, n)-approximate eigenvector. A *basic* $\mathbf{x}$*-consistent partition* at state u is a partition of E_u into

$$E_u = E_u^{(1)} \cup E_u^{(2)} ,$$

with the property that

$$\sum_{e \in E_u^{(1)}} x_{\tau(e)} \geq n\, y^{(1)} \qquad \text{and} \qquad \sum_{e \in E_u^{(2)}} x_{\tau(e)} \geq n\, y^{(2)} \ ,$$

where $y^{(1)}$ and $y^{(2)}$ are positive integers and

$$y^{(1)} + y^{(2)} = x_u \ .$$

The out-splitting determined by this partition is called a *basic* $\mathbf{x}$*-consistent splitting* at state u, and we denote the resulting labeled graph by H'. It is straightforward to check that the induced vector $\mathbf{x}' = (x'_v)_v$, indexed by the states of H', defined by

$$x'_v = \begin{cases} x_v & \text{if} \quad v \neq u \\ y^{(1)} & \text{if} \quad v = u^{(1)} \\ y^{(2)} & \text{if} \quad v = u^{(2)} \end{cases} \ ,$$

is an $(A_{H'}, n)$-approximate eigenvector.

The cube of the $(0,1)$-RLL graph presentation is shown in Figure 5.4 (which is identical to Figure 2.10). Figure 5.5 shows the result of a basic $\mathbf{x}$-consistent splitting for Figure 5.4, with respect to the $(A_G^3, 2^2)$-approximate eigenvector $\mathbf{x} = (2\ 1)^\top$. State 0 is split into two descendant states, $0^{(1)}$ and $0^{(2)}$, according to the partition $E_0^{(1)} = \{011, 110, 010\}$ and $E_0^{(2)} = \{101, 111\}$. The induced vector is $\mathbf{x}' = (1\ 1\ 1)^\top$, and the resulting labeled graph therefore has minimum out-degree at least $2^2 = 4$.

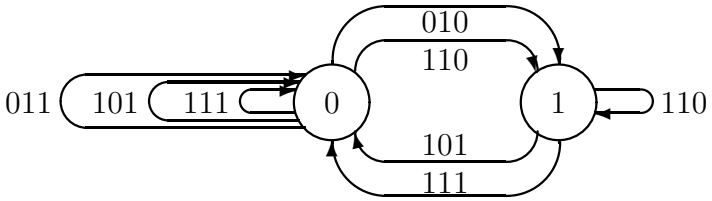


Figure 5.4: Cube of $(0,1)$-RLL graph presentation.

The notion of $\mathbf{x}$-consistency can be extended to out-splittings in general as follows.

Given a labeled graph H, a positive integer n, and an (A_H, n)-approximate eigenvector $\mathbf{x} = (x_v)_{v \in V_H}$, an $\mathbf{x}$*-consistent partition* of H is defined by partitioning the set, E_u, of outgoing edges for each state u in H into $N(u)$ disjoint subsets

$$E_u = E_u^{(1)} \cup E_u^{(2)} \cup \cdots \cup E_u^{(N(u))} \ ,$$

such that

$$\sum_{e \in E_u^{(r)}} x_{\tau(e)} \geq n x_u^{(r)} \qquad \text{for} \qquad r = 1, 2, \ldots, N(u) \ , \tag{5.1}$$

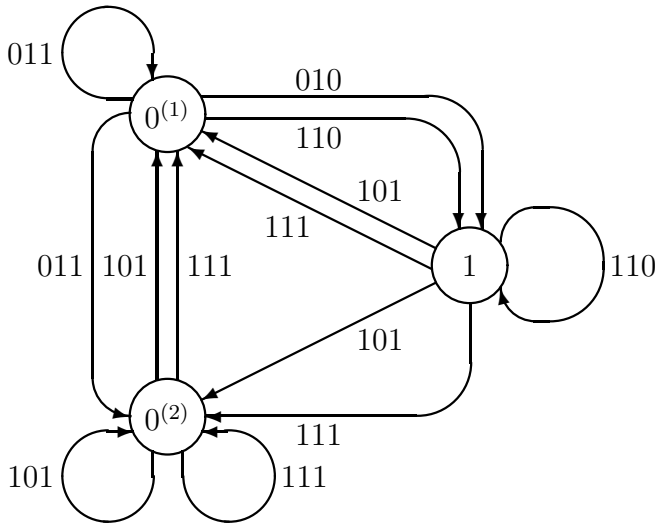


Figure 5.5: Basic **x**-consistent splitting for Figure 5.4.

where $x_u^{(r)}$ are nonnegative integers and

$$\sum_{r=1}^{N(u)} x_u^{(r)} = x_u \quad \text{for every } u \in V_H. \tag{5.2}$$

The out-splitting based upon such a partition is called an **x**-*consistent splitting*. The vector $\mathbf{x}'$ indexed by the states $u^{(r)}$ of the split graph H' and defined by $x'_{u^{(r)}} = x_u^{(r)}$ is called the *induced vector*.

An **x**-consistent partition or splitting is called *non-trivial* if for at least one state u, $N(u) \geq 2$ and $x_u^{(1)}$ and $x_u^{(2)}$ are positive. Observe that any basic **x**-consistent splitting is a non-trivial **x**-consistent splitting.

Figures 5.6 and 5.7 give an example of an **x**-consistent splitting in which two states, 0 and 1, are split simultaneously. An $(A_G, 2)$-approximate eigenvector is $\mathbf{x} = (2\ 2\ 1)^\top$ (in this particular case, it is actually an eigenvector). An **x**-consistent splitting for states 0 and 1 can be carried out as follows. State 0 splits according to the partition $E_0^{(1)} = \{a\}$ and $E_0^{(2)} = \{b, c\}$. State 1 splits, simultaneously, according to the partition $E_1^{(1)} = \{d\}$ and $E_1^{(2)} = \{e\}$. This yields a new labeled graph H' with induced vector $\mathbf{x}' = (1\ 1\ 1\ 1\ 1)^\top$, and the resulting labeled graph therefore has minimum out-degree at least 2.

We summarize in Proposition 5.6 the important features of **x**-consistent out-splittings.

Proposition 5.6 *Let H be a labeled graph and let $\mathbf{x}$ be an (A_H, n)-approximate eigenvector. Suppose that H' is obtained from H by an $\mathbf{x}$-consistent splitting and $\mathbf{x}'$ is the induced vector. Then*

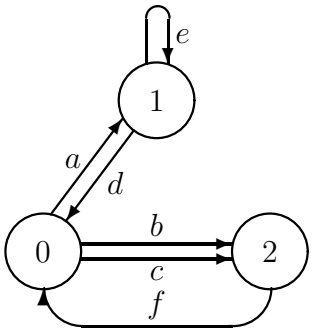


Figure 5.6: Labeled graph to be split.

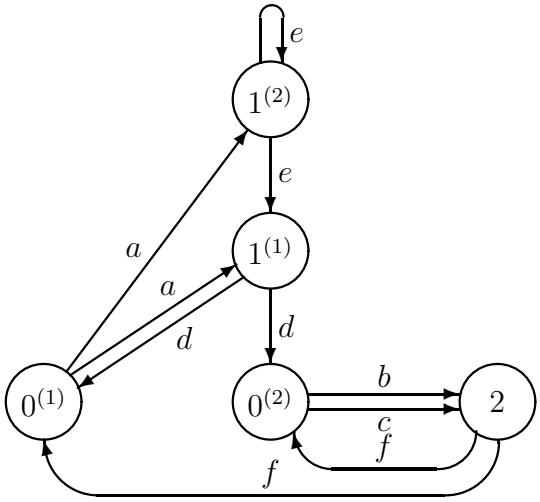


Figure 5.7: **x**-consistent splitting for Figure 5.6.

1. $\mathbf{x}'$ *is an* $(A_{H'}, n)$*-approximate eigenvector.*

2. $\sum_{u \in V_H} x_u = \sum_{v \in V_{H'}} x'_v$.

Proof. 1. The inequality (5.1) says precisely that $(A_{H'}\mathbf{x}')_{u^{(r)}} \geq n\, x'_{u^{(r)}}$ for each state $u^{(r)}$ of H'. So, $\mathbf{x}'$ is an $(A_{H'}, n)$-approximate eigenvector.

2. This follows immediately from (5.2). □

5.3 Constructing the encoder

The key result needed for the construction of our encoders is as follows.

Proposition 5.7 *Let H be an irreducible labeled graph and assume that the all-one vector* $\mathbf{1}$ *is not an (A_H, n)-approximate eigenvector. Let* $\mathbf{x}$ *be a strictly positive (A_H, n)-approximate eigenvector. Then, there is a basic* $\mathbf{x}$*-consistent splitting of H.*

Before giving the proof, we describe how we will make use of it in an iterative fashion to construct finite-state encoders with finite anticipation.

Let G be a deterministic labeled graph presenting S and let p and q be integers such that $p/q \leq \mathsf{cap}(S)$. So, G^q is a deterministic labeled graph presenting S^q. Let $\mathbf{x} = (x_v)_{v \in V_G}$ be an $(A_G^q, 2^p)$-approximate eigenvector (which exists by Theorem 5.4). If $\mathbf{x}$ is a 0–1 vector, then some subgraph of G^q has minimum out-degree at least 2^p, and we are done. So, we may assume that there is no 0–1 $(A_G^q, 2^p)$-approximate eigenvector. Let G' be the labeled subgraph of G^q corresponding to the states of G with nonzero entries of $\mathbf{x}$. The vector $\mathbf{x}'$ obtained by restricting $\mathbf{x}$ to the states in G' is a strictly positive $(A_{G'}, 2^p)$-approximate eigenvector.

If G' is irreducible, then we will be in a position to apply Proposition 5.7 for $H = G'$ and $n = 2^p$. Otherwise, we can restrict to a sink G_0 of G'; recall that a sink is an irreducible component all of whose outgoing edges terminate in the component, and recall that every graph has a sink. Since G_0 is an irreducible component of G', and, by assumption, there is no 0–1 $(A_G^q, 2^p)$-approximate eigenvector, it follows that $\mathbf{1}$ is not an $(A_{G_0}, 2^p)$-approximate eigenvector. Moreover, since G_0 is a sink, it follows that the restriction, $\mathbf{x}_0$, of $\mathbf{x}'$ to G_0 is a strictly positive $(A_{G_0}, 2^p)$-approximate eigenvector. Proposition 5.7 can now be applied to carry out a basic $\mathbf{x}_0$-consistent splitting of G_0, producing an irreducible labeled graph G_1.

By Proposition 5.6, a basic $\mathbf{x}_0$-consistent splitting decomposes an entry of $\mathbf{x}_0$ into strictly smaller positive integers. So, iteration of this state-splitting procedure will produce a sequence of labeled graphs $G_1, G_2, \ldots, G_t$, where the graph G_t has an adjacency matrix A_{G_t} with an all-1's (A_{G_t}, n)-approximate eigenvector. Therefore, by Proposition 5.3, the graph G_t has minimum out-degree at least $n = 2^p$. Since, by Proposition 5.1, out-splitting preserves finite anticipation, the graph G_t has finite anticipation. Deleting excess edges, we pass to a subgraph of G_t' which is an $(S^q, 2^p)$-encoder. Now, tag this encoder with input labels, and we have our rate $p : q$ finite-state encoder for S with finite anticipation.

Having now completely described the construction of the encoder, we have completed the proof of Theorem 4.1 modulo the proof of Proposition 5.7.

Note that the number of iterations required to arrive at the encoder graph is no more than $\sum_{v \in V_G}(x_v - 1)$, since a state v with entry x_v will be split into at most x_v descendant states throughout the whole iteration process. So, by Proposition 5.1, the anticipation of G_t is at most $\sum_{v \in V_G}(x_v - 1)$. For the same reason, the number of states in the encoder graph is at most $\sum_{v \in V_G} x_v$.

If we delete the self-loop at state 1 and assign input tags to the labeled graph in Figure 5.5, we obtain a rate $2 : 3$ finite-state $(0, 1)$-RLL encoder shown in Figure 5.8. We indicate, for

this example, how the encoding and decoding is implemented.

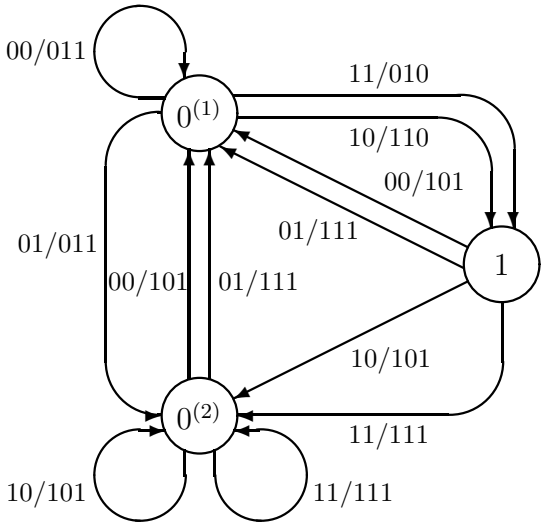


Figure 5.8: Tagged $(0,1)$-RLL encoder.

If we initialize to state $0^{(1)}$, the data sequence of 2-blocks, 00 10 10 11, encodes to the $(0,1)$-RLL sequence of 3-codewords

$$011\ 110\ 101\ 111\ .$$

We decode the $(0,1)$-RLL codeword sequence just generated. Starting at state $0^{(1)}$, the edge determined by the codeword 011, with upcoming codeword 110, is the self-loop $0^{(1)} \stackrel{00/011}{\longrightarrow} 0^{(1)}$, so the decoder will generate the input tag 00. Proceeding, the codeword 110, with upcoming word 101, determines the edge $0^{(1)} \stackrel{10/110}{\longrightarrow} 1$. The reader can decode the next codeword 101 in a similar matter, and that is as far as we can go without knowing more upcoming codewords.

As we will see, the encoder of Figure 5.8 is not the smallest (in terms of number of states) rate $2:3$ finite-state $(0,1)$-RLL encoder. We now proceed with the proof of Proposition 5.7.

Proof of Proposition 5.7. Let $x_{\max} \neq 1$ be the maximum of the entries of $\mathbf{x}$. We will show that there is a state u with the following properties:

$$x_u = x_{\max} \tag{5.3}$$

and

$$(A_H)_{u,v} \neq 0 \quad \text{for some state } v \text{ with } x_v < x_{\max}\ . \tag{5.4}$$

We then show that there is a basic $\mathbf{x}$-consistent splitting at each such state u.

Assume that no such state exists. Then, the outgoing edges for every state u with component $x_{\max}$ must terminate only in states with the component $x_{\max}$. Since the graph H

is assumed to be irreducible, this implies that the approximate eigenvector $\mathbf{x}$ is a constant vector, with all of its components equal to $x_{\max}$. If we divide both sides of the approximate eigenvector inequality ($A_H\mathbf{x} \geq n\mathbf{x}$) by $x_{\max}$, we see that the all-1's vector $\mathbf{1}$ is also an approximate eigenvector. However, this contradicts our assumption about A_H.

Let u be a state satisfying properties (5.3) and (5.4). We claim that $|E_u| \geq n$. To see this, observe that the approximate eigenvector inequality asserts

$$\sum_{v \in V_H} (A_H)_{u,v} x_v \geq n x_u \ .$$

Thus, by property (5.3) above,

$$|E_u|\, x_{\max} \geq \sum_{v \in V_H} (A_H)_{u,v} x_v \geq n x_u \ = \ n x_{\max} \ .$$

Dividing by $x_{\max}$ gives the desired conclusion $|E_u| \geq n$ (actually, with the help of property (5.4) above one can show that $|E_u| \geq n+1$, but this is not really needed now).

Let $M = |E_u| \geq n$. Write $E_u = \{e_1, e_2, \ldots, e_M\}$ and assume that e_1 terminates in state v (i.e., a state satisfying (5.4)), so $x_{\tau(e_1)} < x_{\max}$. Consider the partial accumulated weights

$$\theta_m = \sum_{i=1}^{m} x_{\tau(e_i)} \ , \quad m = 1, 2, \ldots, M$$

and their residues modulo n

$$\rho_m \equiv \theta_m \pmod{n} \ , \quad m = 1, 2, \ldots, M \ .$$

The *pigeon-hole principle*—which states that if one distributes n pigeons into n pigeon-holes, then either every hole has a pigeon, or some hole contains two or more pigeons—implies that the n residues, $\rho_1, \rho_2, \ldots, \rho_n$, satisfy one of the following conditions:

1. $\rho_m \equiv 0 \pmod{n}$ for some $1 \leq m \leq n$, or—
2. $\rho_{m_1} \equiv \rho_{m_2} \pmod{n}$ for some $1 \leq m_1 < m_2 \leq n$.

In the former case, we define a partition of E_u by setting

$$E_u^{(1)} = \{e_i\}_{i=1}^{m} \quad \text{and} \quad E_u^{(2)} = E_u - E_u^{(1)} \ .$$

In the latter case, we set

$$E_u^{(1)} = \{e_i\}_{i=m_1+1}^{m_2} \quad \text{and} \quad E_u^{(2)} = E_u - E_u^{(1)} \ .$$

In either case, the sum of weights of the edges in $E_u^{(1)}$ is divisible by n:

$$\sum_{e \in E_u^{(1)}} x_{\tau(e)} = rn \ .$$

Next we claim that

$$1 \leq r < x_{\max} \ .$$

Clearly $1 \leq r$ since $E_u^{(1)}$ is nonempty. To see that $r < x_{\max}$, observe that in the first case, $E_u^{(1)}$ contains at most n edges and includes e_1 for which $x_{\tau(e_1)} < x_{\max}$; and in the second case, $E_u^{(1)}$ has strictly fewer than n edges, each contributing at most $x_{\max}$ to the sum.

Now,

$$\begin{aligned} \sum_{e \in E_u^{(2)}} x_{\tau(e)} &= \sum_{e \in E_u} x_{\tau(e)} - \sum_{e \in E_u^{(1)}} x_{\tau(e)} \\ &\geq x_u n - rn \\ &= (x_u - r)n \ . \end{aligned}$$

Letting $y^{(1)} = r$ and $y^{(2)} = x_u - r$, we conclude that the partition,

$$E_u = E_u^{(1)} \cup E_u^{(2)} \ ,$$

defines a basic $\mathbf{x}$-consistent splitting. □

The discussion in this chapter implies a encoder construction procedure is known as the *state-splitting algorithm* or the *Adler-Coppersmith-Hassner (ACH) algorithm*. This algorithm is summarized in Figure 5.9.

Hints when constructing an encoder:

- In the course of a sequence of state splittings, the resulting graphs may become too unwieldy to draw. Instead, it may be more convenient to represent the graphs by tables; such a table has rows and columns indexed by the set of states with the (u, v)-entry containing the list of labels of all edges from u to v. Both the untagged encoder and the tagged encoder can also be represented in this way.

- If more then one round of splitting is required, it is convenient at each round to use the notation $u^{i,j}$ for each state. For instance, if state u has weight 5 and is split in the first round into two states, one of weight 3 and the other of weight 2, then after the first round, denote one of the descendant states by $u^{1,3}$ and the other by $u^{4,5}$. After the sequence of splittings is completed, the descendant states of u are denoted u^1, u^2, u^3, u^4, u^5.

1. Select a labeled graph G and integers p and q as follows:
   (a) Find a deterministic labeled graph G (or more generally a labeled graph with finite anticipation) which presents the given constrained system S.
   (b) Find the adjacency matrix A_G of G.
   (c) Compute the capacity $\mathsf{cap}(S) = \log \lambda(A_G)$.
   (d) Select a desired code rate $p : q$ satisfying
   $$\mathsf{cap}(S) \geq \frac{p}{q}$$
   (one usually wants to keep p and q relatively small for complexity reasons).
2. Construct G^q.
3. Using the Franaszek algorithm of Figure 5.3, find an $(A_G^q, 2^p)$-approximate eigenvector $\mathbf{x}$.
4. Eliminate all states u with $x_u = 0$ from G^q, and restrict to an irreducible sink H of the resulting graph. Restrict $\mathbf{x}$ to be indexed by the states of H.
5. Iterate steps 5a–5c below until the labeled graph H has minimum out-degree at least 2^p:
   (a) Find a non-trivial $\mathbf{x}$-consistent partition of the edges in H (the proof of Proposition 5.7 shows how to find such a partition in at least one state).
   (b) Find the $\mathbf{x}$-consistent splitting corresponding to this partition, creating a labeled graph H' and an approximate eigenvector $\mathbf{x}'$.
   (c) Let $H \leftarrow H'$ and $\mathbf{x} \leftarrow \mathbf{x}'$.
6. At each state of H, delete all but 2^p outgoing edges and tag the remaining edges with binary p-blocks, one for each outgoing edge. This gives a rate $p : q$ finite-state encoder for S.

Figure 5.9: State-splitting algorithm.

5.4 Strong decoders

We begin this section by proving Theorem 4.8, thereby showing how to achieve sliding-block decodability for finite-type constraints.

The proof is obtained by applying the state-splitting algorithm to any presentation of S with finite memory. Recall from Propositions 2.7 and 5.1 that higher powers and out-splitting preserve definiteness (although the anticipation may increase under out-splitting). Thus, the $(S^q, 2^p)$-encoder constructed in Section 5.3 is (m, a)-definite for some m and a and so, by Proposition 4.6, is sliding-block decodable. This completes the proof of Theorem 4.8.

Note that we can decode a q-block $\mathbf{w}$ as follows: observe the m previous q-blocks and the a upcoming q-blocks to determine the unique edge that produced $\mathbf{w}$; then read off the input tag on this edge. This defines an (m, a)-sliding-block decoder.

As an example, by examining Figure 5.8, one can see that the $(0,1)$-RLL encoder has a sliding-block decoder with window length 2, as shown in Table 4.1.

Next, we give a very simple example to illustrate how a tagged encoder can be non-catastrophic without being sliding-block decodable.

Consider the constrained system S which is presented by the labeled graph of Figure 5.10. Figure 5.11 exhibits a tagged $(S,3)$-encoder $\mathcal{E}$ which has finite anticipation; in fact it is deterministic. We claim that $\mathcal{E}$ is non-catastrophic. To see this, first observe that every symbol, except for d, is decoded consistently wherever it appears; so, the only decoding ambiguity is caused by the appearance of d as the label of two different edges with different input tags (namely, the edges labeled d outgoing from states 2 and 3). Now, suppose that two right-infinite sequences, $\mathbf{y}^+ = y_1y_2\cdots$ and $\mathbf{z}^+ = z_1z_2\cdots$, that can be generated by right-infinite paths in $\mathcal{E}$, differ in only finitely many places; then for some N and all $i > N$, we have $y_i = z_i$. We must show that when we decode $\mathbf{y}^+$ and $\mathbf{z}^+$ in a state-dependent manner, we get only finitely many differences in their decodings; more precisely, we must show that when $e_1e_2\cdots$ and $e'_1e'_2\cdots$ are right-infinite paths in $\mathcal{E}$ with output labels $\mathbf{y}^+$ and $\mathbf{z}^+$, then their input tags differ in only finitely many places. Well, e_i and e'_i will have the same input tag for $N < i < K$, where K is the first time after N that the symbol d appears in $\mathbf{y}^+$ (equivalently, $\mathbf{z}^+$), i.e., K is the smallest integer such that $K > N$ and $y_K = d$; if d does not appear at all after time N, then e_i and e'_i will have the same input tag for $i > N$, and we will be done. Now, at time K, we may decode $y_K = z_K = d$ in two different ways. But both occurrences of d in Figure 5.11 appear on edges with the same terminal state. So, for $i > K$, we will have $e_i = e'_i$, and decoding will be synchronized. Thus, the number of differences between the decodings of $\mathbf{y}^+$ and $\mathbf{z}^+$ will be at most $N+1$, and therefore our tagged encoder $\mathcal{E}$ is indeed non-catastrophic.

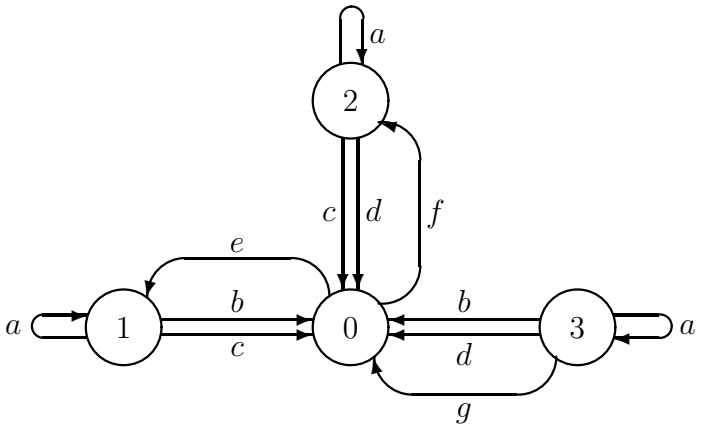


Figure 5.10: Graph presentation of constrained system S.

On the other hand, we claim that $\mathcal{E}$ is not sliding-block decodable. This follows immediately from the fact that for each ℓ, the symbol d in the word $a^\ell dea^\ell$ can appear on either

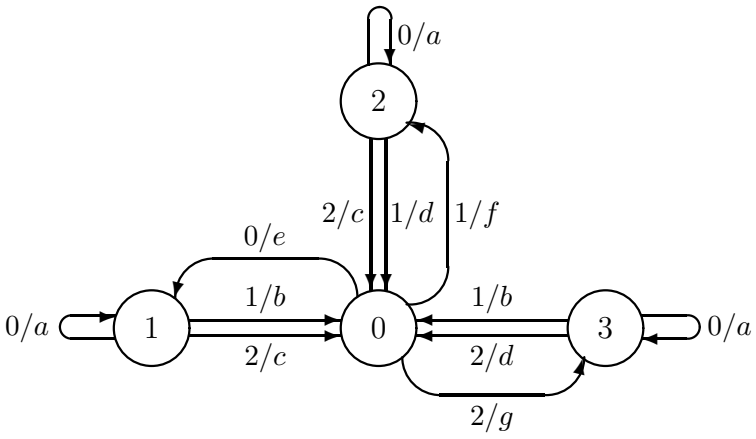


Figure 5.11: $(S, 3)$-encoder for constrained system S presented in Figure 5.10.

edge labeled d, yielding different decodings. In fact, for this particular constraint S, it turns out that there is no sliding-block decodable $(S, 3)$-encoder at all [KarM88].

We remark that if this code were to be used in conjunction with a noisy channel, then an isolated channel error would cause at most two 'bursts' of decoding errors: one burst from the channel error itself followed by a second burst caused by the ambiguity of decoding one occurrence of the symbol d. This 'two-burst' feature holds for the non-catastrophic encoders produced by the construction in Theorem 4.12.

We now give a very rough outline of the proof of Theorem 4.12; for details, see [KarM88]. Let S be an irreducible constrained system and let G be the Shannon cover of S. For simplicity, we assume that $q = 1$ (otherwise, by Theorem 3.7, we take an irreducible constrained system $S' \subseteq S^q$ with $\mathsf{cap}(S') = \mathsf{cap}(S)$). We must show that when $\log n \leq \mathsf{cap}(S)$, there is a non-catastrophic (S, n)-encoder $\mathcal{E}$, and if either $\log n < \mathsf{cap}(S)$ or S is almost-finite-type, then there is a sliding-block decodable (S, n)-encoder.

If $\log n < \mathsf{cap}(S)$, then it follows from Theorem 4.8 and Proposition 3.25 that there is a sliding-block decodable (S, n)-encoder. So, we may suppose that $\mathsf{cap}(S) = \log n$.

We need to introduce some terminology. A *generalized homing word* for a labeled graph is a word $\mathbf{w} = w_1 w_2 \dots w_\ell$ such that for some $1 \leq r \leq \ell$, whenever $e_1 e_2 \dots e_\ell$ and $e'_1 e'_2 \dots e'_\ell$ are paths which generate $\mathbf{w}$, then $\tau(e_r) = \tau(e'_r)$; the integer r is called a *homing coordinate* for $\mathbf{w}$. Observe that in the special case that $r = \ell$, a generalized homing word is a homing word as in Section 2.6.3.

Our tagged encoder $\mathcal{E}$ will satisfy the following property:

(†) There is a generalized homing word $\mathbf{w} \in S(\mathcal{E}) \subseteq S$ for the encoder $\mathcal{E}$ and a positive integer M such that whenever a word $\mathbf{z} = z_{-M} z_{-M+1} \dots z_M$ of length $2M+1$ in $S(\mathcal{E})$

does *not* contain $\mathbf{w}$, and $e_{-M}e_{-M+1}\dots e_M$ and $e'_{-M}e'_{-M+1}\dots e'_M$ are encoder paths which generate $\mathbf{z}$, then e_0 and e'_0 have the same input tag.

Such an encoder is bound to be non-catastrophic for roughly the same reason as the example above (in that example d is the (generalized) homing word): suppose that $\mathbf{y}^+ = y_1y_2\cdots$ and $\mathbf{z}^+ = z_1z_2\cdots$ are two right-infinite sequences that can be generated by the Shannon cover G such that $y_i = z_i$ for all i greater than some N; if we decode $\mathbf{y}^+$ and $\mathbf{z}^+$ then we will obtain the same decoded input tag sequence except possibly for two bounded bursts: a burst from time 0 to time $N+M$ and a burst from time $K-M$ to time $K+r$ where r is a homing coordinate for $\mathbf{w}$ and K is the first time after N at which $\mathbf{w}$ appears in $\mathbf{y}^+$; i.e., K is the smallest integer such that $K > N$ and $y_Ky_{K+1}\dots y_{K+\ell(\mathbf{w})-1} = \mathbf{w}$; after time $K+r$, our decodings of $\mathbf{y}^+$ and $\mathbf{z}^+$ will be synchronized and we will get the same decoded symbol. Thus, our tagged encoder will indeed be non-catastrophic.

How do we construct a tagged encoder which satisfies (†)? Well, recall from Section 2.6 that there is a homing word $\mathbf{w}$ for the Shannon cover G. Let $S_{\mathbf{w}}$ denote the constrained system obtained from S by forbidding the appearance of the word $\mathbf{w}$. Since $\mathsf{cap}(S) = \log n$, it follows that $\mathsf{cap}(S_{\mathbf{w}}) < \log n$. From this, using a state splitting argument, starting with G, and deleting edges, (see [Mar85]), we construct a graph (V, E) endowed with two labelings—input labeling L_I and output labeling L_O—both with finite anticipation, such that

1. the labeled graph (V, E, L_O) presents $S_{\mathbf{w}}$;
2. the labeled graph (V, E, L_I) presents some proper constrained sub-system of the set of all n-ary words; and—
3. whenever γ and γ' are bi-infinite paths in (V, E) with the same L_O-labeling, they have the same L_I-labeling.

Then, by a long and delicate sequence of state splittings, both out-splitting and in-splitting, (see [KarM88]), the labeled graph (V, E, L_O) is transformed and extended to a presentation (V', E', L'_O) of all of S; moreover, this presentation has finite anticipation and constant out-degree n. At the same time, the labeling L_I is transformed to a deterministic labeling which serves as an input tagging L'_I of (V', E', L'_O). This gives our tagged (S, n)-encoder $\mathcal{E}$. While $\mathbf{w}$ need not be a generalized homing word for (V, E, L_O), it does determine a longer word which is a generalized homing word for (V', E', L'_O) and such that the condition (†) holds. This completes our outline of the construction of a non-catastrophic (S, n)-encoder $\mathcal{E}$.

It remains to show that if S is almost-finite-type, then $\mathcal{E}$ is actually sliding-block decodable. By definition, S is presented by some labeled graph with finite anticipation and finite co-anticipation (in fact, by Proposition 2.15, the Shannon cover G is such a presentation). Our encoder $\mathcal{E}$ is constructed from such a presentation by a sequence of state splittings, and thus it too has finite anticipation and finite co-anticipation. To show that it is sliding-block

decodable, it suffices to show that whenever γ and γ' are bi-infinite paths in $\mathcal{E}$ with the same sequence, $\mathbf{y}$, of output labels, then they have the same sequence of input tags.

There are two cases to consider. If $\mathbf{y}$ contains $\mathbf{w}$—the generalized homing word for $\mathcal{E}$—then γ and γ' must arrive at the same state at some time, and so, by finite anticipation and finite co-anticipation, we must have $\gamma = \gamma'$; clearly then γ and γ' have the same sequence of input tags. Otherwise, $\mathbf{y}$ does not contain $\mathbf{w}$ and so by (†), γ and γ' again have the same sequence of input tags; thus, $\mathcal{E}$ is indeed sliding-block decodable, as desired.

5.5 Simplifications

5.5.1 State merging

In practice, it is desirable to design fixed-rate encoders with a small number of states. For a given labeled graph $G = (V, E, L)$, with an (A_G, n)-approximate eigenvector $\mathbf{x} = (x_v)_{v \in V}$, we have shown that the state-splitting algorithm can produce an encoder $\mathcal{E}$ with $|V_{\mathcal{E}}|$ states where

$$|V_{\mathcal{E}}| \leq \sum_{v \in V} x_v \ .$$

This gives an upper bound on the number of states in the smallest (S, n)-encoder. Often, however, one can reduce this number substantially by means of state merging. Although there is not yet a definitive solution to the problem of minimizing the number of encoder states, there are techniques and heuristics that have proved to be very effective in the construction of encoders.

One situation where we can merge states in a labeled graph H is the following. Let u and u' be two states in H and suppose that there is a 1–1 correspondence $e_i \mapsto e'_i$ between the sets of outgoing edges $E_u = \{e_1, e_2, \ldots, e_t\}$ and $E_{u'} = \{e'_1, e'_2, \ldots, e'_t\}$ such that for $i = 1, 2, \ldots, t$ we have $\tau(e_i) = \tau(e'_i)$ and $L(e_i) = L(e'_i)$. Then, we can eliminate one of the states, say u', and all of its outgoing edges, and redirect into u all incoming edges to u'. Clearly, the new labeled graph presents the same constraint, but with one fewer state. Note that this procedure is precisely the inverse of an in-splitting.

Example 5.4 Consider again the $(0, 1)$-RLL constrained system S. If we delete the self-loop $1 \xrightarrow{110} 1$ from Figure 5.5, we can see that states $0^{(2)}$ and 1 can be merged, according to the merging criterion just discussed. The resulting two-state labeled graph is the one shown in Figure 5.12. Tagging the latter, we obtain a tagged $(S^3, 2^2)$-encoder as shown in Figure 5.13; this is the same encoder presented in Figure 4.2. As mentioned in Example 4.2, this encoder is $(0, 1)$-definite. Hence, it is also $(0, 1)$-sliding-block decodable, and the respective decoding table is shown in Table 4.1. Note that the encoder of Figure 5.13 has fewer states than the encoder previously shown in Figure 5.8.

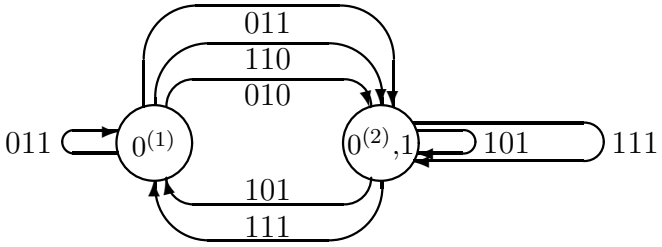


Figure 5.12: Deleting edges and merging states in Figure 5.5.

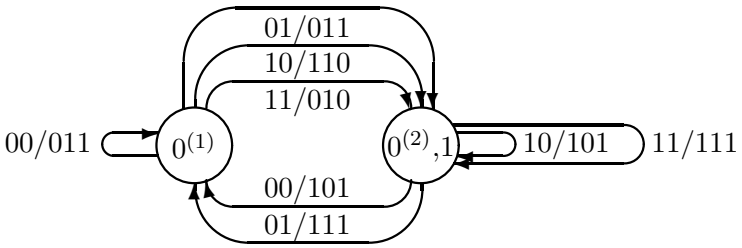


Figure 5.13: Rate $2:3$ tagged two-state encoder for $(0,1)$-RLL constrained system.

Example 5.5 The second power of the Shannon cover of the $(1,3)$-RLL constrained system is given by the graph G in Figure 5.14. One can verify that

$$\mathbf{x} = (x_0\ x_1\ x_2\ x_3)^\top = (1\ 1\ 1\ 0)^\top$$

is an $(A_G^2, 2)$-approximate eigenvector. After deleting state 3 from G, we obtain an $(S(G), 2)$-

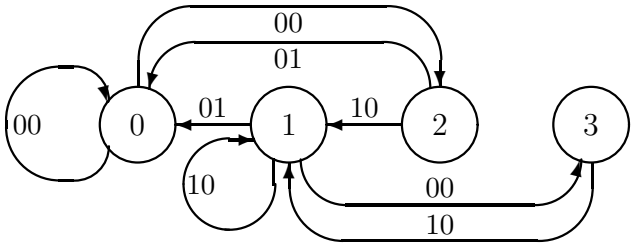


Figure 5.14: Second power of the Shannon cover of the $(1,3)$-RLL constrained system.

encoder in which states 1 and 2 are equivalent. When these two states are merged, we end up with the graph in Figure 5.15. This graph is an untagged version of the MFM encoder in Figure 4.1. □

One key idea for merging states involves ordering the states in a labeled graph to reflect the inclusion relations among the follower sets at each state (defined in Section 2.6): given two states u and u' in a labeled graph G, we say that $u \preceq u'$ if the follower sets $\mathcal{F}_G(u)$

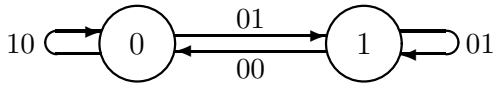


Figure 5.15: Untagged MFM encoder.

and $\mathcal{F}_G(u')$ satisfy $\mathcal{F}_G(u) \subseteq \mathcal{F}_G(u')$. This partial ordering of sets was used by Freiman and Wyner [FW64] in the construction of optimal block codes, mentioned in Section 4.2.

We now generalize the merging operation illustrated above as follows. Let G be a labeled graph and let u and u' be two states in G such that $u \preceq u'$. The (u, u')-*merger of* G is the labeled graph H obtained from G by:

1. eliminating all edges in $E_{u'}$;
2. redirecting into state u all remaining edges coming into state u';
3. eliminating the state u'.

Figures 5.16 and 5.17 show a schematic representation of a (u, u')-merger—before and after merging.

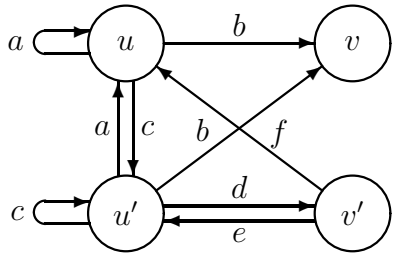


Figure 5.16: Local picture at states u and u' before merging.

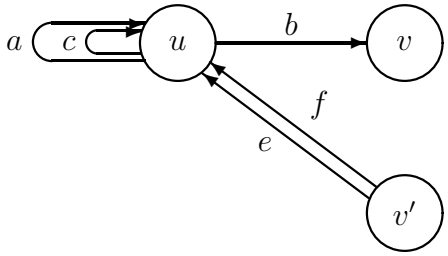


Figure 5.17: (u, u')-merger for Figure 5.16.

It is not hard to see that the merging of states performed on Figure 5.5 to obtain Figure 5.12 is a $(0^{(2)}, 1)$-merger.

The following result shows how we can reduce the final number of encoder states by (u, u')-merging.

Proposition 5.8 *Let G be a labeled graph, with an (A_G, n)-approximate eigenvector $\mathbf{x}$, and let u and u' be states in G satisfying*

(a) $u \preceq u'$, and —

(b) $x_u = x_{u'}$.

Let H denote the (u, u')-merger of G. Then

1. *$S(H) \subseteq S(G)$ and*
2. *The vector $\mathbf{y}$ defined by $y_v = x_v$ for all vertices v of H is an (A_H, n)-approximate eigenvector.*

Proof. 1. Let $\mathbf{w} = w_1 w_2 \dots w_\ell$ be a word generated in H by a path $\gamma = e_1 e_2 \dots e_\ell$. If γ does not contain any edge derived from an edge in G terminating in state u', one can immediately find a corresponding path $\hat{\gamma}$ in G that generates $\mathbf{w}$. Otherwise, let e_t be the last edge of γ that terminates in state u in H and comes from an edge $\hat{e}_t$ in G that terminates in state u'. By hypothesis (a) we have $\mathcal{F}_G(u) \subseteq \mathcal{F}_G(u')$, so there is a path $\hat{e}_{t+1}\hat{e}_{t+2} \dots \hat{e}_\ell$ in G emanating from state u' and generating $w_{t+1}w_{t+2} \dots w_\ell$. If $e_1 e_2 \dots e_{t-1}$ contains no redirected edges, then $e_1 e_2 \dots e_{t-1}\hat{e}_t\hat{e}_{t+1} \dots \hat{e}_\ell$ is a path in G generating $\mathbf{w}$. If it does, let e_k be the last such edge. Then $e_{k+1}e_{k+2} \dots e_{t-1}\hat{e}_t\hat{e}_{t+1} \dots \hat{e}_\ell$ is a path in G that begins at state u and generates $w_{k+1}w_{k+2} \dots w_\ell$. By hypothesis (a), there is another path $e'_{k+1}e'_{k+2} \dots e'_\ell$ in G emanating from u' that also generates $w_{k+1}w_{k+2} \dots w_\ell$. Continuing in this manner, we eventually produce a path in G that generates the entire word $\mathbf{w}$.

2. Let v be a state in H. By hypothesis (b),

$$(A_H \mathbf{y})_v = (A_G \mathbf{x})_v \geq n x_v = n y_v \ ,$$

so $\mathbf{y}$ is an (A_H, n)-approximate eigenvector, as desired. □

So, if states u and u' satisfy hypotheses (a) and (b) of the preceding result, then the number of final encoder states in the state-splitting construction is reduced by x_u.

In a set with partial ordering, there is the possibility of having minimal elements: a state u is *weight-minimal*, with respect to the partial ordering by follower sets and approximate eigenvector $\mathbf{x}$, if, for any other state v, the conditions $v \preceq u$ and $x_v = x_u$ imply that $u = v$.

Proposition 5.8 shows that, by means of preliminary state merging, encoder construction by the state-splitting algorithm can be accomplished using only the subgraph restricted to the weight-minimal states. This reduces the number of final encoder states to the sum of the weights of the weight-minimal states.

Example 5.6 Let G be some power of the Shannon cover of the (d, k)-RLL constrained system. Denoting the states by 0 through k (as in Figure 1.3), it is easy to verify that

$$i + 1 \preceq i$$

for every $d \leq i < k$.

Consider now the special case $(d, k) = (1, 7)$ and let G be the third power of this constrained system. We have mentioned in Example 5.2 that

$$(2\ 3\ 3\ 3\ 2\ 2\ 2\ 0)^\top \tag{5.5}$$

is an $(A_G, 4)$-approximate eigenvector. With respect to this vector, the weight-minimal states in G are 0, 3, 6, and 7. We now delete state 7 (whose weight is zero) and merge the other states into the three remaining weight-minimal states as follows: states 1 and 2 are merged into state 3 to form state 1–3, and states 4 and 5 are merged into state 6 to form state 4–6. This, in turn, yields a graph G' with only three states, as shown in Figure 5.18. The

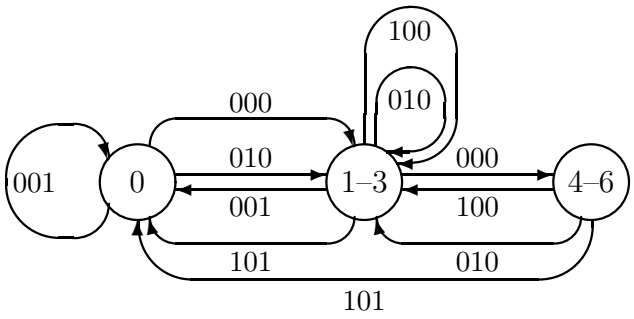


Figure 5.18: Merged graph G' for the $(1, 7)$-RLL constrained system.

adjacency matrix of G' is given by

$$A_{G'} = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 2 & 1 \\ 1 & 2 & 0 \end{pmatrix},$$

with an $(A_{G'}, 4)$-approximate eigenvector $(2\ 3\ 2)^\top$; in fact, this is a true eigenvector of $A_{G'}$ associated with the Perron eigenvalue 4.

If we now apply the state-splitting algorithm to the graph G', we can obtain a rate $2 : 3$ encoder for the $(1, 7)$-RLL constrained system with at most $2 + 3 + 2 = 7$ states. This upper

bound on the number of states is far better than the bound, 17, which we would obtain if the state-splitting algorithm were applied to the original presentation G with the approximate eigenvector in (5.5).

By applying further merging in the course of the state-splitting rounds of G', Weathers and Wolf obtained in [WW91] the encoder of Figure 4.6, which has only four states. It will follow from the discussion in Chapter 7 (Example 7.2) that no rate $2:3$ encoder for the $(1,7)$-RLL constrained system can have less than four states. □

The partial ordering on weight-minimal states also suggests certain out-splitting rounds and further state merging rounds than can simplify the final encoder graph. For instance, if $u \preceq v$ and $x_u < x_v$, it would be tempting to try to split state v into two states $v^{(1)}$ and $v^{(2)}$ with weights x_u and $x_v - x_u$ such that $v^{(1)}$ can be merged with state u. This would then further reduce the number of encoder states by x_u. In most cases of practical interest, this can be done. The paper [MSW92] describes one situation in general and several specific examples where the operations suggested by the partial ordering of the weight-minimal states can actually be implemented (those ideas were used by Weathers and Wolf in [WW91] to obtain their encoder in Figure 4.6). So, the merging principle is a valuable heuristic in encoder design.

However, as we show in Chapter 7, given a constrained system S and a positive integer n, there are lower bounds on the number of states that any (S,n)-encoder can have. In particular, there are limits on the amount of state merging that can be carried out.

Example 5.7 Let G be the 16th power of the $(2,10)$-RLL constrained system. One can verify through the Franaszek algorithm that

$$(1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0)^\top$$

is an $(A_G, 2^8)$-approximate eigenvector. The weight-minimal states in G are 0, 1, 8, and 10, and by deleting states 9 and 10 and merging states 2 through 7 into state 8 we obtain a labeled graph G' with three states, 0, 1, and 2–8. The adjacency matrix of G' is given by

$$A_{G'} = \begin{pmatrix} 83 & 57 & 117 \\ 122 & 83 & 170 \\ 1 & 85 & 173 \end{pmatrix}.$$

By further deleting edges we can obtain the 3-EFM(16) code (see Sections 1.7.2 and 4.4).

In Section 1.7.3, we showed how bit inversions in the output sequence of the encoder can reduce the DSV after precoding; such inversions effectively map certain input tags (bytes) into two possible codewords that differ in one bit. It was also mentioned, however, that the DSV reduction that can be obtained with the 3-EFM(16) code is not enough for optical applications.

The approach adapted in the DVD was designing an $(S(G), n)$-encoder where n is (significantly) greater than 256. The excess out-degree then allows to have $n-256$ input bytes each of which can be mapped into two different codewords; this flexibility in selecting the encoded codeword can then be used to reduce the DSV, especially if the two codeword candidates are such that one contains an even number of 1's while the other contains an odd number.

By Franaszek algorithm we obtain that there exists an (A_G, n)-approximate eigenvector whose largest component is 1 if and only if $n \leq 260$; yet, an out-degree 260 is still too close to 256. Allowing the largest component in the approximate eigenvector to be 2, the values of n can go up to 351. For $n = 351$ we obtain the approximate eigenvector

$$(1\ 1\ 2\ 2\ 2\ 2\ 1\ 1\ 1\ 1\ 0)^\top ,$$

and the respective weight-minimal states in G are 0, 1, 5, 9, and 10. By deleting state 10 and merging states, we obtain a labeled graph H with four states, 0, 1, 2–5, and 6–9, whose adjacency matrix is given by

$$A_H = \begin{pmatrix} 83 & 57 & 98 & 22 \\ 122 & 83 & 142 & 32 \\ 164 & 113 & 192 & 42 \\ 97 & 67 & 113 & 25 \end{pmatrix}$$

and the respective $(A_H, 351)$-approximate eigenvector is

$$(1\ 1\ 2\ 1)^\top .$$

Next, we obtain a labeled graph H' by splitting state 2–5 into two descendant states, $2\text{–}5^{(1)}$ and $2\text{–}5^{(2)}$, with out-degrees 351 and 352. In fact, the splitting can be such that

$$\mathcal{F}_{H'}(2\text{–}5^{(1)}) \subseteq \mathcal{F}_{H'}(6\text{–}9) ,$$

thereby allowing merging. By deleting excess edges we thus obtain a four-state $(S(G), 351)$-encoder that is $(0, 1)$-sliding-block decodable. In particular, we can obtain in this manner the EFMPlus code, which is used in the DVD; see Section 1.7.3 and [Imm95b], [Imm99, Section 14.4.2]. □

5.5.2 Sliding-block decoder window

When a finite-state encoder with sliding block decoder is used in conjunction with a noisy channel, the extent of error propagation is controlled by the size of the decoder window. How large is this window? Well, suppose that we start the state-splitting algorithm with some labeled graph G presenting a constrained system of finite-type and G^q has finite memory $\mathcal{M} = \mathcal{M}(G^q)$ (measured in q-blocks). If t is the number of (rounds of) out-splitting used to

construct the encoder, then the encoder graph $\mathcal{E}$ is $(\mathcal{M}, t)$-definite (measured in q-blocks). It follows that we can design a sliding-block decoder with decoding window length W satisfying the bound

$$\mathsf{W} \leq \mathcal{M} + t + 1$$

(again, measured in q-blocks). Recall from Section 5.3 that an upper bound on the number of (rounds of) out-splitting required is

$$t \leq \sum_{v \in V_G} (x_v - 1) \ ,$$

so,

$$\mathsf{W} \leq \mathcal{M} + \sum_{v \in V_G} (x_v - 1) + 1 \ . \tag{5.6}$$

The guarantee of a sliding-block decoder when S is finite-type and the explicit bound on the decoder window length represent key strengths of the state-splitting algorithm. In practice, however, the upper bound (5.6) on the window length often is larger—sometimes much larger—than the shortest possible.

For the $(0, 1)$-RLL encoder in Figure 5.8, where (referring to Figure 5.4) $\mathcal{M} = 1$, $x_0 = 2$, and $x_2 = 1$, this expression gives an upper bound of 3 (codewords) on the window length. However, we saw, in Table 4.1, a decoder with window length of $\mathsf{W} = 2$. For the rate $2 : 3$ $(1, 7)$-RLL encoder mentioned in Example 5.6 the initial labeled graph has memory $\mathcal{M} = 3$, and the approximate eigenvector is

$$(2\ 3\ 2)^\top \ .$$

The number of rounds of splitting turns out to be only 2, implying

$$\mathsf{W} \leq \mathcal{M} + 3 = 6 \ ,$$

which is, again, less than the upper bound (5.6) gives. In fact, a window length of $\mathsf{W} = 3$ was actually achieved [AHM82], [WW91]. For the rate $8 : 9$ $(0, G/I) = (0, 3/3)$ encoder for PRML discussed in [MSW92], the bound (5.6) was 11, but a window length of $\mathsf{W} = 2$ was achieved [MSW92].

These reduced window lengths were achieved by trying several possibilities for the choices of presentation, approximate eigenvector, out-splittings, elimination of excess edges and input tagging assignment (see, for instance [MSW92], [WW91]). In [KarM88] and [AM95], in-splitting was another tool used in reducing the window length—although for those codes, the ordinary state-splitting algorithm applied to a 'very large' approximate eigenvector will yield codes with the same sliding block decoding window. Recently, Hollmann [Holl95] has found an approach that combines aspects of the state-splitting algorithm with other approaches and has been demonstrated to be of use in further reducing the window length.

To illustrate the importance of the input tagging assignment, consider the encoders in Figures 5.19 and 5.20. In Figure 5.19 (which is the same as the MFM code of Figure 4.1), the

decoder window length is one codeword, but in Figure 5.20, the minimum decoder window length is two codewords (one codeword look-back). The difference is that the assignment in the former is done in a more consistent manner: edges that have the same output label are assigned the same input tag.

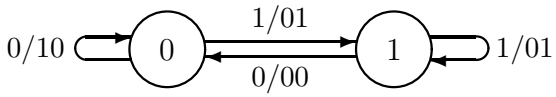


Figure 5.19: One choice of input tags.

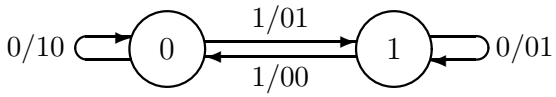


Figure 5.20: Another choice of input tags.

There is a 'brute-force' procedure for deciding if, given m and a, there is an input tagging assignment of a given (S, n)-encoder G which is (m, a)-sliding block decodable: for each edge e in G, let $L(e, \mathsf{m}, \mathsf{a})$ denote the set of all words generated by paths $e_{-\mathsf{m}} \dots e_0 \dots e_{\mathsf{a}}$ such that $e_0 = e$; it is straightforward to verify that an input tagging assignment on G is (m, a)-sliding block decodable if and only if whenever $L(e, \mathsf{m}, \mathsf{a})$ and $L(e', \mathsf{m}, \mathsf{a})$ intersect, e and e' have the same input tag.

In Figure 5.21, we exhibit an $(S, 2)$-encoder which is not block decodable (i.e., $(0, 0)$-sliding block decodable): the reader can easily verify that it is impossible to assign 1-bit input tags in such a way that the sliding-block decoder will have no look-back and no look-ahead. However, since the encoder is $(1, 0)$-definite, any input tag assignment allows a decoder window of length 2.

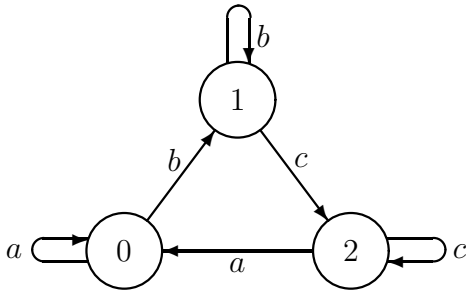


Figure 5.21: Encoder requiring decoder window length ≥ 2.

Theorem 4.9 shows that in general there is not much hope for simplifying the brute-force procedure, described above, when $n \geq 3$ (see the paragraph following the statement of Theorem 4.9).

5.6 Universality of the state-splitting algorithm

We end our treatment of the state-splitting algorithm by reviewing, without proof, some results from [AM95] and [RuR01] on the universality of the state-splitting algorithm.

Given a deterministic graph G, an integer n, and an (A_G, n)-approximate eigenvector $\mathbf{x}$, we say that the triple $(G, n, \mathbf{x})$ *splits into an* (S, n)*-encoder* $\mathcal{E}$ if there is sequence of rounds of out-splitting that can be applied to G such that the following holds:

1. The first round is consistent with $\mathbf{x}$, and each subsequent round is consistent with the respective induced approximate eigenvector.
2. The last round ends produces a labeled graph H in which each state has out-degree at least n.
3. The encoder $\mathcal{E}$ can be obtained from H by deleting excess edges and—in case $\mathcal{E}$ is tagged—by assigning input tags to the edges.

5.6.1 Universality for sliding-block decodable encoders

For integers m, a, and a function $\mathcal{D}$ from $(\mathsf{m}+\mathsf{a}+1)$-blocks of S to the n-ary alphabet (such as a sliding-block decoder), we define $\mathcal{D}_\infty^{\mathsf{m},\mathsf{a}}$ to be the induced mapping on bi-infinite sequences defined by:

$$\mathcal{D}_\infty^{\mathsf{m},\mathsf{a}}(\cdots w_{-1}w_0w_1\cdots) = \cdots s_{-1}s_0s_1\cdots \ ,$$

where

$$s_i = \mathcal{D}(w_{i-\mathsf{m}}\ldots w_{i-1}w_iw_{i+1}\ldots w_{i+\mathsf{a}}) \ .$$

Sometimes, we simply write $\mathcal{D}_\infty = \mathcal{D}_\infty^{\mathsf{m},\mathsf{a}}$. For a tagged (S, n)-encoder $\mathcal{E}$ with sliding-block decoder $\mathcal{D}$, we then have the mapping $\mathcal{D}_\infty$, and we take its domain to be the set of all bi-infinite (output) symbol sequences obtained from $\mathcal{E}$. When we refer to $\mathcal{D}_\infty$, we tacitly assume that its domain is included. We say that a mapping $\mathcal{D}_\infty$ is a *sliding-block* (S, n)*-decoder* if $\mathcal{D}$ is a sliding-block decoder for some tagged (S, n)-encoder.

We have the following positive results from [AM95].

Theorem 5.9 *Let S be an irreducible constrained system and let n be a positive integer.*

(a) Every sliding-block (S, n)-decoder has a unique minimal tagged (S, n)-encoder (here minimality can be taken to be in terms of number of states).

(b) If we allow an arbitrary choice of deterministic presentation G of S and (A_G, n)-approximate eigenvector $\mathbf{x}$, then the triple $(G, n, \mathbf{x})$ splits into a tagged (S, n)-encoder for every sliding-block (S, n)-decoder. If we also allow merging of states (i.e., (u, v)-merging

as in Section 5.5.1), then that triple splits into the minimal tagged (S, n)-encoder for every sliding-block (S, n)-decoder.

(c) If we fix G to be the Shannon cover S, but allow arbitrary (A_G, n)-approximate eigenvector $\mathbf{x}$, then $(G, n, \mathbf{x})$ splits into a tagged (S, n)-encoder for every sliding-block (S, n)-decoder $\mathcal{D}$, modulo a change in the domain of $\mathcal{D}_\infty$, possibly with a constant shift of each bi-infinite sequence prior to applying $\mathcal{D}_\infty$ (but with no change in the decoding function $\mathcal{D}$ itself). If we also allow merging of states, then, modulo the same changes, the triple $(G, n, \mathbf{x})$ splits into the minimal tagged (S, n)-encoder for every sliding-block (S, n)-decoder. In particular, the triple splits into a sliding-block (S, n)-decoder with minimal decoding window length.

On the other hand, we have the following negative results from [AM95].

1. If we fix G to be the Shannon cover of an irreducible constrained system S, then $(G, n, \mathbf{x})$ need not split into a sliding-block (S, n)-decoder with smallest number of encoder states in its minimal tagged (S, n)-encoder.

2. If we fix G to be the Shannon cover of an irreducible constrained system S and we fix $\mathbf{x}$ to be a minimal (A_G, n)-approximate eigenvector (in terms of the value of $\|\mathbf{x}\|_\infty$), then $(G, n, \mathbf{x})$ may fail to split into a sliding-block (S, n)-decoder with minimum decoding window length; examples of this kind were first found by Kamabe [Kam89] and Immink [Imm92], but in [AM95] an example is given where $\mathsf{cap}(S) = \log n$.

5.6.2 Universality for encoders with finite anticipation

Let u and u' be states in labeled graph G. We say that u and u' are 0-*strongly equivalent* if they are follower-set equivalent states; namely, $\mathcal{F}_G(u) = \mathcal{F}_G(u')$.

States u and u' in $G = (V, E, L)$ are t-*strongly equivalent* if the following conditions hold:

1. A one-to-one and onto mapping $\varphi : E_u \to E_{u'}$ can be defined from the set of outgoing edges of u to the set of outgoing edges of u', such that for every $e \in E_u$, both e and $\varphi(e)$ have the same label.

2. For every $e \in E_u$, the terminal states of e and $\varphi(e)$ are $(t-1)$-strongly equivalent.

States that are t-strongly equivalent are also r-strongly equivalent (and in particular they are equivalent states) for every $r < t$.

We say that two states are *strongly equivalent states* if for every $t \geq 0$ the states are t-strongly equivalent. So, when states are strongly equivalent, the infinite trees of paths that

start in those states are the same. In a deterministic graph, two states are equivalent if and only if they are strongly equivalent. On the other hand, in a nondeterministic graph there may be two states that are equivalent but not strongly equivalent.

The following result is proved in [Ru96] and [RuR01].

Theorem 5.10 *Let S be an irreducible constraint and let n be a positive integer where $\mathsf{cap}(S) \geq \log n$. Suppose there exists some irreducible (S,n)-encoder $\mathcal{E}$ with $\mathcal{A}(\mathcal{E}) = t < \infty$. Then there exists an irreducible deterministic (not necessarily reduced) presentation G of S and an (A_G, n)-approximate eigenvector $\mathbf{x}$ that satisfy the following:*

(a) $\|\mathbf{x}\|_\infty \leq n^t$.

(b) The triple $(G, n, \mathbf{x})$ splits in t rounds into an (S,n)-encoder $\mathcal{E}_G$ such that $\mathcal{A}(\mathcal{E}_G) = t$.

(c) In each of the splitting rounds, every state is split into at most n states.

(d) In the ith round, the induced approximate eigenvector $\mathbf{x}^{(i)}$ satisfies $\|\mathbf{x}^{(i)}\|_\infty \leq n^{t-i}$.

(e) The encoder $\mathcal{E}$ can be obtained from $\mathcal{E}_G$ by merging strongly equivalent states.

Theorem 5.10 establishes the universality of the state-splitting algorithm for encoders with finite anticipation: every (S,n)-encoder with finite anticipation can be constructed using the state-splitting algorithm, combined with merging of (strongly) equivalent states, where the input to the process is some irreducible deterministic presentation G of S and an (A_G, n)-approximate eigenvector $\mathbf{x}$. (However, as shown in [RuR01], not always can the Shannon cover be taken as the graph G in Theorem 5.10.)

Another application of Theorem 5.10 is obtaining lower bounds on the anticipation of any (S,n)-encoder. We elaborate more on this in Section 7.4.3.

Problems

Problem 5.1 Construct the graph that is obtained by a complete out-splitting of the Shannon cover of the $(1,3)$-RLL constrained system in Figure 1.16.

Problem 5.2 Let $\mathcal{E}_1$ be an (S,n)-encoder with finite anticipation. What can be said about the graph $\mathcal{E}_2$ that is obtained from $\mathcal{E}_1$ by complete out-splitting?

Problem 5.3 Let $G = (V, E, L)$ be an irreducible graph with $\lambda = \lambda(A_G)$ and let n be a positive integer such that $n < \lambda$. Denote by Δ the largest out-degree of any state in G.

Let $\mathbf{x} = (x_v)_{v \in V}$ be a strictly positive right eigenvector of A_G associated with the eigenvalue λ and let $x_{\min}$ the smallest entry in $\mathbf{x}$. Normalize the vector $\mathbf{x}$ so that $x_{\min} = \Delta/(\lambda - n)$, and define

the integer vector $\mathbf{z} = (z_v)_{v \in V}$ by $z_v = \lfloor x_v \rfloor$, where $\lfloor y \rfloor$ stands for the largest integer not greater than y.

1. Show that
$$A_G \mathbf{z} \geq n\mathbf{z} > \mathbf{0}$$
(that is, $\mathbf{z}$ is an (A_G, n)-approximate eigenvector whose entries of are all positive).

2. Show that the sum of the entries in $\mathbf{z}$ satisfies the inequality
$$\sum_{v \in V} z_v \leq \frac{\Delta}{\lambda - n} \cdot \frac{\lambda^{|V|} - 1}{\lambda - 1} .$$

Problem 5.4 Let S be the $(0, \infty, 2)$-RLL constraint; that is, S consists of all binary words in which the runs of 0's in between consecutive 1's have even length. Denote by G the Shannon cover of S.

1. Construct the graphs G and G^3.
2. Are there any values m and a for which G^3 is (m, a)-definite?
3. Compute the base-2 capacity of S.
4. Construct a $(0, 1)$-definite $(S^3, 4)$-encoder with three states.
5. Does there exist an $(S^3, 4)$-encoder that is block decodable?

Problem 5.5 Let S be the constrained system presented by the graph G in Figure 4.11.

1. Construct an $(S^2, 2^3)$-encoder (i.e., a rate $3:2$ finite-state encoder for S) with two states.
2. Assign input tags to the encoder found in 1 so that it is block decodable (i.e., $(0, 0)$-sliding block decodable).

Problem 5.6 Let S be the constrained system presented by the graph G in Figure 5.22.

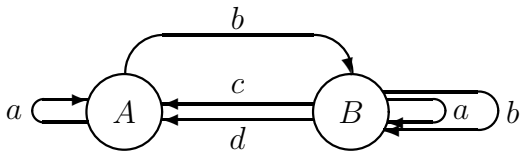


Figure 5.22: Graph G for Problem 5.6.

1. Compute the eigenvalues of A_G.

2. Construct a three-state (S, n)-encoder with $n = \lambda(A_G)$ by applying the state-splitting algorithm to G.

3. What is the anticipation of the encoder found in 2?

4. Let H be the graph in Figure 5.23. Is H an (S, n)-encoder?

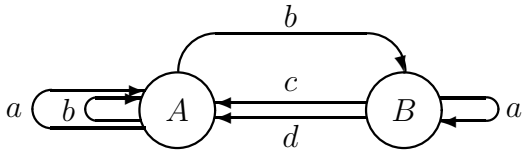


Figure 5.23: Graph H for Problem 5.6.

5. What is the anticipation of the graph H?

6. Can H be obtained by applying a sequence of rounds of state splitting to G (and merging equivalent states if there are any)? If yes, specify the sequence of state-splitting rounds; otherwise, explain.

Problem 5.7 Let S be the constrained system generated by the graph G in Figure 2.22.

1. Let $\mathcal{E}$ be an $(S, 3)$-encoder with the smallest number of states that can be obtained by an application of the state-splitting algorithm (without merging equivalent states) to the graph G. What is the number of states in $\mathcal{E}$?

2. Repeat 1 except that now $\mathcal{E}$ is an $(S^2, 2^3)$-encoder with the smallest number of states obtained by an application of the state-splitting algorithm (without merging equivalent states) on G^2.

3. Repeat 1 except that now $\mathcal{E}$ is an $(S^4, 2^6)$-encoder obtained from G^4.

4. What can be said about the anticipation of the encoder in 3?

Problem 5.8 Let G be the graph in Figure 5.24 in which the edges are assumed to have distinct labels.

1. Compute an $(A_G, 4)$-approximate eigenvector $\mathbf{x}$ in which the largest entry is the smallest possible.

2. Apply the state-splitting algorithm to G and the vector $\mathbf{x}$ from 1, and construct an $(S(G), 4)$-encoder using a minimal number of state-splitting rounds. What is the adjacency matrix of the resulting encoder?

3. Show that the encoder found in 2 satisfies the minimality criterion of number of rounds; that is, explain why no other $(S(G), 4)$-encoder can be obtained from $\mathbf{x}$ by less state-splitting rounds.

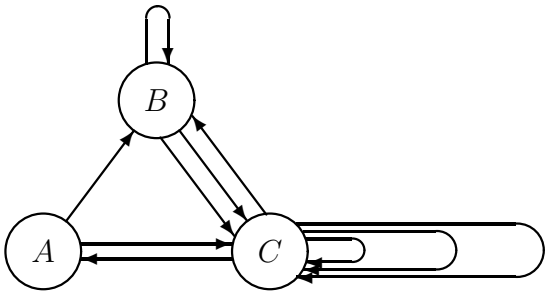


Figure 5.24: Graph G for Problem 5.8.

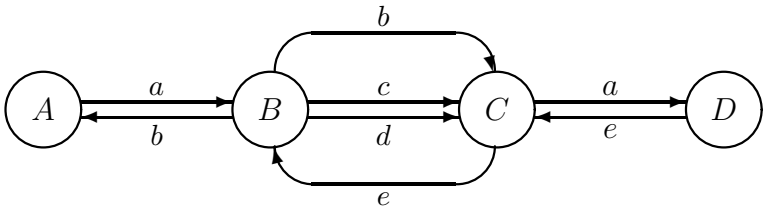


Figure 5.25: Graph G for Problem 5.9.

Problem 5.9 Let G be the graph in Figure 5.25 with labels over $\{a, b, c, d\}$.

1. What is the anticipation of G?
2. Compute the base-2 capacity of $S(G)$.
3. Compute an $(A_G, 2)$-approximate eigenvector in which the largest entry is the smallest possible.
4. Apply the state-splitting algorithm to G and the vector found in 3, and construct an $(S(G), 2)$-encoder with anticipation which is the smallest possible.
5. The anticipation of the encoder found in 4 is smaller than the anticipation of G. Explain how this could happen, in spite of obtaining the encoder by splitting states in G.

Problem 5.10 Let G be a deterministic graph and let $\mathbf{x}$ be an (A_G, n)-approximate eigenvector. Further, assume that an $(S(G), n)$-encoder $\mathcal{E}$ can be obtained from G by *one* $\mathbf{x}$-consistent round of state splitting (and deleting redundant edges), and the resulting $(A_{\mathcal{E}}, n)$-approximate eigenvector is a 0–1 vector.

1. Show that $\mathbf{x}$ must satisfy the inequality

$$A_G \mathbf{1} \geq \mathbf{x} \, ,$$

where $\mathbf{1}$ is the all-one vector.

2. Suppose that, in addition, $\mathbf{x}$ does *not* satisfy the inequality

$$A_G\mathbf{1} \geq 2\mathbf{x} \, .$$

Show that one of the entries in $\mathbf{x}$ is at least n.

Hint: Show that there is a state u in G such that every $\mathbf{x}$-consistent partition of the outgoing edges from u must contain a partition that consists of exactly one edge.

Problem 5.11 Let S be the constrained system presented by the graph G in Figure 2.23.

1. Construct an $(S^4, 2^3)$-encoder (i.e., a rate $3:4$ finite-state encoder for S) with two states.
2. Is it possible to assign input tags to the encoder found in 1 so that it is block decodable (i.e., $(0,0)$-sliding block decodable)? If yes, suggest such a tag assignment; otherwise, explain.

Problem 5.12 Let S be the constrained system generated by the graph G in Figure 5.26.

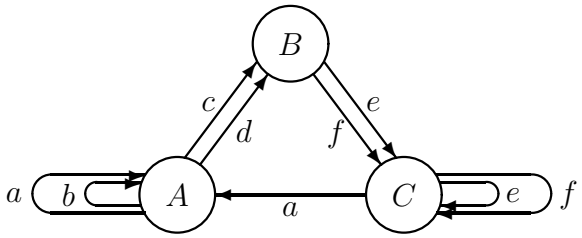


Figure 5.26: Graph G for Problem 5.12.

1. Find the smallest value of M for which there exists an $(A_G, 3)$-approximate eigenvector whose largest entry equals M.
2. Among all $(A_G, 3)$-approximate eigenvectors whose largest entries equal the value M found in 1, find a vector $\mathbf{x}$ whose sum of entries is minimal.
3. Construct an $(S, 3)$-encoder $\mathcal{E}$ by applying the state-splitting algorithm to G and the vector $\mathbf{x}$ found in 2.
4. How many state-splitting rounds were required in 3? Why can't an $(S, 3)$-encoder be obtained from G and $\mathbf{x}$ using a smaller number of rounds?
5. What is the anticipation of $\mathcal{E}$?
6. An (m, a)-sliding block decoder is sought for the encoder $\mathcal{E}$. Find the smallest possible values of m and a for which such a decoder exists regardless of the tag assignment to the edges in $\mathcal{E}$.

7. Is it possible to have an (m, a)-sliding block decoder for $\mathcal{E}$ with a smaller m by a clever tag assignment to the edges in $\mathcal{E}$? If yes, suggest such a tag assignment. Otherwise explain.

8. Repeat 7 with respect to the parameter a.

Problem 5.13 Let S be the constrained system presented by the graph G in Figure 5.27 (which is the same as Figure 2.25).

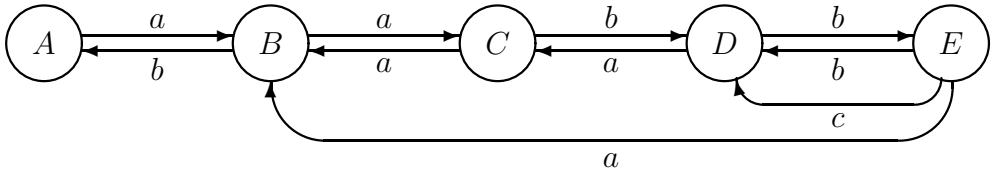


Figure 5.27: Graph G for Problem 5.13.

1. Compute the base-2 capacity of S.

2. Compute an $(A_G, 2)$-approximate eigenvector in which the largest entry is the smallest possible.

3. Show that an $(S, 2)$-encoder can be obtained by four rounds of splitting of G.

4. Construct a $(0, 1)$-definite $(S^2, 4)$-encoder with two states.

Problem 5.14 Let S be the constrained system presented by the graph G in Figure 2.24.

1. Compute the base-2 capacity of S.

2. Using the state-splitting algorithm, construct an $(S^2, 2)$-encoder (i.e., a rate $1 : 2$ finite-state encoder for S) with six states and anticipation at most 1.

3. Is it possible to assign input tags to the encoder in 2 so that it is block decodable? If yes, suggest such an assignment; otherwise, explain.

4. Show how to construct an $(S^2, 2)$-encoder with finite anticipation yet that anticipation is greater than 1.

5. Is there a positive integer ℓ for which there exists a block $(S^{2\ell}, 2^{\ell})$-encoder (i.e., a rate $\ell : 2\ell$ one-state encoder for S)? If yes, construct such an encoder; otherwise, explain.

Problem 5.15 Let S be the constrained system generated by the graph G in Figure 5.28.

1. What is the memory of G?

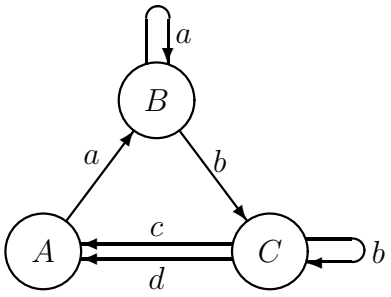


Figure 5.28: Graph G for Problem 5.15.

2. Compute the base-2 capacity of $S(G)$.

3. Compute an $(A_G, 2)$-approximate eigenvector $\mathbf{x}$ in which the largest entry is the smallest possible.

4. Apply the state-splitting algorithm to G and the vector $\mathbf{x}$ from 3, and construct an $(S(G), 2)$-encoder using a minimal number of state-splitting rounds. Merge states to obtain an encoder with three states. What is the anticipation of this encoder?

5. Assign input tags to the encoder found in 4 so that it is (m, a)-sliding-block decodable with the smallest possible window length $\mathsf{m} + \mathsf{a} + 1$.

6. Is there a positive integer ℓ for which there exists a deterministic $(S^\ell, 2^\ell)$-encoder? If yes, construct such an encoder; otherwise, explain.

Problem 5.16 Let G be the second power of the Shannon cover of the $(2, 7)$-RLL constrained system and let $bldx$ be the vector

$$(2\ 3\ 4\ 4\ 3\ 3\ 1\ 1)^\top .$$

As mentioned in Example 5.3, this vector is an $(A_G, 2)$-approximate eigenvector.

1. Find the weight-minimal states in G with respect to $\mathbf{x}$.

2. By merging states in G into its weight-minimal states, construct a deterministic graph G' with five states such that $S(G') \subseteq S(G)$ and $\mathsf{cap}(S(G')) \geq 1$.

Chapter 6

Other Code Construction Methods

In Chapter 5, we focused on the state-splitting algorithm which gives a rigorous proof of the general existence theorems (Theorems 4.1 and 4.8) for constrained code constructions. The algorithm has other virtues: it is fairly easy to use, it has yielded many codes of practical interest, and, as stated in Section 5.6, it is a universal method of constructing every sliding-block decoder. However, as pointed out in Section 5.5, there are lots of choices in the algorithm: the presentation, the approximate eigenvector, the sequence of out-splittings, the deletion of excess edges, and the input-tagging assignment. It is not known how to make these choices to yield optimal encoders/decoders.

There have been many other important and interesting approaches to constrained code construction—far too many to mention here. In this chapter we review, without proof, some alternative methods for constructing constrained codes (see also Sections 4.2 and 4.4). Since these methods are all aimed at the same goal, it is perhaps not surprising that they have a lot in common. In fact, other methods are probably just as universal as the state-splitting algorithm. Some of these methods give very useful heuristics for constructing sliding-block decodable encoders with small decoding window length as well as some rigorous upper bounds on the smallest possible anticipation (or decoding delay) and decoding window length.

We will focus on constructions of tagged (S, n)-encoders. So, throughout this chapter, we will always assume that S is an irreducible constrained system with $\mathsf{cap}(S) \geq \log n$. For rate $p : q$ finite-state encoders for S, one can apply the results and methods of this chapter by passing to the power S^q.

6.1 IP encoders

We start with an encoder construction due to Franaszek that is very closely related to the state-splitting construction. We need the following notation: for a path γ in a labeled graph

$G = (V, E, L)$, let $\hat{\gamma}$ denote the truncation of γ obtained by deleting the last edge, and for a set Γ of paths in G, let $\hat{\Gamma}$ denote the set $\{\hat{\gamma} : \gamma \in \Gamma\}$.

Now, for each state $u \in V$, let $N = N(u)$ be a positive integer and let $\Gamma_u = \{\Gamma_u^{(1)}, \Gamma_u^{(2)}, \ldots, \Gamma_u^{(N)}\}$ be a partition of a set of paths outgoing from u, of some fixed length T, subject to the following condition:

IP condition: Whenever e is an edge from state u to state v and γ_1 and γ_2 are paths that belong to the same element of Γ_v (in particular, they both have initial state v), then $e\hat{\gamma}_1$ and $e\hat{\gamma}_2$ belong to the same element of Γ_u.

Note that this is a condition imposed on the entire collection of partitions, $\{\Gamma_u\}_{u \in V}$, not on an individual partition Γ_u. The IP condition was developed by Franaszek [Fra80a], [Fra80b], [Fra82], who called the individual partition elements, *independent path sets* or *IP sets*. The IP condition was called the *left Markov property* in [AM95].

Now suppose that $\{\Gamma_u\}_{u \in V}$ is a collection of partitions satisfying the IP condition. Construct the following labeled graph $G' = (V', E', L')$: the states in G' are the independent path sets $\Gamma_u^{(i)}$; there is an edge $\Gamma_u^{(i)} \xrightarrow{b} \Gamma_v^{(j)}$ in G' for each edge e labeled b from state u to state v in G such that

$$e\,\widehat{\Gamma}_v^{(j)} \subseteq \Gamma_u^{(i)} \ .$$

Now, it can be shown that G' is lossless (Problem 6.1). Moreover, if the vector $\mathbf{x}$ defined by $x_u = N(u)$ is an (A_G, n)-approximate eigenvector, then G' has minimum out-degree at least n; so, by deleting edges and adding input tags, we obtain an ordinary tagged (S, n)-encoder, which we call an *IP encoder*. If G has finite memory $\mathcal{M}$, then this encoder is sliding-block decodable with decoding window length $\leq \mathcal{M}+T+1$.

It turns out that any IP encoder can also be constructed by $\mathbf{x}$-consistent out-splittings (see [AM95] and [Holl95]). It follows from this result and the state-splitting construction in Chapter 5 that the IP approach is bound to succeed provided that T is taken sufficiently large. In fact, one can view the state-splitting algorithm as giving a constructive procedure for manufacturing IP sets. Recently, Franaszek and Thomas [FT93] discovered an algorithm which reduces the search required for finding IP sets.

6.2 Stethering encoders

Next, we describe a more general framework of encoder construction within which the state-splitting construction fits. Let $G = (V, E, L)$ be a deterministic labeled graph with $S = S(G)$ and let $\mathbf{x} = (x_u)_{u \in V}$ be an (A_G, n)-approximate eigenvector. As usual, we may assume that

the entries of $\mathbf{x}$ are all strictly positive. We describe a particular class of (S,n)-encoders $\mathcal{E}$, built from $\mathbf{x}$, as follows. The set of states of $\mathcal{E}$ is given by

$$V_{\mathcal{E}} = \Big\{ (u,i) \;\Big|\; u \in V_G \quad \text{and} \quad i = 0,1,\ldots,x_u{-}1 \Big\} .$$

Recall that E_u denotes the set of edges outgoing from a state u, and we will use $L(E_u)$ here to denote the set of L-labels of edges in E_u. Since G is deterministic, the mapping $E_u \to L(E_u)$ defined by $e \mapsto L(e)$ is one-to-one and onto. So, it makes sense, for each $u \in V_G$ and $a \in L(E_u)$, to define $\tau(u;a)$ to be the terminal state of the unique edge outgoing from u with label a. Now, let

$$\Delta_u = \{ (a,j) \,:\, a \in L(E_u) \quad \text{and} \quad j = 0,1,\ldots,x_{\tau(u;a)}{-}1 \} .$$

By definition of approximate eigenvector we have $|\Delta_u| \geq n x_u$. Thus, we can partition Δ_u into $x_u{+}1$ subsets $\Delta_u^{(0)}, \Delta_u^{(1)}, \ldots, \Delta_u^{(x_u)}$ such that $|\Delta_u^{(i)}| = n$ for each i, except for $i = x_u$ ($\Delta_u^{(x_u)}$ may be empty). Now, for each $i = 0,1,\ldots,x_u{-}1$ and $(a,j) \in \Delta_u^{(i)}$, we endow $\mathcal{E}$ with an edge $(u,i) \stackrel{a}{\to} (\tau(u;a), j)$. This completely defines $\mathcal{E}$.

It turns out that $\mathcal{E}$ is an (S,n)-encoder (the proof is essentially contained in [AGW77]—see also [AMR95]); in fact, if G were merely lossless, then one could modify the construction so that $\mathcal{E}$ would still be an (S,n)-encoder. Also, it is not hard to see that the (S,n)-encoders constructed by the state-splitting algorithm are of this type. Clearly the construction of this type of encoder is somewhat easier than the state-splitting construction: there is no iterative process to go through. However, there are lots of examples of encoders of this type that do not have finite anticipation; it is not at all clear how to choose the partitions of each Δ_u to achieve finite anticipation.

On the other hand, if we have sufficient excess capacity and we choose our partitions of Δ_u with some extra care, then it turns out that $\mathcal{E}$ will have finite anticipation; and if G has finite memory (more generally, if G is definite), then $\mathcal{E}$ will be definite, and so any tagging of $\mathcal{E}$ will yield a sliding-block decodable encoder. We describe this special construction as follows.

The definition of the partitions assumes some ordering on the edges in E_u for each $u \in V_G$—equivalently, in light of the assumption that G is deterministic, an ordering on the symbols in $L(E_u)$. We allow symbols to have different ordering relations in $L(E_u)$ for different states u. For $u \in V_G$ and $a \in L(E_u)$, define $\phi(u;a)$ by

$$\phi(u;a) = \sum_{\{b \in L(E_u) \,:\, b < a\}} x_{\tau(u;b)} ,$$

where the sum is zero on an empty set. For $i = 0,1,\ldots,x_u{-}1$, let

$$\Delta_u^{(i)} = \{ \, (a,j) \,:\, a \in L(E_u) \quad \text{and} \quad in \leq \phi(u;a) + j < (i+1)n \, \} ,$$

and $\Delta_u^{(x_u)}$ is whatever is left over. This means that for each $u, v \in V_G$, $0 \leq i < x_u$, $0 \leq j < x_v$, and symbol a, we have one edge $(u,i) \stackrel{a}{\to} (v,j)$ in $\mathcal{E}$ if and only if

$$a \in L(E_u) , \quad v = \tau(u;a) , \quad \text{and} \quad in \leq \phi(u;a) + j < (i+1)n . \tag{6.1}$$

Such an encoder is called a *stethering* (S, n)*-encoder* because the elements of each partition element $\Delta_u^{(i)}$ form a contiguous interval and thus are 'stuck together.' This construction is illustrated in Figure 6.1: there is an edge from each state (u, i) in the first row to each state in the second row that sits below (u, i).

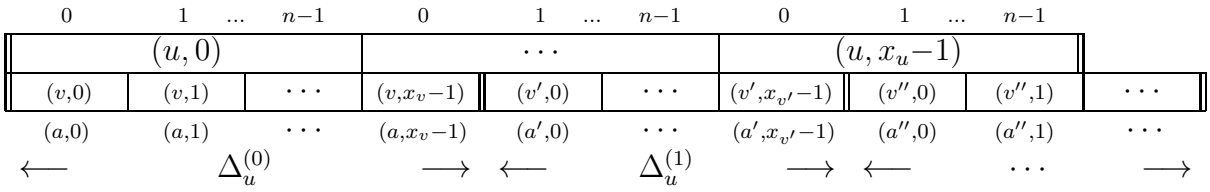


Figure 6.1: Stethering encoders.

Suppose that we have enough excess capacity that $\mathsf{cap}(S) \geq \log(n{+}1)$. Then, using an $(A_G, n{+}1)$-approximate eigenvector, we can form a stethering $(S, n{+}1)$-encoder $\mathcal{E}$. Now, for each state u, the ordering on $L(E_u)$ induces a lexicographic ordering on Δ_u, and hence an ordering on each $\Delta_u^{(i)}$. We tag $\mathcal{E}$ by assigning the input tags $\{0, 1, \ldots, n\}$ in an ordering-preserving way to the outgoing edges specified by each $\Delta_u^{(i)}$. From $\mathcal{E}$ we form an (S, n)-encoder $\mathcal{E}'$ by deleting all edges in $\mathcal{E}$ tagged by the input symbol n. We call such an encoder a *punctured stethering* (S, n)*-encoder*. In [AMR95], it is shown that any punctured stethering (S, n)-encoder has finite anticipation.

Now, what is the advantage of such an encoder? Well, not only is it easy to construct, but, as we will discuss in Section 7.4, its anticipation is in some sense 'small,' especially compared to the upper bound on anticipation that we gave in Section 5.3. Also, with sufficient excess capacity, this construction leads to sliding-block decoders with 'small' decoding window length (see the discussion in Section 7.4).

6.3 Generalized tagged (S, n)-encoders

In order to describe some of the other approaches to code construction, we will now formulate what appears to be a more general notion of tagged (S, n)-encoder. However, these more general encoders can be transformed to ordinary tagged (S, n)-encoders.

A *generalized tagged* (S, n)*-encoder* $\mathcal{E} = (V, E, L_I/L_O)$ is a finite directed graph (V, E) endowed with two labelings—input labeling (tagging) L_I and output labeling L_O—yielding two labeled graphs $G_I = (V, E, L_I)$ and $G_O = (V, E, L_O)$ such that

1. G_I presents the unconstrained system of all n-ary sequences;

2. G_I has finite anticipation;

3. $S(G_O) \subseteq S$;

4. G_O is lossless.

As with ordinary tagged encoders, we will use the notation $u \stackrel{s/b}{\to} v$ for an edge from state u to state v in $\mathcal{E}$ with L_I-labeling (tagging) s and L_O-labeling b.

The main difference between a generalized tagged encoder and an ordinary tagged encoder is that the input tagging is merely required to have finite anticipation, rather than to be deterministic. However, since G_I presents all n-ary sequences, it can be shown directly that $(V, E, L_I/L_O)$ can encode unconstrained data into S at the cost of finite delay. Alternatively, we can transform any generalized tagged (S, n)-encoder into an (ordinary) tagged (S, n)-encoder. This fact is implicit in many papers (e.g., [AFKM86], [Heeg91, p. 770]) and has recently been made more explicit by Hollmann [Holl95]. However, it is also a simple consequence of an old symbolic dynamics result. The transformation is outlined in Section 6.7.

6.4 Encoders through variable-length graphs

6.4.1 Variable-length graphs and n-codability

In the following, we show how a certain kind of labeled variable-length graph yields a generalized tagged (S, n)-encoder (and therefore an ordinary tagged (S, n)-encoder).

A *(labeled) variable-length graph* (in short, VLG) is a labeled finite directed graph $G = (V, E, L)$ in which each edge e has a length $\ell(e)$, assumed to be a positive integer, and is labeled by a word $L(e)$ (in some alphabet) of length $\ell(e)$. An ordinary labeled finite directed graph is a VLG where each edge has length 1.

Any VLG, G, may be viewed as an ordinary labeled graph by inserting dummy states. In this way, a VLG presents a constrained system. So, the notation $S(G)$ makes sense for a VLG, and we can apply the various notions of determinism, losslessness, definiteness, finite memory, finite anticipation, etc., to VLG's. An advantage of VLG's is that they tend to be more compact representations of constraints. For instance, any (d, k)-RLL constraint naturally has an ordinary presentation with $k+1$ states and a VLG presentation with only one state, as shown in Figure 6.2.

A VLG G is *n-codable* if at each state $u \in V_G$, the length distribution of the set E_u of outgoing edges satisfies the Kraft inequality with equality, namely, we have

$$\sum_{e \in E_u} n^{-\ell(e)} = 1 \ .$$

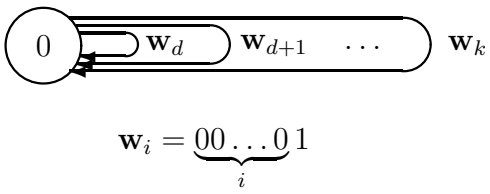


Figure 6.2: VLG presentation of (d, k)-RLL constrained system.

Now, suppose that G is an n-codable lossless VLG presentation of a subset of a constrained system S. Then, we can select, for each state $u \in V_G$, a prefix-free list X_u of n-ary words whose length distribution matches that of E_u. Using any length-preserving correspondence between X_u and E_u, we endow G with an input tagging L_I. It can be shown that $(V, E, L_I/L)$, viewed as an ordinary graph with two labelings—input tagging L_I and output labeling L—is a generalized tagged (S, n)-encoder (Problem 6.2).

6.4.2 Variable-length state splitting

Variable-length state splitting is a code construction technique, due to Adler, Friedman, Kitchens, and Marcus [AFKM86] (see also Heegard, Marcus, and Siegel [Heeg91]), which adapts the state-splitting algorithm to variable-length graphs. In this method, we begin with an ordinary lossless presentation $G = (V, E, L)$ of a constrained system S, and we select a subset $V' \subseteq V$ of states such that every sufficiently long path in G meets some state of V'. We then convert G into a VLG $G' = (V', E', L')$, where E' is the set of paths in G which start and terminate in V', but do not contain any states of V' in their interior, with the naturally inherited labeling (the length of an edge in G' is, of course, the length of the path in G that it represents). The assumption on V' guarantees that every bi-infinite path in G can be parsed into edges of G', and so we do not really lose anything in passing from G to G'. We remark that if we view G' as an ordinary labeled graph (by inserting dummy states), then, while G' may not exactly reproduce G, it is a lossless labeled graph with $S(G') = S(G)$.

One can develop notions of state splitting, approximate eigenvectors, etc. for VLG's, and see that state splitting preserves finite anticipation and definiteness as in Proposition 5.1. Then we apply a variable-length version of the state-splitting algorithm to iteratively transform G' into a new VLG which is n-codable, thereby yielding a generalized tagged (S, n)-encoder. If the original presentation G has finite anticipation (resp., has finite memory), then the encoder will have finite anticipation (resp., be sliding-block decodable).

The variable-length state-splitting procedure offers the advantage of a simpler construction method because there are fewer states to split. In some cases, it also suggests ways to find codes with reduced complexity (see [AFKM86], [Heeg91] for examples).

6.4.3 Method of poles

Our next alternative method is the *method of poles*, due to Béal [Béal90a], [Béal93a], [Béal90b]. This method is a modification of the *method of principal states*, developed by Franaszek [Fra69]. A related method was used in [HorM76] to obtain the first practical rate $2:3$ finite-state encoder for the $(1,7)$-RLL constraint.

In this method, we assume that $\mathsf{cap}(S) > \log n$, although it does work sometimes even when $\mathsf{cap}(S) = \log n$ (see [AB94]). Again, we begin with an ordinary lossless presentation G of S. Given a positive integer M, we find a subset P of V, called *principal states*, and, for each $u \in P$, a collection $\Gamma(u)$ of paths in G satisfying the following four conditions:

(PS-1) each path in $\Gamma(u)$ starts at u;

(PS-2) each path terminates in some element of P;

(PS-3) each path has length at most M;

(PS-4) the reverse Kraft inequality holds at every state $u \in P$, namely,

$$\sum_{\gamma \in \Gamma(u)} n^{-\ell(\gamma)} \geq 1 \ .$$

Note that, in contrast to variable-length state splitting, paths in $\Gamma(u)$ are allowed to visit states of P in their interior (i.e., not just at the beginning or end). In Section 4.4 we presented a special case of this definition, where each $\Gamma(u)$ consisted of paths of length exactly $q = M$.

Assuming $\mathsf{cap}(S) > \log n$, for sufficiently large M, such a set of principal states always exists [Béal90a]. There are algorithms which give short-cuts to an exhaustive search for sets of principal states [Fra69], [Béal90a].

A generalized tagged (S,n)-encoder with sliding-block decoder can be found using the lists $\Gamma(u)$ as follows. We first extract a subset $\Gamma'(u) \subseteq \Gamma(u)$ such that the following condition holds:

(PS-4') $\quad \sum_{\gamma \in \Gamma'(u)} n^{-\ell(\gamma)} = 1.$

This can always be done [Béal90a]. We now define a VLG presentation $\overline{G} = (\overline{V}, \overline{E}, \overline{L})$ of a subset of $S = S(G)$ as follows. For each principal state $u \in P$, we endow $\overline{G}$ with a state $\overline{u}$, called a *pole state*, and for each path $\gamma \in \Gamma'(u)$ in G terminating in a state $v \in P$, we endow $\overline{G}$ with an edge $\overline{\gamma} = \overline{u} \to \overline{v}$ of length $\ell(\gamma)$. The edge $\overline{\gamma}$ in $\overline{G}$ is labeled with the word generated by the corresponding path γ in G.

Clearly, $S(\overline{G}) \subseteq S(G) = S$, and it is not hard to see that since G is lossless, so is $\overline{G}$. Moreover, the VLG $\overline{G}$ is n-codable. So, as described in Section 6.4.1, we can endow $\overline{G}$ with an input tagging to obtain a generalized tagged (S, n)-encoder.

Now, the presentation $\overline{G}$ need not be definite even if G has finite memory. However, Béal [Béal90a] showed that if G does have finite memory and the lists $\Gamma'(u)$ satisfy an optimization condition (condition (PS-5) below), then $\overline{G}$ will be definite. Namely, she showed that if G has finite memory and $P \subseteq V_G$ is a set of states such that for each $u \in P$, the list $\Gamma'(u)$ satisfies conditions (PS-1)–(PS-3), (PS-4'), and (PS-5), then $\overline{G}$ is definite, and so any tagging of the resulting (S, n)-encoder will be sliding-block decodable. The additional condition (PS-5) is as follows:

(PS-5) for each state $u \in P$, the list $\Gamma'(u)$ minimizes the sum $\sum_{\gamma \in \Gamma'(u)} \ell(\gamma)$, among all possible lists satisfying conditions (PS-1)–(PS-3) and (PS-4').

We note that Béal [Béal93b] has found other optimality conditions that can replace condition (PS-5) for guaranteeing sliding-block decodability.

6.5 Look-ahead encoders

Next, we discuss some methods, due to Franaszek [Fra79], [Fra82], [Fra89], which predate the state-splitting approach, but involve some very closely related ideas. In our description, we adopt the viewpoint of Hollmann [Holl95], whose terminology is somewhat different from that of Franaszek.

We begin with look-ahead encoding, a variation of ordinary finite-state coding. At each state u of the encoder, the allowable input tag sequences are not arbitrary; namely, there is a list of n-ary words $W(u)$, all of the same length K, such that an input tag sequence labeling a path from u is allowable if and only if its K-prefix belongs to $W(u)$. So, when we encode an input tag (an input symbol), we do so with the commitment to encode thereafter only a certain specified collection of input tag sequences. The encoded input tag and the next encoder state are dictated by the upcoming sequence of input tags, subject to the requirement that each edge in the encoder graph is associated with exactly one input tag.

In order to describe look-ahead encoding precisely, we will make use of the following notation: for sets U and V of (finite) words, we denote by UV the set of all concatenations uv such that $u \in U$ and $v \in V$. We will use the notation B_n for the n-ary set $\{0, 1, \ldots, n-1\}$.

A *look-ahead encoder* (or more precisely, a *K-tag look-ahead encoder with input alphabet B_n*) for a lossless graph $G = (V, E, L)$ is defined by an input tagging $L_I : E \to B_n$ of the edges of G and subsets $W(u)$ of $(B_n)^K$ for each state $u \in V$ such that the following two conditions hold:

(LA-1) for at least one state $u \in V$ we have $W(u) \neq \emptyset$;

(LA-2) for each $u \in V$ we have

$$W(u)B_n \subseteq \cup_{e \in E_u} L_I(e)W(\tau_G(e)) \ .$$

Now, we can transform a look-ahead encoder into a generalized tagged $(S(G), n)$-encoder (and therefore an ordinary tagged $(S(G), n)$-encoder) as follows. For each state $u \in V$, $\mathbf{s} \in W(u)$, and $b \in B_n$, use condition (LA-2) above to choose a particular outgoing edge $e = e(u, \mathbf{s}b)$ from u in G such that $\mathbf{s}b = a\mathbf{s}'$, $a = L_I(e)$, and $\mathbf{s}' \in W(\tau_G(e))$. Now, construct a new graph $\mathcal{E} = (V', E', L'_I/L'_O)$ with two labelings—input labeling L'_I and output labeling L'_O—as follows. The states of $\mathcal{E}$ are all pairs $[u, \mathbf{s}]$, where $u \in V$ and $\mathbf{s} \in W(u)$. We draw an edge $[u, \mathbf{s}] \stackrel{a/c}{\rightarrow} [u', \mathbf{s}']$ (with L'_I-labeling a and L'_O-labeling c) if and only if the following three conditions hold:

(a) $\mathbf{s}b = a\mathbf{s}'$, where a is the first B_n-symbol in $\mathbf{s}$ and b is the last B_n-symbol in $\mathbf{s}'$.

(b) $u' = \tau_G(e(u, \mathbf{s}b))$.

(c) $c = L(e(u, \mathbf{s}b))$.

It can be shown that $\mathcal{E}$ is a generalized tagged (S, n)-encoder (Problem 6.5). Moreover, if G has finite anticipation (resp., has finite memory), then $\mathcal{E}$ has finite anticipation (resp., is sliding-block decodable). In particular, if G has finite memory $\mathcal{M}$, then $\mathcal{E}$ is sliding-block decodable with decoding window length $\leq \mathcal{M}+1$: any word of length $\mathcal{M}+1$ in $S(G)$ uniquely determines an edge e of G and the decoded input tag in $\mathcal{E}$ is then $L'_I(e)$. So, if G has memory zero, then the decoding window length is 1 (note that this does not mean that the encoder is block decodable, i.e., $(0, 0)$-sliding-block decodable).

A very special case of the main result in [AM95] is that for any K-tag look-ahead encoder for a labeled graph G, there is an equivalent encoder (in the sense that the encoding function is the same modulo a shift) obtained from G by at most K rounds of out-splitting. Now, if G has memory $\mathcal{M}$, then the upper bound in Section 5.5.2 given for the smallest decoding window length of a sliding-block decoder is $\mathcal{M}+K+1$. However, as mentioned above, such an encoder is sliding-block decodable with window length at most $\mathcal{M}+1$. So, here is an instance where the bound on the decoding window length given by the state-splitting algorithm is much larger than the actual decoding window length.

Lempel and Cohn [LemCo82] proposed a generalization of the look-ahead encoding method. Their method differs from look-ahead encoding in several respects. First, at each state, the set of input words is allowed to have variable lengths (this is not really an essential difference, and this possibility was already considered by Franaszek [Fra80a]). Second, the outgoing edge dictated by an input tag is allowed to depend on the sequence of previous input tags (as well as following input tags). Third, an edge in the encoder graph may be associated with more than one input tag. While this still gives a well-defined encoder, it

may not have finite anticipation, and so decoding can be problematic. Nevertheless, by a series of examples, Lempel and Cohn showed that this method works very well in many circumstances.

6.6 Bounded-delay encoders

The bounded-delay method is a generalization of look-ahead encoding. Let $G = (V, E, L)$ be a labeled graph. The *Tth order Moore form of G*, denoted $G^{\{T\}}$, is the labeled graph defined as follows. The states of $G^{\{T\}}$ are the paths of length $T-1$ in G, and for each path $e_1 e_2 \dots e_T$ of length T in G, there is an edge $e_1 e_2 \dots e_{T-1} \to e_2 e_3 \dots e_T$ in $G^{\{T\}}$ labeled $L(e_T)$. The labeled graph $G^{\{2\}}$ is the ordinary Moore form of G, as was defined in Section 2.2.7.

Let $G = (V, E, L)$ be a lossless presentation of a constrained system S, and let K and T be positive integers. A *bounded-delay encoder* (or more precisely, a *K-tag, delay-T bounded-delay encoder with input alphabet B_n*) for a lossless graph G is a K-tag look-ahead encoder with input alphabet B_n for the Tth order Moore form $G^{\{T\}}$ of G. Observe that if G has finite memory $\mathcal{M}$, then such an encoder is sliding-block decodable with decoding window length $\mathcal{M}+T+1$. We remark that Franaszek [Fra82] originally framed his IP approach (discussed in Section 6.1) in terms of bounded-delay encoding.

Hollmann [Holl95] has recently developed an approach which combines features of the bounded-delay method and the state-splitting method. The idea is to first split states sufficiently until one obtains a graph which is amenable to a certain variation of the bounded-delay method. In many cases, the result of this procedure is a sliding-block decodable encoder whose decoder has smaller window length than would be expected. Several very nice codes for specific constraints of practical importance were constructed in [Holl95].

Hollmann's approach was influenced by earlier work of Immink [Imm92], as well as by Franaszek's bounded-delay encoding method. Immink showed that in many situations, while there may be no block decodable tagged (S, n)-encoder, it may still be possible to construct a tagged (S, n)-encoder which is $(-1, 1)$-sliding-block decodable. The decoder produces a decoded input tag at time i, by examining only the output symbol at time $i+1$; in other words, it decodes by looking into the future and ignoring the present. What does this mean in terms of the input tagging of an (S, n)-encoder $\mathcal{E}$? Well, suppose, for simplicity, that $\mathcal{E}$ has memory 1 and at each state, the input tags on the *incoming* edges are all the same. Then, any output symbol uniquely determines a state v in $\mathcal{E}$ and is decoded to the input tag which is written on the incoming edges to v. Of course, it is not at all obvious how to construct such encoders. However, useful heuristics, illustrated by several examples, were given in [Imm92], as well as in subsequent papers by Immink [Imm95a] and Hollmann [Holl94]. In [Holl95], [Holl96], Hollmann develops codes that look 'further' into the future: in his construction, he aims for $(-\mathsf{m}, \mathsf{a})$-sliding-block decodable encoders, where $0 \le \mathsf{m} \le \mathsf{a}$; that is, the decoder produces a decoded input tag at time i by examining only

the output symbols $w_{i+\mathsf{m}}w_{i+\mathsf{m}-1}\ldots w_{i+\mathsf{a}}$.

6.7 Transforming a generalized encoder to an ordinary encoder

Here, we outline how to transform a generalized tagged (S,n)-encoder to an ordinary tagged (S,n)-encoder.

Let G be a labeled graph with finite anticipation $\mathcal{A}$. We define the *induced labeled graph* $G'=(V',E',L')$ of G in the following manner. The states of G' are all pairs $[u,\mathbf{s}]$, where $u\in V_G$ and $\mathbf{s}$ is a word of length $\mathcal{A}$ that can be generated from u in G. We draw an edge $[u,\mathbf{s}]\stackrel{b}{\rightarrow}[u',\mathbf{s}']$ if and only if the following two conditions hold:

(a) $\mathbf{s}b=a\mathbf{s}'$, where a is the first symbol in $\mathbf{s}$;

(b) u' is the terminal state of the first (unique) edge $u\stackrel{a}{\rightarrow}u'$, denoted $e(u,\mathbf{s}b)$, in any path of length $\mathcal{A}+1$ that is labeled by the word $\mathbf{s}b=a\mathbf{s}'$ in G starting at state u.

Kitchens [Kit81] (see also [BKM85]) showed that whenever $G=(V,E,L)$ is a labeled graph with finite anticipation, then the corresponding induced labeled graph G' is deterministic and $S(G)=S(G')$ (Problem 6.6).

Now, given a generalized tagged (S,n)-encoder $\mathcal{E}=(V,E,L_I/L_O)$, we apply the preceding result to $\mathcal{E}_I=(V,E,L_I)$, yielding a deterministic induced tagged graph $\mathcal{E}'_I=(V',E',L'_I)$ (i.e., the labels given by L'_I are regarded as "input tags"). We form a generalized tagged (S,n)-encoder $\mathcal{E}'=(V',E',L'_I/L'_O)$ by endowing $\mathcal{E}'_I$ with an output labeling L'_O to reflect the labeling L_O: namely, the L'_O-label, c, of the edge $[u,\mathbf{s}]\stackrel{b/c}{\longrightarrow}[u',\mathbf{s}']$ in $\mathcal{E}'$ is set to $c=L_O(e(u,\mathbf{s}b))$.

Clearly, both $\mathcal{E}_I$ and $\mathcal{E}'_I$ present the set of all n-ary words through their labelings L_I and L'_I. Hence, since $\mathcal{E}'_I$ is deterministic, there is an (input-)labeled subgraph $\mathcal{E}''_I=(V'',E'',L'_I)$ of $\mathcal{E}'_I$ with constant out-degree n which still presents the unconstrained system of n-ary words. Now, by the losslessness of $\mathcal{E}_O=(V,E,L_O)$ we have that the labeled graph $\mathcal{E}'_O=(V',E',L'_O)$ and, therefore, also $\mathcal{E}''_O=(V'',E'',L'_O)$, is an (S,n)-encoder. The tagging L'_I then converts it into an (ordinary) tagged (S,n)-encoder $\mathcal{E}''=(V'',E'',L'_I/L'_O)$. In this way, we have transformed the generalized tagged (S,n)-encoder $\mathcal{E}=(V,E,L_I/L_O)$ to the ordinary tagged (S,n)-encoder $\mathcal{E}''$ (although the encoding mapping is shifted in the process). Moreover, if $\mathcal{E}_O$ has finite anticipation, then the resulting (S,n)-encoder $\mathcal{E}''_O$ will have finite anticipation, and if $\mathcal{E}_O$ has finite memory, then the resulting tagged (S,n)-encoder $\mathcal{E}''$ will be sliding-block decodable. Therefore, the notions of finite anticipation and sliding-block decodability can be applied to generalized encoders.

The induced labeled graph described above resembles that produced by the determinizing

construction of Section 2.2.1. However, it is different: the determinizing construction yields a labeled graph which is 'too small' to allow the definition of a well-defined labeling L'_O that reflects the labeling L_O.

Problems

Problem 6.1 Show that the graph G', constructed in Section 6.1, is lossless.

Problem 6.2 Show that $(V, E, L_I/L)$, as described at the end of Section 6.4.1, is a generalized tagged (S, n)-encoder (when viewed as an ordinary graph with two labelings—input tagging L_I and output labeling L).

Problem 6.3 Let $G = (V, E, L)$ be a variable-length graph (VLG). Denote by $Q_G(z)$ the $|V| \times |V|$ matrix in the indeterminate z, with entries given by

$$(Q_G(z))_{u,v} = \sum_e z^{-\ell(e)} ,$$

where $u, v \in V$ and the summation is taken over all edges e in G that originate in u and terminate in v.

Define the *ordinary form of* G as the ordinary graph H obtained from G by replacing each edge e, of length $\ell(e)$, with a chain of $\ell(e)$ edges connected through $\ell(e)-1$ new states.

1. Show that if G is an ordinary graph (i.e., $\ell(e) = 1$ for every $e \in E$), then
$$Q_G(z) = z^{-1} A_G ,$$
where A_G is the adjacency matrix of G.

2. Show that if H is the ordinary form of a VLG G, then μ is a nonzero eigenvalue of A_H if and only if
$$\det(Q_G(\mu) - I) = 0 .$$
In particular, the largest real solution of $\det(Q_G(z) - I) = 0$ is $z = \lambda(A_H)$.

Problem 6.4 A finite set Γ of words over an alphabet Σ is called *exhaustive* if every word $\mathbf{w}$ over Σ is either a prefix of a word in Γ or there is a word in Γ that is a prefix of $\mathbf{w}$. Denote by $\ell(\mathbf{w})$ the length of the word $\mathbf{w}$.

1. Let Γ be an exhaustive set of words over an alphabet of size n. Show that
$$\sum_{\mathbf{w} \in \Gamma} n^{-\ell(\mathbf{w})} \geq 1 .$$
Hint: Let $\ell_{\max}$ be the largest length of any word in Γ; show that
$$\sum_{\mathbf{w} \in \Gamma} n^{\ell_{\max} - \ell(\mathbf{w})} \geq n^{\ell_{\max}} .$$

2. Let $m_1, m_2, \ldots, m_t$ be nonnegative integers that satisfy

$$\sum_{i=1}^{t} n^{-m_i} \geq 1 \ .$$

Show that there exist integers $\ell_1, \ell_2, \ldots, \ell_t$ such that $\ell_i \geq m_i$ and

$$\sum_{i=1}^{t} n^{-\ell_i} = 1 \ .$$

3. A finite set Γ of words over am alphabet Σ is called *prefix-free* is no word in Γ is a prefix of any other word in Γ. Show that if Γ is prefix-free, then

$$\sum_{\mathbf{w} \in \Gamma} n^{-\ell(\mathbf{w})} \leq 1 \ .$$

4. Let $\ell_1, \ell_2, \ldots, \ell_t$ be nonnegative integers that satisfy the equality

$$\sum_{i=1}^{t} n^{-\ell_i} = 1 \ .$$

Suggest an efficient algorithm for constructing a set Γ of t words over an alphabet of size n such that the following three conditions hold:

(a) Γ is exhaustive;

(b) Γ is prefix-free; and —

(c) the ith word in Γ has length ℓ_i.

Problem 6.5 Let $\mathcal{E}$ be the encoder, based on a lossless graph G, described in Section 6.5.

1. Show that $\mathcal{E}$ is a generalized tagged (S, n)-encoder.
2. Show that if G has finite anticipation (resp., has finite memory), then $\mathcal{E}$ has finite anticipation (resp., is sliding-block decodable).

Problem 6.6 Let G be a labeled graph with finite anticipation, and let G' be the induced labeled graph described in Section 6.7.

1. Show that G' is deterministic.
2. Show that $S(G) = S(G')$.

Chapter 7

Complexity of Encoders

7.1 Complexity criteria

There are various criteria that are used to measure the performance and complexity of encoders, and their corresponding decoders. We list here the predominant factors that are usually taken into account while designing rate $p : q$ finite-state encoders.

The values of p and q. Typically, the rate p/q is chosen to be as close to $\mathsf{cap}(S)$ as possible, subject to having p and q small enough: The reason for the latter requirement is minimizing the number of outgoing edges, 2^p, from each state in the encoder and keeping to a minimum the number of input–output connections of the encoder.

Number of states in an encoder. In both hardware and software implementation of encoders $\mathcal{E}$, we will need $\lceil \log |V_{\mathcal{E}}| \rceil$ bits in order to represent the current state of $\mathcal{E}$. This motivates an encoder design with a relatively small number of states [Koh78, Ch. 9].

Gate complexity. In addition to representing the state of a finite-state encoder, we need, in hardware implementation, to realize the next-state function and the output function as a gate circuit. Hardware complexity is usually measured in terms of the number of required gates (e.g., NAND gates), and this number also includes the implementation of the memory bit cells that represent the encoder state (each memory bit cell can be realized by a fixed number of gates).

The number of states in hardware implementation becomes more significant in applications where we run several encoders *in parallel* with a common circuit for the next-state and output functions, but with duplicated hardware for representing the state of each encoder.

Time and space complexity of a RAM program. When the finite-state encoder is to be implemented as a computer program on a random-access machine (RAM) [AHU74,

Ch. 1], the complexity is usually measured by the space requirements and the running time of the program.

Encoder anticipation. One way to implement a decoder for an encoder $\mathcal{E}$ with finite anticipation $\mathcal{A}(\mathcal{E})$ is by accumulating the past $\mathcal{A}(\mathcal{E})$ symbols (in $\Sigma(S^q)$) that were generated by $\mathcal{E}$; these symbols, with the current symbol, allow the decoder to simulate the state transitions of $\mathcal{E}$ and, hence, to reconstruct the sequence of input tags (see Section 4.1). The size of the required buffer thus depends on $\mathcal{A}(\mathcal{E})$.

Window length of sliding-block decodable encoders. A typical decoder of an (m, a)-sliding-block decodable encoder consists of a buffer that accumulates the past $\mathsf{m}+\mathsf{a}$ symbols (in $\Sigma(S^q)$) that were generated by the encoder. A decoding function $\mathcal{D} : (\Sigma(S^q))^{(\mathsf{m}+\mathsf{a}+1)} \rightarrow \{0, 1, \ldots, 2^p-1\}$ is then applied to the current symbol and to the contents of the buffer to reconstruct an input tag in $\{0,1\}^p$ (see Section 4.3). From a complexity point-of-view, the window length, $\mathsf{m}+\mathsf{a}+1$, determines the size of the required buffer.

In order to establish a general framework for comparing the complexity of encoders generated by different methods of encoder synthesis, we need to set some canonical presentation of the constrained system S, in terms of which the complexity will be measured. We adopt the Shannon cover of S as such a distinguished presentation.

7.2 Number of states in the encoder

In this section we present upper and lower bounds on the smallest number of states in any (S, n)-encoder for a given constrained system S and integer n.

Let G be a deterministic presentation of S. As was described in Chapter 5, the state-splitting algorithm [ACH83] starts with an (A_G, n)-approximate eigenvector $\mathbf{x} = (x_v)_{v \in V_G}$, which guides the splitting of the states in G until we obtain an (S, n)-encoder with at most $\sum_{v \in V_G} x_v = \|\mathbf{x}\|_1$ states. Hence, we have the following.

Theorem 7.1 *Let S be a constrained system presented by a deterministic graph G and let n be a positive integer. Assume that $\mathsf{cap}(S) \geq \log n$. Then, there exists an (S, n)-encoder $\mathcal{E}$ such that*

$$|V_{\mathcal{E}}| \leq \min_{\mathbf{x} \in \mathcal{X}(A_G, n)} \|\mathbf{x}\|_1 \ .$$

On the other hand, the following lower bound on the number of states of any (S, n)-encoder was obtained in [MR91].

Theorem 7.2 *Let S be a constrained system presented by a deterministic graph G and let n be a positive integer. Assume that $\mathsf{cap}(S) \geq \log n$. Then, for any (S,n)-encoder $\mathcal{E}$,*

$$|V_{\mathcal{E}}| \geq \min_{\mathbf{x} \in \mathcal{X}(A_G,n)} \|\mathbf{x}\|_\infty \ .$$

Proof: Let $\mathcal{E}$ be an (S,n)-encoder and let $\Sigma = \Sigma(S)$. The following construction effectively provides an (A_G,n)-approximate eigenvector $\mathbf{x}$ which satisfies the inequality $|V_{\mathcal{E}}| \geq \|\mathbf{x}\|_\infty$.

(a) Construct a deterministic graph $H = H(\mathcal{E})$ which presents $S' = S(\mathcal{E})$.

This can be done using the determinizing graph of Section 2.2.1.

(b) For an irreducible sink H' of H, define a vector $\boldsymbol{\xi} \neq \mathbf{0}$ such that $A_{H'}\boldsymbol{\xi} = n\boldsymbol{\xi}$.

Let H' be an irreducible sink of H. Recall that each state $Z \in V_{H'} \left(\subseteq V_H\right)$ is a subset $T_{\mathcal{E}}(\mathbf{w}, v)$ of states of $\mathcal{E}$ that can be reached in $\mathcal{E}$ from a given state $v \in V_{\mathcal{E}}$ by paths that generate a given word $\mathbf{w}$. Let $\xi_Z = |Z|$ denote the number of states of $\mathcal{E}$ in Z and let $\boldsymbol{\xi}$ be the positive integer vector defined by $\boldsymbol{\xi} = (\xi_Z)_{Z \in V_{H'}}$. We now claim that

$$A_{H'}\boldsymbol{\xi} = n\boldsymbol{\xi} \ .$$

Consider a state $Z \in V_{H'}$. Since $\mathcal{E}$ has out-degree n, the number of edges in $\mathcal{E}$ outgoing from the set of states $Z \subseteq V_{\mathcal{E}}$ is $n|Z|$. Now, let E_a denote the set of edges in $\mathcal{E}$ labeled a that start at the states of $\mathcal{E}$ in Z and let Z_a denote the set of terminal states, in $\mathcal{E}$, of these edges. Note that the sets E_a, for $a \in \Sigma$, induce a partition on the edges of $\mathcal{E}$ outgoing from Z. Clearly, if $Z_a \neq \emptyset$, there is an edge $Z \xrightarrow{a} Z_a$ in H and, since H' is an irreducible sink, this edge is also contained in H'. We now claim that any state $u \in Z_a$ is accessible in $\mathcal{E}$ by exactly one edge labeled a whose initial state is in Z; otherwise, if $Z = T_{\mathcal{E}}(\mathbf{w}, v)$, the word $\mathbf{w}a$ could be generated in $\mathcal{E}$ by two distinct paths which start at v and terminate in u, contradicting the losslessness of $\mathcal{E}$. Hence, $|E_a| = |Z_a|$ and, so, the entry of $A_{H'}\boldsymbol{\xi}$ corresponding to the state Z in H' satisfies

$$\begin{aligned} (A_{H'}\boldsymbol{\xi})_Z &= \sum_{Y \in V_{H'}} (A_{H'})_{Z,Y}\xi_Y = \sum_{Y \in V_{H'}} (A_{H'})_{Z,Y}|Y| \\ &= \sum_{a \in \Sigma} |Z_a| = \sum_{a \in \Sigma} |E_a| = n|Z| = n\xi_Z \ , \end{aligned}$$

as desired.

(c) Construct an (A_G,n)-approximate eigenvector $\mathbf{x} = \mathbf{x}(\mathcal{E})$ from $\boldsymbol{\xi}$.

As G and H' comply with the conditions of Lemma 2.13, each follower set of a state in H' is contained in a follower set of some state in G. Let $\mathbf{x} = (x_u)_{u \in V_G}$ be the nonnegative integer vector defined by

$$x_u = \max\{\, \xi_Z \,:\, Z \in V_{H'} \text{ and } \mathcal{F}_{H'}(Z) \subseteq \mathcal{F}_G(u) \,\} \ , \quad u \in V_G \ ,$$

and denote by $Z(u)$ some particular state Z in H' for which the maximum is attained. In case there is no state $Z \in V_{H'}$ such that $\mathcal{F}_{H'}(Z) \subseteq \mathcal{F}_G(u)$, define $x_u = 0$ and $Z(u) = \emptyset$. We claim that $\mathbf{x}$ is an (A_G, n)-approximate eigenvector. First, since $V_{H'}$ is nonempty, we have $\mathbf{x} \neq \mathbf{0}$. Now, let u be a state in G; if $x_u = 0$ then, trivially, $(A_G\mathbf{x})_u \geq nx_u$ and, so, we can assume that $x_u \neq 0$. Let $Z_a(u)$ be the terminal state in H' for an edge labeled a outgoing from $Z(u)$. Since $\mathcal{F}_{H'}\Big(Z(u)\Big) \subseteq \mathcal{F}_G(u)$, there exists an edge labeled a in G from u which terminates in some state u_a in G; and, since G and H' are both deterministic, we have $\mathcal{F}_{H'}\Big(Z_a(u)\Big) \subseteq \mathcal{F}_G(u_a)$. Furthermore, by the way $\mathbf{x}$ was defined, we have $x_{u_a} \geq \xi_{Z_a(u)}$ and, so, letting $\Sigma_{Z(u)}$ denote the set of labels of edges in H' outgoing from $Z(u)$, we have

$$(A_G\mathbf{x})_u \geq \sum_{a \in \Sigma_{Z(u)}} x_{u_a} \geq \sum_{a \in \Sigma_{Z(u)}} \xi_{Z_a(u)} = (A_{H'}\boldsymbol{\xi})_{Z(u)} = n\xi_{Z(u)} = nx_u \ ,$$

where we have used the equality $A_{H'}\boldsymbol{\xi} = n\boldsymbol{\xi}$. Hence, $A_G\mathbf{x} \geq n\mathbf{x}$.

The theorem now follows from the fact that each entry in $\mathbf{x}$ is a size of a subset of states of $V_{\mathcal{E}}$. □

The bound of Theorem 7.2 can be effectively computed by the Franaszek algorithm which was described in Section 5.2.2. The upper bound of Theorem 7.1 is at most $|V_G|$ times the lower bound of Theorem 7.2, which amounts to an additive term of $\log |V_G|$ in the number of bits required to represent the current state of the encoder.

There are examples of sequences of labeled graphs G for which the lower bound of Theorem 7.2 is *exponential* in the number of states of G [Ash88], [MR91]. We give here such an example, which appears in [AMR95] and [MR91].

Example 7.1 Let r be a positive integer and let Σ denote the alphabet of size r^2+r+1 given by $\{a\} \cup \{b_i\}_{i=1}^{r^2+r-1} \cup \{c\}$. Consider the constrained systems S_k that are presented by the graphs G_k of Figure 7.1 (from each state $u \leq k$ there are r^2+r-1 parallel outgoing edges labeled by the b_i's to state $k+u$). It is easy to verify that $\lambda(A_{G_k}) = \lambda = r+1$ and that every (A_{G_k}, λ)-approximate eigenvector is a multiple of $(\lambda \ \lambda^2 \ \ldots \ \lambda^k \ 1 \ \lambda \ \ldots \ \lambda^{k-1})^\top$. Hence, by Theorem 7.2, every $(S_k, r+1)$-encoder must have at least $(r+1)^k = \exp\{O(|V_{G_k}|)\}$ states.

On the other hand, note that the vector $\mathbf{x} = (x_u)_u$, whose nonzero components are $x_k = r$ and $x_{2k} = 1$, is an (A_{G_k}, r)-approximate eigenvector. Hence, if we can compromise on the rate and construct (S_k, r)-encoders instead, then the state-splitting algorithm provides such encoders with at most $r+1$ states. □

The bound of Theorem 7.2 is based on the existence of an approximate eigenvector $\mathbf{x} = \mathbf{x}(\mathcal{E})$, where each of the entries in $\mathbf{x}$ is a size of a subset of $V_{\mathcal{E}}$. The improvements on this bound, given in [MR91], are obtained by observing that some of these subsets might be disjoint. One such improvement (with proof left to the reader) is as follows.

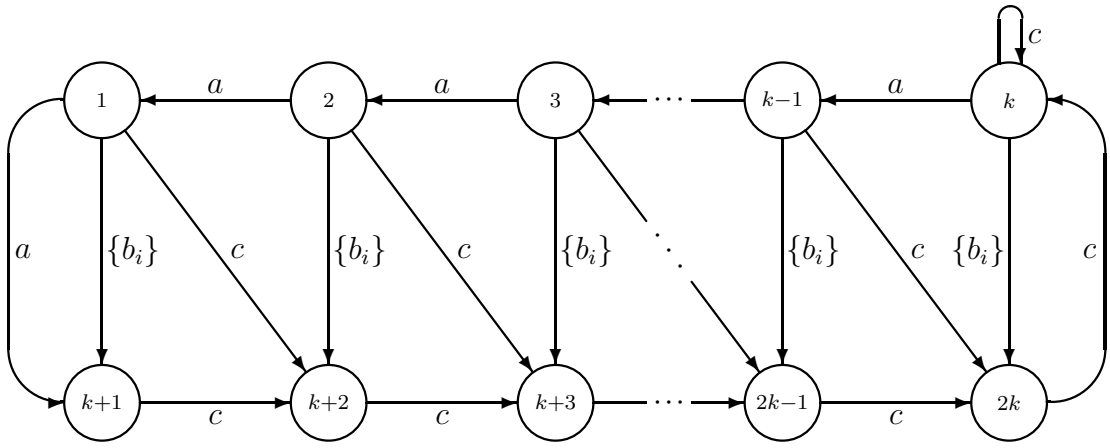


Figure 7.1: Labeled graph G_k for Example 7.1.

Theorem 7.3 *Let S be a constrained system presented by a deterministic graph G and let n be a positive integer. Assume that $\mathsf{cap}(S) \geq \log n$. Then, for any (S, n)-encoder $\mathcal{E}$,*

$$|V_{\mathcal{E}}| \geq \min_{\mathbf{x} \in \mathcal{X}(A_G, n)} \max_{U} \sum_{u \in U} x_u,$$

where the maximum is taken over all subsets $U \subseteq V_G$ such that $\mathcal{F}_G(u) \cap \mathcal{F}_G(u') = \emptyset$ for every distinct states u and u' in U.

In fact, the preceding result can be generalized further to obtain the best general lower bound known on the number of states in any (S, n)-encoder, as it is stated in [MR91]. In order to state this result, we need the following definitions.

Let S be a constrained system presented by a deterministic graph G. For a state $u \in V_G$ and a word $\mathbf{w} \in \mathcal{F}_G(u)$, let $\tau_G(\mathbf{w}, u)$ be the terminal state of the path in G that starts at u and generates $\mathbf{w}$. (Using the notations of Section 2.2.1, we thus have $T_G(\mathbf{w}, u) = \{\tau_G(\mathbf{w}, u)\}$.) For a word $\mathbf{w} \notin \mathcal{F}_G(u)$, define $\tau_G(\mathbf{w}, u) = \emptyset$.

Let n be a positive integer and $\mathbf{x} = (x_u)_{u \in V_G}$ be an (A_G, n)-approximate eigenvector. For a word $\mathbf{w}$ and a subset $U \subseteq V_G$, let $I_G(\mathbf{x}, \mathbf{w}, U)$ denote a state $u \in U$ such that $x_{\tau_G(\mathbf{w}, u)}$ is maximal (for the case where $\tau_G(\mathbf{w}, u) = \emptyset$, we define $x_\emptyset = 0$).

Let U be a subset of V_G. A list C of words is *U-complete* in G, if every word in $\bigcup_{u \in U} \mathcal{F}_G(u)$ either has a prefix in C or is a prefix of a word in C. Let $\mathcal{C}_G(U)$ denote the set of all finite U-complete lists in G. For example, the list $\mathcal{F}_G^m(U)$ of all words of length m that can be generated in G from states of U, belongs to $\mathcal{C}_G(U)$.

Finally, given an integer n, an (A_G, n)-approximate eigenvector $\mathbf{x}$, a subset U of V_G, and

a list C of words, we define $\mu_G(\mathbf{x}, n, U, C)$ by

$$\mu_G(\mathbf{x}, n, U, C) = \sum_{u \in U} x_u - \sum_{\mathbf{w} \in C} n^{-\ell(\mathbf{w})} \sum_{u \in U - \{I_G(\mathbf{x}, \mathbf{w}, U)\}} x_{\tau_G(\mathbf{w}, u)}$$

(recall that $\ell(\mathbf{w})$ is the length of the word $\mathbf{w}$).

Theorem 7.4 *Let S be a constrained system presented by a deterministic graph G and let n be a positive integer. Assume that $\mathsf{cap}(S) \geq \log n$. Then, for any (S, n)-encoder $\mathcal{E}$,*

$$|V_{\mathcal{E}}| \geq \min_{\mathbf{y} \in \mathcal{X}(A_G, n)} \max_{U \subseteq V_G} \sup_{C \in \mathcal{C}_G(U)} \mu_G(\mathbf{y}, n, U, C) \ .$$

In particular, for all $U \subseteq V_G$ and m,

$$|V_{\mathcal{E}}| \geq \min_{\mathbf{y} \in \mathcal{X}(A_G, n)} \mu_G(\mathbf{y}, n, U, \mathcal{F}_G^m(U)) \ .$$

Example 7.2 Figure 4.6 depicts a rate $2 : 3$ four-state encoder for the $(1, 7)$-RLL constrained system. The example therein is due to Weathers and Wolf [WW91], whereas the example of an encoder used in practice is due to Adler, Hassner, and Moussouris [AHM82] and has five states (see also [How89]). Using Theorem 7.4, a lower bound of 4 on the number of states of any such encoder is presented in [MR91]. Thus, the Weathers–Wolf encoder has the smallest possible number of encoder states. □

Example 7.3 Figure 4.4 depicts a rate $1 : 2$ six-state encoder for the $(2, 7)$-RLL constrained system. This encoder is used in practice and is due to Franaszek [Fra72] (see also [EH78], [How89]). On the other hand, the encoder shown in Figure 4.5, which is due to Howell [How89], has only five states. Using Theorem 7.4, it can be shown that 5 is a lower bound on the number of encoder states for this system; thus, the Howell encoder has the smallest possible number of encoder states (note, however, that the anticipation of the Howell encoder is larger than Franaszek's). □

7.3 Values of p and q

When $\mathsf{cap}(S)$ is a rational number p/q, we can attain the bound of Theorem 4.2 by a rate $p : q$ finite-state encoder for S: Taking a deterministic presentation G of S, we have in this case an $(A_G^q, 2^p)$-approximate eigenvector which yields, by the state-splitting algorithm, an $(S^q, 2^p)$-encoder.

On the other hand, when $\mathsf{cap}(S)$ is not a rational number, we cannot attain the bound of Theorem 4.2 by a rate $p : q$ finite-state encoder for S. Still, we can approach the bound

$\mathsf{cap}(S)$ from below by a sequence of rate $p_m : q_m$ finite-state encoders $\mathcal{E}_m$. In fact, as stated in Theorem 4.3, we can approach capacity from below even by block encoders. It can be shown that in this way we obtain a sequence of rate $p_m : q_m$ block encoders $\mathcal{E}_m$ for S such that

$$\left| \frac{p_m}{q_m} - \mathsf{cap}(S) \right| \leq \frac{\beta}{q_m}$$

for some constant $\beta = \beta(G)$. However, the constant β might be very large (e.g., exponential) in terms of the number of states of G. This means that q_m might need to be extremely large in order to have rates p_m/q_m close to capacity.

Obviously, for every sequence of rate $p_m : q_m$ finite-state encoders $\mathcal{E}_m$ for S, the number of edges in $\mathcal{E}_m$ is increasing exponentially with p_m. The question is whether convergence of p_m/q_m to $\mathsf{cap}(S)$ that is faster than $O(1/q_m)$ might force the number of states in $\mathcal{E}_m$ to blow up as well. The answer is given in the following result, which is proved in [MR91] using Theorem 7.2.

Theorem 7.5 *Let S be a constrained system with $\mathsf{cap}(S) = \log \lambda$.*

(a) If $\lambda = k^{s/t}$ for some positive integers k, s, and t, then there exists an integer N such that for any two positive integers p, q, where $p/q \leq \log \lambda$ and t divides q, there is an $(S^q, 2^p)$-encoder with at most N states.

(b) If λ is <u>not</u> a rational power of an integer and, in addition,

$$\lim_{m \to \infty} \left(\frac{p_m}{q_m} - \log \lambda \right) \cdot q_m = 0 \, ,$$

then for any sequence of $(S^{q_m}, 2^{p_m})$-encoders $\mathcal{E}_m$,

$$\lim_{m \to \infty} |V_{\mathcal{E}_m}| = \infty \, .$$

Sketch of proof. Case (a): Let G be a deterministic presentation of S. Since $\lambda = k^{s/t}$, the matrix $(A_G)^t$ has an integer largest eigenvalue and an associated integer nonnegative right eigenvector $\mathbf{x}$. An (S^{tm}, k^{sm})-encoder $\mathcal{E}_m$ can therefore be obtained by the state-splitting algorithm for every m, with number of states which is at most $N = \|\mathbf{x}\|_1$. Write $q = tm$; we have $2^p \leq k^{sm}$ and, so, an $(S^q, 2^p)$-encoder can be obtained by deleting excess edges from $\mathcal{E}_m$.

Case (b): It can be shown that if the values p_m/q_m approach $\log \lambda$ faster than $O(1/q_m)$, then the respective $((A_G)^{q_m}, 2^{p_m})$-approximate eigenvectors $\mathbf{x}$ (when scaled to have a fixed norm) approach a right eigenvector which must contain an irrational entry. Therefore, the largest components in such approximate eigenvectors tend to infinity. The result then follows from Theorem 7.2. □

If we choose p_m and q_m to be the continued fraction approximants of $\log \lambda$, we get

$$\left| \frac{p_m}{q_m} - \log \lambda \right| < \frac{\beta}{q_m^2}$$

for some constant β. So, in case (b), the fastest approach to capacity necessarily forces the number of states to grow without bound.

7.4 Encoder anticipation

7.4.1 Deciding upon existence of encoders with a given anticipation

We start with the following theorem, taken from [AMR96], which shows that checking whether there is an (S, n)-encoder with anticipation t is a decidable problem. A special case of this theorem, for $t = 0$, was alluded to in Section 4.4. Recall that $\mathcal{F}_G^t(u)$ stands for the set of words of length t that can be generated from a state u in a labeled graph G.

Theorem 7.6 *Let S be an irreducible constrained system with a Shannon cover G, let n and t be positive integers, and, for every state u in G, let $N(u,t) = |\mathcal{F}_G^t(u)|$. If there exists an (S,n)-encoder with anticipation t, then there exists an (S,n) encoder with anticipation $\leq t$ and at most $\sum_{u \in V_G}(2^{N(u,t)} - 1)$ states.*

By Lemma 2.9, we may assume that there is an *irreducible* (S, n)-encoder $\mathcal{E}$ with anticipation at most t. The proof of Theorem 7.6 is carried out by effectively constructing from $\mathcal{E}$ an (S, n)-encoder $\mathcal{E}'$ with anticipation $\leq t$ and with a number of states which is at most the bound stated in the theorem. We describe the construction of $\mathcal{E}'$ below, and the theorem will follow from the next two lemmas.

For a state $u \in V_G$ and a nonempty subset $\mathcal{F}$ of $\mathcal{F}_G^t(u)$, let $\Gamma(u, \mathcal{F})$ denote the set of all states v in $\mathcal{E}$ for which $\mathcal{F}_{\mathcal{E}}(v) \subseteq \mathcal{F}_G(u)$ and $\mathcal{F}_{\mathcal{E}}^t(v) = \mathcal{F}$. Whenever $\Gamma(u, \mathcal{F})$ is nonempty we designate a specific such state $v \in \Gamma(u, \mathcal{F})$ and call it $v(u, \mathcal{F})$. By Lemma 2.13, at least one $\Gamma(u, \mathcal{F})$ is nonempty.

We now define the labeled graph $\mathcal{E}'$ as follows. The states of $\mathcal{E}'$ are the pairs $(u, \mathcal{F})$ such that $\Gamma(u, \mathcal{F})$ is nonempty. We draw an edge $(u, \mathcal{F}) \xrightarrow{a} (\widehat{u}, \widehat{\mathcal{F}})$ in $\mathcal{E}'$ if and only if there is an edge $u \xrightarrow{a} \widehat{u}$ in G and an edge $v(u, \mathcal{F}) \xrightarrow{a} \widehat{v}$ in $\mathcal{E}$ for some $\widehat{v} \in \Gamma(\widehat{u}, \widehat{\mathcal{F}})$.

Lemma 7.7 *For every $\ell \leq t+1$,*

$$\mathcal{F}_{\mathcal{E}'}^{\ell}\Big((u, \mathcal{F})\Big) = \mathcal{F}_{\mathcal{E}}^{\ell}\Big(v(u, \mathcal{F})\Big) \, .$$

Proof. We prove that $\mathcal{F}^{\ell}_{\mathcal{E}'}\Big((u,\mathcal{F})\Big) \subseteq \mathcal{F}^{\ell}_{\mathcal{E}}\Big(v(u,\mathcal{F})\Big)$ by induction on ℓ. We leave the reverse inclusion (which is not used here) to the reader.

The result is immediate for $\ell = 0$. Assume now that the result is true for some fixed $\ell \leq t$. Let $w_0 w_1 \ldots w_\ell \in \mathcal{F}^{\ell+1}_{\mathcal{E}'}\Big((u,\mathcal{F})\Big)$, which implies that there is in $\mathcal{E}'$ a path of the form $(u,\mathcal{F}) \stackrel{w_0}{\to} (u_1,\mathcal{F}_1) \stackrel{w_1}{\to} (u_2,\mathcal{F}_2) \to \ldots \to (u_\ell,\mathcal{F}_\ell) \stackrel{w_\ell}{\to} (u_{\ell+1},\mathcal{F}_{\ell+1})$. By the inductive hypothesis, there is a path $v(u_1,\mathcal{F}_1) \stackrel{w_1}{\to} v_2 \stackrel{w_2}{\to} v_3 \to \ldots \to v_\ell \stackrel{w_\ell}{\to} v_{\ell+1}$ in $\mathcal{E}$. Therefore, the word $\mathbf{w} = w_1 w_2 \ldots w_\ell$ belongs to $\mathcal{F}_{\mathcal{E}}\Big(v(u_1,\mathcal{F}_1)\Big)$ and, since $\ell \leq t$, we can extend $\mathbf{w}$ to form a word $\mathbf{w}\mathbf{w}'$ of length t that belongs to $\mathcal{F}_1$. Now, by definition of the edges in $\mathcal{E}'$, there is an edge $v(u,\mathcal{F}) \stackrel{w_0}{\to} \widehat{v}$ in $\mathcal{E}$ for some $\widehat{v} \in \Gamma(u_1,\mathcal{F}_1)$. Since $\mathbf{w}\mathbf{w}' \in \mathcal{F}_1$, there is a path labeled $\mathbf{w}$ outgoing from $\widehat{v}$ in $\mathcal{E}$ and, so, there is a path labeled $w_0 w_1 \ldots w_\ell$ outgoing from $v(u,\mathcal{F})$ in $\mathcal{E}$. Hence, $\mathcal{F}^{\ell+1}_{\mathcal{E}'}\Big((u,\mathcal{F})\Big) \subseteq \mathcal{F}^{\ell+1}_{\mathcal{E}}\Big(v(u,\mathcal{F})\Big)$, as desired. □

The next lemma shows that $\mathcal{E}'$ is an (S,n)-encoder with anticipation $\leq t$.

Lemma 7.8 *The following three conditions hold:*

(a) The out-degree of each state in $\mathcal{E}'$ is n;

(b) $S(\mathcal{E}') \subseteq S$; and —

(c) $\mathcal{E}'$ has anticipation $\leq t$.

Proof. *Part (a):* It suffices to show that there is a one-to-one correspondence between the outgoing edges of $(u,\mathcal{F})$ in $\mathcal{E}'$ and those of $v(u,\mathcal{F})$ in $\mathcal{E}$. Consider the mapping Φ from outgoing edges of $(u,\mathcal{F})$ to outgoing edges of $v(u,\mathcal{F})$ defined by

$$\Phi\Big((u,\mathcal{F}) \stackrel{a}{\to} (\widehat{u},\widehat{\mathcal{F}})\Big) = \Big(v(u,\mathcal{F}) \stackrel{a}{\to} \widehat{v}\Big)$$

where $\widehat{v} \in \Gamma(\widehat{u},\widehat{\mathcal{F}})$. To see that Φ is well-defined, observe that since $\mathcal{E}$ has anticipation at most t, there cannot be two distinct edges $v(u,\mathcal{F}) \stackrel{a}{\to} \widehat{v}$ and $v(u,\mathcal{F}) \stackrel{a}{\to} \widehat{v}'$ with $\widehat{v}$ and $\widehat{v}'$ both belonging to the same $\Gamma(\widehat{u},\widehat{\mathcal{F}})$. To see that Φ is onto, first consider an outgoing edge $v(u,\mathcal{F}) \stackrel{a}{\to} \widehat{v}$ from $v(u,\mathcal{F})$, and note that since $\mathcal{F} \subseteq \mathcal{F}_G(u)$, there is in G an outgoing edge $u \stackrel{a}{\to} \widehat{u}$ for some $\widehat{u}$. Let $\widehat{\mathcal{F}} = \mathcal{F}^t_{\mathcal{E}}(\widehat{v})$. We claim that $\widehat{v} \in \Gamma(\widehat{u},\widehat{\mathcal{F}})$. Of course $\mathcal{F}^t_{\mathcal{E}}(\widehat{v}) = \widehat{\mathcal{F}}$; and since $\mathcal{F}_{\mathcal{E}}(v(u,\mathcal{F})) \subseteq \mathcal{F}_G(u)$ and G is deterministic, $\mathcal{F}_{\mathcal{E}}(\widehat{v}) \subseteq \mathcal{F}_G(\widehat{u})$. Thus, by definition of $\mathcal{E}'$ there is an edge $(u,\mathcal{F}) \stackrel{a}{\to} (\widehat{u},\widehat{\mathcal{F}})$. We thus conclude that Φ is onto. Since u and a determine $\widehat{u}$ and since $\widehat{v}$ determines $\widehat{\mathcal{F}}$, it follows that Φ is 1–1. This completes the proof of (a).

Part (b): By definition of $\mathcal{E}'$, we see that whenever there is a path $(u_0,\mathcal{F}) \stackrel{w_0}{\to} (u_1,\mathcal{F}) \stackrel{w_1}{\to} (u_2,\mathcal{F}) \to \ldots \to (u_{\ell-1},\mathcal{F}) \stackrel{w_{\ell-1}}{\to} (u_\ell,\mathcal{F})$ in $\mathcal{E}'$, there is also a path $u_0 \stackrel{w_0}{\to} u_1 \stackrel{w_1}{\to} \ldots \to u_{\ell-1} \stackrel{w_{\ell-1}}{\to} u_\ell$ in G. Thus $S(\mathcal{E}') \subseteq S(G) = S$, as desired.

Part (c): We must show that the initial edge of any path γ of length $t{+}1$ in $\mathcal{E}'$ is determined by its label $w_0 w_1 \ldots w_t$ and its initial state $(u,\mathcal{F})$. Write the initial edge of γ as:

$(u, \mathcal{F}) \stackrel{w_0}{\to} (\widehat{u}, \widehat{\mathcal{F}})$. By Lemma 7.7, there is a path in $\mathcal{E}$ with label $w_0 w_1 \dots w_t$ that begins at state $v(u, \mathcal{F})$. Since $\mathcal{E}$ has anticipation at most t, the label sequence $w_0 w_1 \dots w_t$ and $v(u, \mathcal{F})$ determine the initial edge $v(u, \mathcal{F}) \stackrel{w_0}{\to} \widehat{v}$ of this path. So, it suffices to show that u, w_0, and $\widehat{v}$ determine $\widehat{u}$ and $\widehat{\mathcal{F}}$; for then $(u, \mathcal{F})$ and $w_0 w_1 \dots w_t$ will determine the initial edge of γ.

Indeed, by definition of $\mathcal{E}'$, there must be an edge $u \stackrel{w_0}{\to} \widehat{u}$ in G such that $\widehat{v} \in \Gamma(\widehat{u}, \widehat{\mathcal{F}})$. Since G is deterministic, u and w_0 determine $\widehat{u}$. Furthermore, for any fixed $\widehat{u}$, the sets $\Gamma(\widehat{u}, \mathcal{G})$ are disjoint for distinct $\mathcal{G}$, and, so, $\widehat{v}$ determines $\widehat{\mathcal{F}}$. It follows that u, w_0, and $\widehat{v}$ determine $\widehat{u}$ and $\widehat{\mathcal{F}}$, as desired, thus proving (c). □

Now, for every state $u \in V_G$, the number of distinct nonempty subsets $\Gamma(u, \mathcal{F})$ is bounded from above by $2^{N(u,t)} - 1$. This yields the desired upper bound of Theorem 7.6 on the number of states of $\mathcal{E}'$.

It follows by Theorem 7.6 that in order to verify whether there exists an (S, n)-encoder with anticipation t, we can exhaustively check all irreducible graphs $\mathcal{E}$ with labeling over $\Sigma(S)$, with out-degree n, and with number of states $|V_{\mathcal{E}}|$ which is at most the bound of Theorem 7.6. Checking that such a labeled graph $\mathcal{E}$ is an (S, n)-encoder can be done by the following finite procedure: Construct the determinizing graph H of $\mathcal{E}$ as in Section 2.2.1. Since $\mathcal{E}$ is irreducible, the states of any irreducible sink H' of H, as subsets of $V_{\mathcal{E}}$, must contain all the states of $\mathcal{E}$. Hence, we must have $S(\mathcal{E}) = S(H')$. Then, we verify that $S(G * H') = S(H')$; to this end, it suffices, by Lemma 2.9, to check that $S(H')$ is presented by an irreducible (deterministic) component G' of $G * H'$. The equality $S(H') = S(G')$, in turn, can be checked by Theorem 2.12, using the Moore algorithm of Section 2.6.2.

Finally, testing whether $\mathcal{E}$ has anticipation $\leq t$ can be done by the efficient algorithm described in Section 2.7.2.

By the Moore co-form construction of Section 2.2.7, the existence of such an encoder implies the existence of an (S, n)-encoder with anticipation *exactly* t.

7.4.2 Upper bounds on the anticipation

Continuing the discussion of Section 7.4.1, we now obtain more tractable upper and lower bounds on the smallest attainable anticipation of (S, n)-encoders in terms of n and a deterministic presentation of the constrained system S.

Let G be a deterministic presentation of S. The anticipation of an encoder obtained by the state-splitting algorithm [ACH83] is bounded from above by the number of splitting rounds. This, in turn, yields the following result, which is, so far, the best general upper bound known for the anticipation obtained by direct application of the state-splitting algorithm.

Theorem 7.9 *Let S be a constrained system presented by a deterministic graph G and let n be a positive integer. Assume that $\mathsf{cap}(S) \geq \log n$. Then, there exists an (S, n)-encoder*

$\mathcal{E}$, obtained by the state-splitting algorithm, such that,

$$\mathcal{A}(\mathcal{E}) \leq \min_{\mathbf{x} \in \mathcal{X}(A_G, n)} \left\{ \|\mathbf{x}\|_1 - \mathsf{w}(\mathbf{x}) \right\} ,$$

where $\mathsf{w}(\mathbf{x})$ is the number of nonzero components in $\mathbf{x}$. Furthermore, if G has finite memory, then $\mathcal{E}$ is $(\mathcal{M}(G), \mathcal{A}(\mathcal{E}))$-definite.

This bound is quite poor, since it may be exponential in $|V_G|$, as is, indeed, the case for the constrained systems of Example 7.1. On the other hand, if G has finite memory, then the encoder $\mathcal{E}$ obtained by the state-splitting algorithm is guaranteed to be definite.

Now, suppose that G^t can be split fully in one round; that is, the splitting yields a labeled graph $\mathcal{E}_1$ with out-degree $\geq n^t$ at each state. By deleting excess edges, $\mathcal{E}_1$ can be made an (S^t, n^t)-encoder $\mathcal{E}_2$ with anticipation 1 over $\Sigma(S^t)$. Let $\mathcal{E}_3$ be the Moore co-form of $\mathcal{E}_2$ as in Section 2.2.7. Then $\mathcal{E}_3$ is an (S^t, n^t)-encoder with anticipation 2. If we replace the n^t outgoing edges from each state in $\mathcal{E}_3$ by an n-ary tree of depth t, we obtain an (S, n)-encoder $\mathcal{E}_4$ with anticipation $\leq 3t-1$. Therefore, we have the following.

Theorem 7.10 *Let S be a constrained system presented by a deterministic graph G and let n and t be positive integers. Suppose that G^t can be split in one round, yielding a labeled graph with minimum out-degree at least n^t. Then, there is an (S, n)-encoder with anticipation $\leq 3t-1$.*

In [Ash87b] and [Ash88], Ashley shows that for $t = O(|V_G|)$, G^t can be split in one round, yielding a labeled graph with minimum out-degree at least n^t; moreover, the splitting can be chosen to be $\mathbf{x}$-consistent with respect to *any* (A_G, n)-approximate eigenvector $\mathbf{x}$. This provides encoders with anticipation which is at most *linear* in $|V_G|$. The following theorem is a statement of Ashley's result.

Theorem 7.11 *Let S be a constrained system presented by a deterministic graph G and let n be a positive integer. Assume that $\mathsf{cap}(S) \geq \log n$. Then, there exists an (S, n)-encoder $\mathcal{E}$ such that,*

(a) when $n = \lambda(A_G)$,

$$\mathcal{A}(\mathcal{E}) \leq 9|V_G| + 6\lceil \log_n |V_G| \rceil - 1 ;$$

(b) when $n < \lambda(A_G)$,

$$\mathcal{A}(\mathcal{E}) \leq 15|V_G| + 3\lceil \log_n |V_G| \rceil - 1 .$$

Note, however, that the encoders obtained by splitting the tth power of G are typically not sliding-block decodable when $t > 1$, even when G has finite memory.

A further improvement on the upper bound of the smallest attainable anticipation is presented in [AMR95], using the stethering method which, in turn, is based on an earlier result by Adler, Goodwyn, and Weiss [AGW77] (see Chapter 6). The following result applies to the case where $n \leq \lambda(A_G) - 1$.

Theorem 7.12 *Let S be a constrained system presented by a deterministic graph G and let n be a positive integer $\leq \lambda(A_G) - 1$. Then, there is an (S, n)-encoder $\mathcal{E}$, obtained by the (punctured) stethering method, such that*

$$\mathcal{A}(\mathcal{E}) \leq 1 + \min_{\mathbf{x} \in \mathcal{X}(A_G, n+1)} \left\{ \lceil \log_{n+1} \|\mathbf{x}\|_\infty \rceil \right\} .$$

Furthermore, if G has finite memory, then $\mathcal{E}$ is $(\mathcal{M}(G), \mathcal{A}(\mathcal{E}))$-definite, and hence any tagged (S, n)-encoder based on $\mathcal{E}$ is $(\mathcal{M}(G), \mathcal{A}(\mathcal{E}))$-sliding-block decodable.

In particular, when $n \leq \lambda(A_G) - 1$, there always exists an $(A_G, n+1)$-approximate eigenvector $\mathbf{x}$ such that $\|\mathbf{x}\|_\infty \leq (n+1)^{2|V_G|}$ [Ash87a], [Ash88]. Hence, we have the following.

Corollary 7.13 *Let S be a constrained system presented by a deterministic graph G and let n be a positive integer $\leq \lambda(A_G) - 1$. Then, there is an (S, n)-encoder $\mathcal{E}$, obtained by the (punctured) stethering method, such that*

$$\mathcal{A}(\mathcal{E}) \leq 2|V_G| + 1 .$$

Furthermore, if G has finite memory, then $\mathcal{E}$ is $(\mathcal{M}(G), 2|V_G|+1)$-definite, and hence any tagged (S, n)-encoder based on $\mathcal{E}$ is $(\mathcal{M}(G), 2|V_G|+1)$-sliding block decodable.

In terms of rate $p : q$ finite-state encoders, the requirement $n \leq \lambda(A_G) - 1$ is implied by

$$\frac{p}{q} \leq \mathsf{cap}(S) - \frac{1}{2^p q \log_e 2} ;$$

namely, we need a margin between the rate and capacity which decreases exponentially with p.

Applying the stethering method on a power of G, the following result is obtained in [AMR95].

Theorem 7.14 *Let S be a constrained system presented by a deterministic graph G and let n be a positive integer smaller than $\lambda(A_G)$. Then, there is an (S, n)-encoder $\mathcal{E}$ such that*

$$\mathcal{A}(\mathcal{E}) \leq 12|V_G| - 1 .$$

Theorem 7.14 improves on Theorem 7.11, but it does not cover the case $n = \lambda(A_G)$. Also, the encoders guaranteed by Theorem 7.14 are typically not sliding-block decodable.

7.4.3 Lower bounds on the anticipation

The next theorem, taken from [MR91], provides a lower bound on the anticipation of any (S, n)-encoder. A special case of this bound appears in [Fra89].

Theorem 7.15 *Let S be a constrained system presented by a deterministic graph G and let n be a positive integer. Assume that $\mathsf{cap}(S) \geq \log n$. Then, for any (S, n)-encoder $\mathcal{E}$,*

$$\mathcal{A}(\mathcal{E}) \geq \min_{\mathbf{x} \in \mathcal{X}(A_G, n)} \left\{ \log_n \|\mathbf{x}\|_\infty \right\} .$$

Proof. The theorem trivially holds if $\mathcal{A}(\mathcal{E}) = \infty$, so we assume that $\mathcal{E}$ has finite anticipation $\mathcal{A}$. Let $\mathbf{x} = \mathbf{x}(\mathcal{E}) = (x_u)_{u \in V_G}$ be as in the proof of Theorem 7.2. We recall that by the way $\mathbf{x}$ was constructed, each nonzero component of $\mathbf{x}$ is a size of some subset $Z = T_{\mathcal{E}}(\mathbf{w}, v)$ of states in $\mathcal{E}$ which are accessible from $v \in V_{\mathcal{E}}$ by paths labeled $\mathbf{w}$.

Let $T_{\mathcal{E}}(\mathbf{w}, v)$ be such a subset whose size equals the largest component of $\mathbf{x}$ and let $\ell = \ell(\mathbf{w})$ (i.e., the length of $\mathbf{w}$). Since the out-degree of $\mathcal{E}$ is n, we have n^ℓ paths of length ℓ starting at v in $\mathcal{E}$ and, so,

$$n^\ell \geq |T_{\mathcal{E}}(\mathbf{w}, v)| = \max_{u \in V_G} x_u ,$$

implying

$$\ell \geq \min_{(y_u)_u \in \mathcal{X}(A_G, n)} \left\{ \log_n \left(\max_{u \in V_G} y_u \right) \right\} .$$

Therefore, when $\mathcal{A} \geq \ell$, we are done.

Assume now that $\ell > \mathcal{A}$. Since $\mathcal{E}$ has finite anticipation $\mathcal{A}$, the first $\ell - \mathcal{A}$ edges of any path in $\mathcal{E}$ labeled $\mathbf{w}$ are uniquely determined, once we know the initial state v. It thus follows that the paths from v to $T_{\mathcal{E}}(\mathbf{w}, v)$ labeled $\mathbf{w}$ may differ only in their last $\mathcal{A}$ edges. Hence, we can have at most $n^{\mathcal{A}}$ such paths. Recalling that the number of such paths is $|T_{\mathcal{E}}(\mathbf{w}, v)|$, we have

$$n^{\mathcal{A}} \geq |T_{\mathcal{E}}(\mathbf{w}, v)| = \max_{u \in V_G} x_u \geq \min_{(y_u)_u \in \mathcal{X}(A_G, n)} \max_{u \in V_G} y_u ,$$

as claimed. □

There is some resemblance between the lower bound of Theorem 7.15 and the upper bound of Theorem 7.12. And, indeed, there are many cases where the difference between these bounds is at most 1. Note, however, that for the constrained systems $S_k = S(G_k)$ of Example 7.1, we obtain, by Theorem 7.12, an upper bound of $1 + \frac{1}{2}|V_{G_k}|$ on the smallest anticipation of any (S_k, r)-encoder, where the lower bound of Theorem 7.15 equals 2. In fact, this lower bound is tight [AMR95].

The following bound, proved in [AMR96] is, in a way, a converse of Theorem 7.10.

Theorem 7.16 *Let S be an irreducible constrained system presented by an irreducible deterministic graph G and let n and t be positive integers. If there is an (S, n)-encoder with anticipation t, then G^t can be split in one round, yielding a graph with minimum out-degree at least n^t.*

Proof. Let $\mathcal{E}$ be an (S, n)-encoder with anticipation t, let $\Sigma = \Sigma(S)$, and let H be the determinizing graph constructed from $\mathcal{E}$ as in Section 2.2.1. Recall that each state $Z \in V_H$ is a subset $T_{\mathcal{E}}(\mathbf{w}, v)$ of states of $\mathcal{E}$ that can be reached in $\mathcal{E}$ from a given state $v \in V_{\mathcal{E}}$ by paths that generate a given word $\mathbf{w}$. Let H' be an irreducible sink of H and let $\mathbf{x} = (x_u)_{u \in V_G}$ be the nonnegative integer vector defined in the proof of Theorem 7.2:

$$x_u = \max \{ |Z| : Z \in V_{H'} \text{ and } \mathcal{F}_{H'}(Z) \subseteq \mathcal{F}_G(u) \} , \quad u \in V_G ;$$

in case there is no state $Z \in V_{H'}$ such that $\mathcal{F}_{H'}(Z) \subseteq \mathcal{F}_G(u)$, define $x_u = 0$. Then, as in the proof of Theorem 7.2, $\mathbf{x}$ is an (A_G, n)-approximate eigenvector.

Let $Z = T_{\mathcal{E}}(\mathbf{w}, v)$ be a state in H' and suppose that Z contains two distinct states, z and z', of $\mathcal{E}$. First, we claim that there is no word $\mathbf{w}'$ of length t that can be generated in $\mathcal{E}$ from both z and z'. Otherwise, we would have in $\mathcal{E}$ two paths of length $\ell(\mathbf{w}) + t$, starting at the same state v, with the same labeling $\mathbf{w}\mathbf{w}'$, that do not agree in at least one of their first $\ell(\mathbf{w})$ edges. This, however, contradicts the fact that $\mathcal{E}$ has anticipation t.

For $\mathbf{w}' \in \mathcal{F}^t_{H'}(Z)$, denote by $Z_{\mathbf{w}'}$ the terminal state in H' of a path labeled $\mathbf{w}'$ starting at Z. As we have just shown, a word $\mathbf{w}' \in \mathcal{F}^t_{H'}(Z)$ can be generated in $\mathcal{E}$ from exactly one state $z \in Z$. Therefore, the sets $\mathcal{F}^t_{\mathcal{E}}(z)$, $z \in Z$, form a partition of $\mathcal{F}^t_{H'}(Z)$. Furthermore, by the losslessness of $\mathcal{E}$, the number of paths in $\mathcal{E}$ that start at $z \in Z$ and generate $\mathbf{w}' \in \mathcal{F}^t_{\mathcal{E}}(u)$ equals $|T_{\mathcal{E}}(\mathbf{w}', z)| = |Z_{\mathbf{w}'}|$. Since $\mathcal{E}$ is an (S, n)-encoder, we conclude:

$$\sum_{\mathbf{w}' \in \mathcal{F}^t_{\mathcal{E}}(z)} |Z_{\mathbf{w}'}| = n^t \quad \text{for every } z \in Z . \tag{7.1}$$

For each state $u \in V_G$ such that $x_u \neq 0$, select some $Z = Z(u) \in V_{H'}$ such that $|Z| = x_u$ and $\mathcal{F}_{H'}(Z) \subseteq \mathcal{F}_G(u)$. Now, the partition $\{\mathcal{F}^t_{\mathcal{E}}(z) : z \in Z\}$ of $\mathcal{F}^t_{H'}(Z)$ may be regarded as a partition of $\mathcal{F}^t_G(u)$ by appending the complement $\mathcal{F}^t_G(u) \setminus \mathcal{F}^t_{H'}(Z)$ to one of the atoms $\mathcal{F}^t_{\mathcal{E}}(z)$, $z \in Z$. Since G^t is deterministic, this defines a partition $P_{G^t}(u) = \{E_{G^t}(u, z)\}_{z \in Z(u)}$ of the outgoing edges from u in G^t into $|Z(u)| = x_u$ atoms. For $\mathbf{w}' \in \mathcal{F}_{\mathcal{E}}(z)$, let u' denote the terminal state of the edge in G^t that begins at u and is labeled $\mathbf{w}'$. Now, if $\mathbf{w}'' \in \mathcal{F}_{H'}(Z_{\mathbf{w}'})$, then $\mathbf{w}'\mathbf{w}'' \in \mathcal{F}_{H'}(Z) \subseteq \mathcal{F}_G(u)$. Since G is deterministic, this implies that $\mathbf{w}'' \in \mathcal{F}_G(u')$. Thus $\mathcal{F}_{H'}(Z_{\mathbf{w}'}) \subseteq \mathcal{F}_G(u')$ and, so, $|Z_{\mathbf{w}'}| \leq x_{u'}$. This, together with Equation (7.1), shows that the splitting of G^t defined by the partition $P_{G^t}(u)$ satisfies the following inequality:

$$\sum_{e \in E_{G^t}(u,z)} x_{\tau(e)} \geq n^t \quad \text{for every } u \in V_G \text{ and } z \in Z(u) .$$

Hence, the split graph has minimum out-degree at least n^t. □

Theorem 7.16 may be regarded as a lower bound on the anticipation of an encoder. This result, together with Theorem 7.10, shows that by one round of splitting of some power of G, one can obtain an encoder whose anticipation is within a constant factor from the smallest anticipation possible.

There are examples which show that neither of the lower bounds in Theorems 7.15, 7.16 implies the other. On the other hand, when $\mathsf{cap}(S) = \log n$, we claim that for irreducible constrained systems the lower bound of Theorem 7.16 implies that of Theorem 7.15. Indeed, let t denote the bound of Theorem 7.16. Then for each state $u \in V_G$ there is a partition $\{E_{G^t}(u,i)\}_{i=1}^{x_u}$ of the outgoing edges from u in G^t such that the vector $\mathbf{x} = (x_u)_u$ is a positive (A_G, n)-approximate eigenvector and

$$\sum_{e \in E_{G^t}(u,i)} x_{t(e)} \geq n^t \quad \text{for each } (u,i)\ . \tag{7.2}$$

We now claim that (7.2) holds in our case with equality for every (u,i). Otherwise, the corresponding splitting would yield an irreducible, lossless presentation of S^t with minimum out-degree at least n^t and at least one state with out-degree greater than n^t—contradicting the equality $\mathsf{cap}(S) = \log n$.

Let $u_{\max}$ be a state in G for which $x_{u_{\max}} = \|\mathbf{x}\|_\infty$. Also, let v be a state with an outgoing edge, in G^t, to $u_{\max}$. Then any edge e from v to $u_{\max}$ in G^t belongs to some $E_{G^t}(v,i)$ and so the equality $\sum_{e \in E_{G^t}(v,i)} x_{t(e)} = n^t$ implies

$$x_{u_{\max}} \leq n^t$$

—i.e., $t \geq \log_n \|\mathbf{x}\|_\infty \geq \min_{\mathbf{y} \in \mathcal{X}(A_G,n)} \left\{ \log_n \|\mathbf{y}\|_\infty \right\}$.

We end this section by mentioning without proof the improvements on Theorems 7.15 and 7.16 that have been obtained in [Ru96] and [RuR01].

Recall that Theorem 5.10 in Section 5.6.2 provides a necessary and sufficient condition for having (S,n)-encoders with anticipation t Such a characterization also implies a lower bound on the anticipation of (S,n)-encoders: given S and n, the anticipation any (S,n)-encoder is at least the smallest nonnegative integer t for which there exists a presentation G of S and an (A_G,n)-approximate eigenvector $\mathbf{x}$ that satisfy conditions (a)–(e) of Theorem 5.10.

The following is another result obtained in [Ru96] and [RuR01].

Theorem 7.17 *Let S be an irreducible constraint, let n be a positive integer where $\mathsf{cap}(S) \geq \log n$, and let G be <u>any</u> irreducible deterministic presentation of S. Suppose there exists some irreducible (S,n)-encoder with anticipation $t < \infty$. Then there exists an (A_G,n)-approximate eigenvector $\mathbf{x}$ such that the following holds:*

(a) $\|\mathbf{x}\|_\infty \leq n^t$.

(b) For every $k = 1, 2, \ldots, t$, the states of G^k can be split in one round consistently with the (A_G^k, n^k)-approximate eigenvector $\mathbf{x}$, such that the induced approximate eigenvector $\mathbf{x}'$ satisfies $\|\mathbf{x}'\|_\infty \leq n^{t-k}$, and each of the states in G^k is split into no more than n^k states.

While Theorem 5.10 gives a necessary and sufficient condition on the existence of (S, n)-encoders with a given anticipation t, Theorem 7.17 gives only a *necessary* condition on the existence of such encoders. On the other hand, Theorem 7.17 allows to obtain a lower bound on the anticipation using *any* irreducible deterministic presentation of S—in particular the Shannon cover of S. Therefore, it will typically be easier to compute bounds using Theorem 7.17.

Note that Theorem 7.15 is equivalent to Theorem 7.17(a), while Theorem 7.16 is equivalent to Theorem 7.17(b) for the special case $k = t$. Examples in [RuR01] show that Theorem 7.17 (and hence Theorem 5.10) yields stronger bounds than these two former results.

The results in [RuR01] also imply tight lower bounds in certain practical cases. For example, it is shown therein that any rate $2 : 3$ finite-state encoder for the $(1, 7)$-RLL constraint must have anticipation at least 2, and the Weathers-Wolf encoder in 4.6 does attain this bound (and so does the encoder of Adler Coppersmith, and Hassner in [ACH83]). Similarly, any rate $1 : 2$ encoder for the $(2, 7)$-RLL constraint must have anticipation at least 3, and this bound is attained by the Franaszek encoder in Figure 4.4. A lower bound of 3 applies also to the anticipation of any rate $2 : 5$ encoder for the $(2, 18, 2)$-RLL constraint (see Figure 1.12); this bound is tight due to the constructions by Weigandt [Weig88] and Hollmann [Holl95].

7.5 Sliding-block decodability

The following is the analog of Theorem 7.6 for sliding-block decodable encoders. The special case of block decodable encoders was treated in Section 4.4.

Theorem 7.18 *Let S be an irreducible constrained system with a Shannon cover G, and let n be a positive integer and m and a be nonnegative integers. For every state u in G, let $N(u, \mathsf{a}) = |\mathcal{F}_G^{\mathsf{a}}(u)|$ and let $P(u, \mathsf{m})$ be the number of words of length m that can be generated in G by paths that terminate in state u. If there exists an (m, a)-sliding-block decodable (S, n)-encoder, then there exists such an encoder with at most $\sum_{u \in V_G} P(u, \mathsf{m})(2^{N(u,\mathsf{a})} - 1)$ states.*

Proof. The proof is similar to that of Theorem 7.6. In fact, that proof applies almost verbatim to the case of $(0, \mathsf{a})$-sliding-block decodable encoders, so we assume here that m is strictly positive. Let $\mathcal{E}$ be an irreducible (m, a)-sliding-block decodable (S, n)-encoder. Also,

let H' be an irreducible sink of the determinizing graph of $\mathcal{E}$ obtained by the construction of Section 2.2.1. By construction of H', for every path from state v to state $\widehat{v}$ in $\mathcal{E}$ that generates a word $\mathbf{w}$, there is a path in H' that generates $\mathbf{w}$, starting at a state Z and terminating in a state $\widehat{Z}$, such that $v \in Z$ and $\widehat{v} \in \widehat{Z}$. Hence, by Lemma 2.13, there also exists a path in G that generates $\mathbf{w}$, starting at a state u and terminating in a state $\widehat{u}$, such that $\mathcal{F}_{\mathcal{E}}(v) \subseteq \mathcal{F}_G(u)$ and $\mathcal{F}_{\mathcal{E}}(\widehat{v}) \subseteq \mathcal{F}_G(\widehat{u})$. It thus follows that for every state $\widehat{v}$ in $\mathcal{E}$ and a word $\mathbf{w}$ that can be generated in $\mathcal{E}$ by a path terminating in $\widehat{v}$, there is a path in G that generates $\mathbf{w}$ whose terminal state, $\widehat{u}$, satisfies $\mathcal{F}_{\mathcal{E}}(\widehat{v}) \subseteq \mathcal{F}_G(\widehat{u})$.

For a state $u \in V_G$, a word $\mathbf{w}$ of length m that can be generated in G by a path terminating in u, and a nonempty subset $\mathcal{F} \subseteq \mathcal{F}_G^{\mathsf{a}}(u)$, we define $\Gamma(u, \mathbf{w}, \mathcal{F})$ to be the set of all states v in $\mathcal{E}$ which are terminal states of paths in $\mathcal{E}$ that generate $\mathbf{w}$ and such that $\mathcal{F}_{\mathcal{E}}(v) \subseteq \mathcal{F}_G(u)$ and $\mathcal{F}_{\mathcal{E}}^{\mathsf{a}}(v) = \mathcal{F}$. Note that each state of $\mathcal{E}$ is contained in some set $\Gamma(u, \mathbf{w}, \mathcal{F})$ and, so, at least one such set is nonempty.

A tagged (S, n)-encoder $\mathcal{E}'$ is now defined as follows. In each nonempty set $\Gamma(u, \mathbf{w}, \mathcal{F})$, we designate a state of $\mathcal{E}$ and call it $v(u, \mathbf{w}, \mathcal{F})$. The states of $\mathcal{E}'$ are triples $(u, \mathbf{w}, \mathcal{F})$ for which $\Gamma(u, \mathbf{w}, \mathcal{F})$ is nonempty.

Let u and $\widehat{u}$ be states in G and let $\mathbf{w} = w_1 w_2 \dots w_{\mathsf{m}}$ and $\widehat{\mathbf{w}} = \widehat{w}_1 \widehat{w}_2 \dots \widehat{w}_{\mathsf{m}}$ be two words that can be generated by paths in G that terminate in u and $\widehat{u}$, respectively. If $\Gamma(u, \mathbf{w}, \mathcal{F})$ and $\Gamma(\widehat{u}, \widehat{\mathbf{w}}, \widehat{\mathcal{F}})$ are nonempty, then we draw a tagged edge $(u, \mathbf{w}, \mathcal{F}) \stackrel{s/b}{\to} (\widehat{u}, \widehat{\mathbf{w}}, \widehat{\mathcal{F}})$ in $\mathcal{E}'$ if and only if the following four conditions hold:

(a) $\widehat{w}_j = w_{j+1}$ for $j = 1, 2, \dots, \mathsf{m}-1$;

(b) $b = \widehat{w}_{\mathsf{m}}$;

(c) there is a tagged edge $v(u, \mathbf{w}, \mathcal{F}) \stackrel{s/b}{\to} \widehat{v}$ in $\mathcal{E}$ for some $\widehat{v} \in \Gamma(\widehat{u}, \widehat{\mathbf{w}}, \widehat{\mathcal{F}})$;

(d) there is an edge $u \stackrel{b}{\to} \widehat{u}$ in G.

By the proof of Theorem 7.6, it follows that $\mathcal{E}'$ is, indeed, an (S, n)-encoder and that $\mathcal{F}_{\mathcal{E}'}^{\mathsf{a}+1}\big((u, \mathbf{w}, \mathcal{F})\big) = \mathcal{F}_{\mathcal{E}}^{\mathsf{a}+1}\big(v(u, \mathbf{w}, \mathcal{F})\big)$. Furthermore, it can be shown by induction that, for every $\ell \leq \mathsf{m}$, the paths of length ℓ in $\mathcal{E}'$ that terminate in $(u, w_1 w_2 \dots w_{\mathsf{m}}, \mathcal{F})$, all have the same labeling $w_{\mathsf{m}-\ell+1} w_{\mathsf{m}-\ell+2} \dots w_{\mathsf{m}}$. Since the 'outgoing picture'—including tagging—from state $(u, \mathbf{w}, \mathcal{F})$ in $\mathcal{E}'$ is the same as that from state $v(u, \mathbf{w}, \mathcal{F})$ in $\mathcal{E}$, it follows that $\mathcal{E}'$ is (m, a)-sliding-block decodable.

The upper bound on $|V_{\mathcal{E}'}|$ is now obtained by counting the number of distinct states $(u, \mathbf{w}, \mathcal{F})$. □

The upper bound on the number of states in Theorem 7.18 is doubly-exponential in the decoding look-ahead a. In [AM95], a stronger result is obtained where the upper bound on the number of states is singly-exponential. It is still open whether such an improvement is

possible also for the doubly-exponential bound of Theorem 7.6.

The following bound is easily verified.

Proposition 7.19 *Let $\mathcal{E}$ be an irreducible (m, a)-sliding-block decodable encoder. Then, $\mathsf{a} \geq \mathcal{A}(\mathcal{E})$.*

Hence, we can apply the lower bounds on the anticipation which were presented in Section 7.4.3, to obtain lower bounds on the attainable look-ahead of sliding-block decodable encoders (but these do not give lower bounds on the decoding window length, $\mathsf{m} + \mathsf{a} + 1$, since m may be negative). On the other hand, Theorems 7.9 and 7.12 and Corollary 7.13 provide upper bounds on the look-ahead of encoders obtained by constructions that yield sliding-block decodable encoders for finite-type constrained systems.

We remark that Theorem 7.18 implies upper bounds on the the size of encoders which are sliding-block decodable also when m is negative: simply apply the theorem with $\mathsf{m} = 0$.

And finally we note that, at least for finite-type constrained systems, Hollmann [Holl96] has given a procedure for deciding if there exists a sliding block decodable (S, n)-encoder with a given window length; here, the window length $L = \mathsf{m} + \mathsf{a} + 1$, rather than m and a, is specified. Even for $L = 1$, this is a non-trivial problem, because one must consider the possibility that $\mathsf{a} = -\mathsf{m}$ may be arbitrarily large.

7.6 Gate complexity and time–space complexity

In this section, we discuss the gate complexity and time-space complexity of some of the encoding schemes that were mentioned in the previous sections. We start with the time-space complexity criterion, assuming that the encoders are to be implemented as a program on a random-access machine (RAM) [AHU74, Ch. 1]. The results on gate complexity will then follow by known results in complexity theory.

We define an *encoding scheme* as a function $(G, q, n) \mapsto \mathcal{E}(G, q, n)$, that maps a deterministic graph G and integers q and n into an $(S(G^q), n)$-encoder $\mathcal{E}(G, q, n)$. The state-splitting algorithm of [ACH83], the method described by Ashley in [Ash88], and the stethering method of [AMR95] are examples of encoding schemes.

For a given encoding scheme $(G, q, n) \mapsto \mathcal{E}(G, q, n)$, we can formalize the *encoding problem* as follows: We are to write an encoding program P on a RAM; an input instance to P consists of the following entries:

- a deterministic graph G over an alphabet Σ,
- an integer q,

- an integer $n \leq \lambda(A_G^q)$,
- a state u of $\mathcal{E}(G, q, n)$,
- an input tag $s \in \{0, 1, \ldots, n-1\}$.

For any input instance, the program P computes an output q-block over Σ and the next state of the tagged $(S(G^q), n)$-encoder $\mathcal{E}(G, q, n)$, given we are at state u in $\mathcal{E}(G, q, n)$ and the current input tag is s. Note that in order to perform its function, the program P does not necessarily have to generate the whole graph presentation of $\mathcal{E}(G, q, n)$.

We denote by $\mathrm{Poly}(\cdot)$ a fixed arbitrary multinomial, whose coefficients are absolute constants, independent of its arguments.

The following was proved in [AMR95] for the stethering coding scheme and for a variation of Ashley's construction [Ash88].

Theorem 7.20 *There exists an encoding scheme $(G, q, n) \mapsto \mathcal{E}(G, q, n)$ (such as the one presented in [Ash88] or [AMR95]), for which there is an encoding program P on a RAM that solves the encoding problem in time complexity which is at most* $\mathrm{Poly}\Big(|V_G|, q, \log|\Sigma|\Big)$.

In particular, if we now fix G, q, and n, we obtain an encoding program that simulates $\mathcal{E}(G, q, n)$ with a polynomial time and space complexity.

Theorem 7.20 applies to the constructions covered in Theorems 7.11, 7.12, and 7.14. In contrast, it is not known yet whether a polynomial encoder can be obtained by a direct application of the state-splitting algorithm.

For a positive integer ℓ, denote by I_ℓ the set of all possible inputs to P of size ℓ, according to some standard representation of the input. Now, if the time complexity of P on each element of I_ℓ is polynomial in ℓ, then for any input size ℓ, there exists a circuit C_ℓ with $\mathrm{Poly}(\ell) = \mathrm{Poly}\Big(|V_G|, q, \log|\Sigma|\Big)$ gates that implements P for inputs in I_ℓ. Furthermore, such circuits C_ℓ are 'uniform' in the sense that there is a program on a RAM that generates the layouts of C_ℓ in time complexity which is $\mathrm{Poly}(\ell)$. This is a consequence of a known result on the equivalence between polynomial circuit complexity and polynomial RAM complexity of decision problems [ST93, Theorem 2.3].

By Theorem 7.20, it thus follows that we can have such a polynomial circuit at hand for both Ashley's construction and the stethering construction. We summarize this in the following theorem.

Theorem 7.21 *For every constrained system S over an alphabet Σ, presented by a deterministic graph G, and for any positive integers q and $n \leq \lambda(A_G^q)$, there exists an (S^q, n)-encoder that can be implemented by a circuit consisting of* $\mathrm{Poly}\Big(|V_G|, q, \log|\Sigma|\Big)$ *gates and*

$O(|V_G| \log n)$ memory bit-cells. Furthermore, there exists a program on a RAM that generates the layout of such an implementation in polynomial-time.

Theorems 7.20 and 7.21 apply also to the decoding complexity of the corresponding encoders.

Problems

Problem 7.1 Prove Theorem 7.3 by modifying the end of the proof of Theorem 7.2.

Problem 7.2 Verify the assertion of Example 7.2.

Problem 7.3 Verify the assertion of Example 7.3.

Problem 7.4 Let S be the constrained system presented by the graph G in Figure 2.24. Is there a positive integer ℓ for which there exists a deterministic $(S^{2\ell}, 2^{\ell})$-encoder? If yes, construct such an encoder; otherwise, explain.

Problem 7.5 Let S be the constrained system presented by the graph G in Figure 7.2.

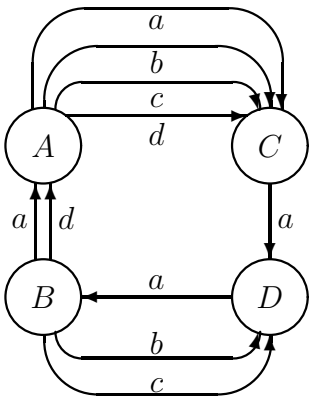


Figure 7.2: Graph G for Problem 7.5.

1. Compute the capacity of S.
2. Compute an $(A_G, 2)$-approximate eigenvector in which the largest entry is the smallest possible.
3. For every positive integer ℓ, determine the smallest anticipation of any $(S^{\ell}, 2^{\ell})$-encoder.

Problem 7.6 Let S be the constrained system presented by the graph G in Figure 5.27.

1. Find the smallest anticipation possible of any $(S, 2)$-encoder.
2. Find the smallest number of states of any $(S^2, 4)$-encoder.

Chapter 8

Error Correction and Concatenation

The preceding chapters have addressed properties of constrained systems and construction of encoders which encode user data sequences into constrained sequences. In practice, these constrained codes, as applied in data recording systems, may be viewed as part of the modulation/demodulation process of input/output signals of the system. Most systems require the use of some form of error-correction coding (ECC) in addition to constrained coding of the input signal or symbol sequence. It is therefore natural to investigate the interplay between these two forms of coding and the possibilities for efficiently combining their functions into a single coding operation, in analogy to the coded-modulation techniques now in wide use in data transmission.

In this chapter, we give, in the first four sections, a very brief introduction to error-correcting linear block codes (the rough idea of error-correction coding was discussed briefly in Section 1.3). This includes description of the basic properties of linear block codes, finite fields, and culminates with a discussion of the celebrated Reed-Solomon codes.

In Section 8.5, we consider three schemes for concatenating ECC codes and constrained codes. The third scheme involves a data compression idea. This idea is formalized in Section 8.6, and the performance in this context of some specific compression codes is given in Section 8.7. Then, in Section 8.8, we show how a dual version of state-splitting ideas from Chapter 5 can be used to give a general construction of compression codes that are useful in this context.

In Chapter 9, we will consider codes which have combined error-correction and constrained coding properties—in particular codes that are designed to handle error mechanisms that arise in magnetic recording.

8.1 Error-Correction Coding

An (n, M) *(block) code* over a finite alphabet F is a nonempty subset $\mathcal{C}$ of size M of F^{n}. The elements of the alphabet are referred to as *symbols*. The parameter n is called the *code length* and M is the *code size*. The *dimension* (or *information length*) of $\mathcal{C}$ is defined by $\mathsf{k} = \log_{|F|} M$, and the *rate* of $\mathcal{C}$ is $R = \mathsf{k}/\mathsf{n}$. The elements of a code are called *codewords*.

In addition to the parameters, code length and code size, there is a third parameter, called the minimum distance, which gives a rough sense of the robustness of the code to channel noise. For this we need to define the following notion of distance.

The *Hamming distance* between two words $\mathbf{x}, \mathbf{y} \in F^{\mathsf{n}}$ is the number of coordinates in which $\mathbf{x}$ and $\mathbf{y}$ differ. We denote the Hamming distance by $\Delta(\mathbf{x}, \mathbf{y})$.

It is easy to verify that Hamming distance satisfies the following properties of a metric for every three words $\mathbf{x}, \mathbf{y}, \mathbf{z} \in F^{\mathsf{n}}$:

- $\Delta(\mathbf{x}, \mathbf{y}) \geq 0$, with equality if and only if $\mathbf{x} = \mathbf{y}$.
- Symmetry: $\Delta(\mathbf{x}, \mathbf{y}) = \Delta(\mathbf{y}, \mathbf{x})$.
- The triangle inequality: $\Delta(\mathbf{x}, \mathbf{y}) \leq \Delta(\mathbf{x}, \mathbf{z}) + \Delta(\mathbf{z}, \mathbf{y})$.

Let $\mathcal{C}$ be an (n, M) code over F with $M > 1$. The *minimum distance* of $\mathcal{C}$ is the minimum Hamming distance between any two distinct codewords of $\mathcal{C}$. That is, the minimum distance d is given by

$$\mathsf{d} = \min_{\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C} \,:\, \mathbf{c}_1 \neq \mathbf{c}_2} \Delta(\mathbf{c}_1, \mathbf{c}_2) \; .$$

An (n, M) code with minimum distance d is called an $(\mathsf{n}, M, \mathsf{d})$ *code*.

Example 8.1 The binary $(3, 2, 3)$ *repetition code* is the code $\{000, 111\}$ over $F = \{0, 1\}$. The dimension of the code is $\log_2 2 = 1$ and its rate is $1/3$. □

Example 8.2 The binary $(3, 4, 2)$ *parity code* is the code $\{000, 011, 101, 110\}$ over $F = \{0, 1\}$. The dimension is $\log_2 4 = 2$ and the code rate is $2/3$. □

Given an (n, M, d) code $\mathcal{C}$ over F, let $\mathbf{c} \in \mathcal{C}$ be a codeword transmitted over a noisy channel, and let $\mathbf{y} \in F^{\mathsf{n}}$ be the received word. By an *error* we mean the event of changing an entry in the codeword $\mathbf{c}$. The number of errors equals $\Delta(\mathbf{y}, \mathbf{c})$, and the error locations are the indices of the entries in which $\mathbf{c}$ and $\mathbf{y}$ differ. The task of error correction is to recover the error locations and the error values.

The following result shows that for any code with minimum distance d, there is a procedure that can correct up to $\lfloor (\mathsf{d}-1)/2 \rfloor$ many errors.

Proposition 8.1 *Let $\mathcal{C}$ be a block code over F with code length n and minimum distance d. For a received word $\mathbf{y}$, let $\mathcal{D}(\mathbf{y})$ denote the codeword in $\mathcal{C}$ that is closest (with respect to Hamming distance) to $\mathbf{y}$. If codeword $\mathbf{c}$ is transmitted, $\mathbf{y}$ is received, and $\Delta(\mathbf{y},\mathbf{c}) \leq (\mathsf{d}-1)/2$ then $\mathcal{D}(\mathbf{y}) = \mathbf{c}$.*

Proof. Suppose to the contrary that $\mathbf{c}' = \mathcal{D}(\mathbf{y}) \neq \mathbf{c}$. By definition,

$$\Delta(\mathbf{y},\mathbf{c}') \leq \Delta(\mathbf{y},\mathbf{c}) \leq (d-1)/2 \; .$$

So, by the triangle inequality,

$$\mathsf{d} \leq \Delta(\mathbf{c},\mathbf{c}') \leq \Delta(\mathbf{y},\mathbf{c}) + \Delta(\mathbf{y},\mathbf{c}') \leq \mathsf{d}-1 \; ,$$

which is a contradiction. □

8.2 Linear Codes

Most of the theory of ECC has focused on linear codes. Such a code is defined as a finite-dimensional vector space over a finite field. We assume that the reader is familiar, from a course in elementary linear algebra, with the notion of a vector space or linear space. But since such a course does not necessarily treat finite fields, we give a brief introduction to finite fields in Section 8.3; in particular, we give in Section 8.3 a construction of the finite field GF(q). For a more thorough introduction to ECC, we refer the reader to any of the excellent textbooks on the subject, such as [LinCo83], [Mcl77] or [Wic95].

8.2.1 Definition

An $(\mathsf{n}, M, \mathsf{d})$ code $\mathcal{C}$ over a finite field $F = \mathrm{GF}(q)$ is called *linear* if $\mathcal{C}$ is a linear sub-space of F^{n} over F, namely, for every $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}$ and $a_1, a_2 \in F$ we have $a_1\mathbf{c}_1 + a_2\mathbf{c}_2 \in \mathcal{C}$.

The *dimension* of a linear (n, M, d) code $\mathcal{C}$ over F is the dimension of $\mathcal{C}$ as a linear sub-space of F^{n} over F. If k is the dimension of $\mathcal{C}$, then we say that $\mathcal{C}$ is a linear $[\mathsf{n}, \mathsf{k}, \mathsf{d}]$ code over F. The difference $\mathsf{n} - \mathsf{k}$ is called the *redundancy* of $\mathcal{C}$.

Every basis of a linear $[\mathsf{n}, \mathsf{k}, \mathsf{d}]$ code $\mathcal{C}$ over $F = \mathrm{GF}(q)$ contains k codewords, the linear combinations of which are distinct and generate the whole set $\mathcal{C}$. Therefore, $|\mathcal{C}| = M = q^{\mathsf{k}}$ and the code rate is $R = (\log_q M)/\mathsf{n} = \mathsf{k}/\mathsf{n}$.

Words $\mathbf{y} = y_1 y_2 \ldots y_{\mathsf{n}}$ over a field F—in particular, codewords of a linear $[\mathsf{n}, \mathsf{k}, \mathsf{d}]$ code over F—will sometimes be denoted by $(y_1 \; y_2 \; \ldots \; y_{\mathsf{n}})$, to emphasize that they are elements of a vector space F^{n}.

Example 8.3 The $(3, 4, 2)$ parity code over GF(2) is a linear $[3, 2, 2]$ code since it is spanned by (1 0 1) and (0 1 1). □

The *Hamming weight* of $\mathbf{e} \in F^{\mathsf{n}}$ is the number of nonzero entries in $\mathbf{e}$. We denote the Hamming weight by $\mathsf{w}(\mathbf{e})$. Note that for every two words $\mathbf{x}, \mathbf{y} \in F^{\mathsf{n}}$,

$$\Delta(\mathbf{x}, \mathbf{y}) = \mathsf{w}(\mathbf{y} - \mathbf{x}) = \Delta(\mathbf{y} - \mathbf{x}, \mathbf{0}),$$

where $\mathbf{0}$ denotes the (all-)zero codeword, which is an element of any linear code (since a linear space always contains the zero vector). The following result characterizes the minimum distance of a linear code in terms of its minimum Hamming weight.

Proposition 8.2 *Let $\mathcal{C}$ be a linear $[\mathsf{n}, \mathsf{k}, \mathsf{d}]$ code over F. Then*

$$\mathsf{d} = \min_{\mathbf{c} \in \mathcal{C} \setminus \{\mathbf{0}\}} \mathsf{w}(\mathbf{c}) \, .$$

Proof. Since $\mathcal{C}$ is linear,

$$\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C} \quad \Longrightarrow \quad \mathbf{c}_1 - \mathbf{c}_2 \in \mathcal{C} \, .$$

Now, $\Delta(\mathbf{c}_1, \mathbf{c}_2) = \mathsf{w}(\mathbf{c}_1 - \mathbf{c}_2)$ and, so,

$$\mathsf{d} = \min_{\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C} \, : \, \mathbf{c}_1 \neq \mathbf{c}_2} \Delta(\mathbf{c}_1, \mathbf{c}_2) = \min_{\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C} \, : \, \mathbf{c}_1 \neq \mathbf{c}_2} \mathsf{w}(\mathbf{c}_1 - \mathbf{c}_2) = \min_{\mathbf{c} \in \mathcal{C} \setminus \{\mathbf{0}\}} \mathsf{w}(\mathbf{c}) \, .$$

□

8.2.2 Generator Matrix

A *generator matrix* of a linear $[\mathsf{n}, \mathsf{k}, \mathsf{d}]$ code over F is a $\mathsf{k} \times \mathsf{n}$ matrix whose rows form a basis of the code.

Example 8.4 The matrix

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

is a generator matrix of the $[3, 2, 2]$ parity code over GF(2), and so is the matrix

$$\hat{G} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \, .$$

In general, the $[\mathsf{n}, \mathsf{n}-1, 2]$ *parity code* over a field F is defined as the code with a generator matrix

$$G = \left(\begin{array}{c|c} & -1 \\ I & -1 \\ & \vdots \\ & -1 \end{array} \right) ,$$

where I is the $(\mathsf{n}-1) \times (\mathsf{n}-1)$ identity matrix. □

Example 8.5 The $(3, 2, 3)$ repetition code over GF(2) is a linear $[3, 1, 3]$ code generated by

$$G = (\ 1 \quad 1 \quad 1 \ ) .$$

In general, the $[\mathsf{n}, 1, \mathsf{n}]$ *repetition code* over a field F is defined as the code with a generator matrix

$$G = (\ 1 \quad 1 \quad \ldots \quad 1 \ ) .$$

□

Let $\mathcal{C}$ be a linear $[\mathsf{n}, \mathsf{k}, \mathsf{d}]$ code over F and G be a generator matrix of $\mathcal{C}$. We can encode information words into codewords of $\mathcal{C}$ by regarding the former as vectors $\mathbf{u} \in F^{\mathsf{k}}$ and using a mapping $F^{\mathsf{k}} \to \mathcal{C}$ defined by

$$\mathbf{u} \mapsto \mathbf{u}G .$$

Since $\mathrm{rank}(G) = \mathsf{k}$, we can apply elementary operations to the rows of G to obtain a $\mathsf{k} \times \mathsf{k}$ identity matrix as a sub-matrix of G.

A $\mathsf{k} \times \mathsf{n}$ generator matrix is called *systematic* if it has the form

$$\left(\ I \,\middle|\, A \ \right) ,$$

where I is a $\mathsf{k} \times \mathsf{k}$ identity matrix and A is a $\mathsf{k} \times (\mathsf{n}-\mathsf{k})$ matrix.

Not always does a code $\mathcal{C}$ have a systematic generator matrix. However, we can always permute the code coordinates to obtain an *equivalent* (although different) code $\hat{\mathcal{C}}$ for which the first k columns of any generator matrix are linearly independent, in which case the code has a systematic generator matrix. The code $\hat{\mathcal{C}}$ has the same length, dimension, and minimum distance as the original code $\mathcal{C}$.

When using a systematic generator matrix $G = (\, I \,|\, A \,)$ for encoding, the mapping $\mathbf{u} \mapsto \mathbf{u}G$ takes the form $\mathbf{u} \mapsto (\, \mathbf{u} \,|\, \mathbf{u}A \,)$; that is, the information vector is part of the encoded codeword.

8.2.3 Parity-check matrix

Let $\mathcal{C}$ be a linear $[\mathsf{n}, \mathsf{k}, \mathsf{d}]$ code over F. A *parity-check matrix* of $\mathcal{C}$ is an $r \times \mathsf{n}$ matrix H over F such that for every $\mathbf{c} \in F^{\mathsf{n}}$,

$$\mathbf{c} \in \mathcal{C} \qquad \Longleftrightarrow \qquad H\mathbf{c}^\top = \mathbf{0} \; .$$

In other words, the code $\mathcal{C}$ is the (right) kernel, $\ker(H)$, of H in F^{n}. We thus have

$$\operatorname{rank}(H) = \mathsf{n} - \dim \ker(H) = \mathsf{n} - \mathsf{k} \; .$$

So, in (the most common) case where the rows of H are linearly independent we have $r = \mathsf{n} - \mathsf{k}$.

Let G be a $\mathsf{k} \times \mathsf{n}$ generator matrix of $\mathcal{C}$. The rows of G span $\ker(H)$ and, in particular,

$$HG^\top = 0 \qquad \Longrightarrow \qquad GH^\top = 0 \; .$$

Also,

$$\dim \ker(G) = \mathsf{n} - \operatorname{rank}(G) = \mathsf{n} - \mathsf{k} \; .$$

Hence, the rows of H span $\ker(G)$. So, a parity-check matrix of a linear code can be computed by finding a basis of the kernel of a generator matrix of the code.

In the special case where G is a systematic matrix $(\, I \,|\, A \,)$, we can take the $(\mathsf{n}-\mathsf{k}) \times \mathsf{n}$ matrix $H = (\, -A^\top \,|\, I \,)$ as a parity-check matrix.

Example 8.6 The matrix

$$\begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}$$

is a parity-check matrix of the $[\mathsf{n}, \mathsf{n}-1, 2]$ parity code over a field F, and

$$\left(\begin{array}{c|c} & -1 \\ I & -1 \\ & \vdots \\ & -1 \end{array}\right)$$

is a parity-check matrix of the $[\mathsf{n}, 1, \mathsf{n}]$ repetition code over F. □

Example 8.7 The linear $[7, 4, 3]$ *Hamming code* over GF(2) is defined by the parity-check matrix

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \; .$$

A corresponding generator matrix is given by

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} .$$

One can check exhaustively that the minimum distance of this code is indeed 3. □

The following theorem provides a characterization of the minimum distance of a linear code through any parity-check matrix of the code.

Theorem 8.3 *Let H be a parity-check matrix of a linear code $\mathcal{C} \neq \{\mathbf{0}\}$. The minimum distance of $\mathcal{C}$ is the largest integer d such that every set of $d-1$ columns in H is linearly independent.*

Proof. Write $H = (\,\mathbf{h}_1\ \mathbf{h}_2\ \ldots\ \mathbf{h}_{\mathsf{n}}\,)$ and let $\mathbf{c} = (c_1\ c_2\ \ldots\ c_{\mathsf{n}})$ be a codeword in $\mathcal{C}$ with Hamming weight $t > 0$. Let $J \subseteq \{1, 2, \ldots, \mathsf{n}\}$ be the set of $|J| = t$ indexes of the nonzero entries in $\mathbf{c}$. By $H\mathbf{c}^\top = \mathbf{0}$ we have

$$\sum_{j \in J} c_j \mathbf{h}_j = \mathbf{0} ,$$

namely, the t columns of H that are indexed by J are linearly dependent.

Conversely, every set of t linearly dependent columns in H corresponds to at least one nonzero codeword $\mathbf{c} \in \mathcal{C}$ with $\mathsf{w}(\mathbf{c}) \leq t$.

Given d as defined in the theorem, it follows that no nonzero codeword in $\mathcal{C}$ has Hamming weight less than d, but there is at least one codeword in $\mathcal{C}$ whose Hamming weight is d. □

Example 8.8 For an integer $m > 1$, the $[2^m{-}1, 2^m{-}1{-}m, 3]$ Hamming code over $F = \mathrm{GF}(2)$ is defined by an $m \times (2^m{-}1)$ parity-check matrix H whose columns range over all the nonzero elements of F^m. Every two columns in H are linearly independent and, so, the minimum distance of the code is at least 3. In fact, the minimum distance is exactly 3, since there are three dependent columns, e.g., $(0\ \ldots\ 0\ 0\ 1)^\top$, $(0\ \ldots\ 0\ 1\ 0)^\top$, and $(0\ \ldots\ 0\ 1\ 1)^\top$. □

8.3 Introduction to Finite Fields

Roughly speaking, a finite field is a finite set of elements in which notions of addition and multiplication are defined subject to the following conditions:

- The usual rules (associative, distributive and commutative) of arithmetic hold.
- Each element has an additive inverse (i.e., a "negative" of itself) and a multiplicative inverse (i.e., a "reciprocal"); in other words, you can subtract and divide elements of the field and still remain in the field.

The simplest non-trivial field is $GF(2)$, which consists of just two elements: $\{0, 1\}$ with modulo-2 arithmetic, i.e.,

$$\begin{array}{rcl} 0+0 & = & 1+1 = 0 \\ 0+1 & = & 1+0 = 1 \\ 0\cdot 0 & = & 0 \\ 1\cdot 1 & = & 1 \\ 0\cdot 1 & = & 1\cdot 0 = 0 \end{array}$$

More generally, for a prime number p, the *prime field* GF(p) consists of the elements $\{0, 1, \ldots, p-1\}$, with arithmetic modulo p. It can be shown that GF(p) contains a *primitive element*, defined as an element α such that the powers of α exhaust all nonzero elements of the field.

Example 8.9 In GF(7) we have

$$2 \cdot 4 = 3 \cdot 5 = 6 \cdot 6 = 1 \cdot 1 = 1 \, .$$

The elements 3 and 5 are primitive elements:

$$\begin{array}{rcl} 3^0 = 1 = 5^0 & \qquad & 3^3 = 6 = 5^3 \\ 3^1 = 3 = 5^5 & & 3^4 = 4 = 5^2 \\ 3^2 = 2 = 5^4 & & 3^5 = 5 = 5^1 \end{array}$$

and in fact they are the only primitive elements. □

It turns out that the size of any finite field is a power of a prime number $q = p^h$. Such a field is denoted GF(q) and is defined as the set of all polynomials of degree less than h in an indeterminate x with coefficients in GF(p). Addition is then defined as the usual polynomial addition (using arithmetic modulo p).

In order to define multiplication, we make use of an *irreducible polynomial*, $P(x)$ of degree exactly h, with coefficients in GF(p), i.e., a polynomial that cannot be factored nontrivially over GF(p). It can be shown that such a polynomial exists for each prime p and integer h.

Example 8.10 The following are all the irreducible polynomials over GF(2) with degree at most 4:

degree 1: $\quad x,\ x+1$

degree 2: $x^2 + x + 1$
degree 3: $x^3 + x + 1,\ x^3 + x^2 + 1$
degree 4: $x^4 + x + 1,\ x^4 + x^3 + 1,\ x^4 + x^3 + x^2 + x + 1$

□

We then define multiplication in GF(q) by multiplication of polynomials followed by reduction by $P(x)$: precisely, to find the product of $a(x)$ and $b(x)$, first form the ordinary product $s(x) = a(x)b(x)$ and then compute the remainder of $s(x)$ when divided by $P(x)$. Irreducbility of $P(x)$ is required in order to guarantee that each nonzero element of GF(q) has a multiplicative inverse.

Example 8.11 Let $F = \mathrm{GF}(2)$ and $P(x) = x^3 + x + 1$. We construct the field $\mathrm{GF}(2^3)$ as the set of polynomials over F of degree less than 3, resulting in Table 8.1, where the elements of the field are written as polynomials in the indeterminate x. The third column in the table

000	0	0
001	1	1
010	x	x
011	$x+1$	x^3
100	x^2	x^2
101	x^2+1	x^6
110	x^2+x	x^4
111	x^2+x+1	x^5

Table 8.1: The field $\mathrm{GF}(2^3)$

expresses each nonzero element in the field as a power of the monomial $x = 0\cdot 1 + 1\cdot x + 0\cdot x^2$. Indeed,

$$\begin{aligned}
x^3 &\equiv x+1 \pmod{P(x)}\,,\\
x^4 &\equiv x\cdot x^3 \equiv x(x+1) \equiv x^2+x \pmod{P(x)}\,,\\
x^5 &\equiv x\cdot x^4 \equiv x(x^2+x) \equiv x^3+x^2 \equiv x^2+x+1 \pmod{P(x)}\,,\\
x^6 &\equiv x\cdot x^5 \equiv x(x^2+x+1) \equiv x^3+x^2+x \equiv x^2+1 \pmod{P(x)}\,,
\end{aligned}$$

and

$$x^7 \equiv x\cdot x^6 \equiv x(x^2+1) \equiv x^3+x \equiv 1 \pmod{P(x)}\,.$$

□

Just as for prime fields, any finite field GF(q) contains a primitive element. In the preceding example, the monomial x is primitive. In general the monomial x may not be

a primitive element of the field. However, it turns out that there is always a choice of the irreducible polynomial $P(x)$ such that the monomial x is a primitive element. Such a polynomial is called a *primitive polynomial*.

Finally, we mention that elements of GF(q) can be viewed as h-dimensional vectors over the field GF(p) and this way define a vector space of dimension h over GF(p). In this way, we may regard the symbols of a code over GF(2^8) as bytes.

8.4 The Singleton bound and Reed-Solomon codes

Theorem 8.4 (The Singleton bound) *For any* $(\mathsf{n}, M, \mathsf{d})$ *code over an alphabet of size* q*,*

$$\mathsf{d} \leq \mathsf{n} - \lceil \log_q M \rceil + 1 \; .$$

Proof. Let $\ell = \lceil \log_q M \rceil - 1$. Since $q^\ell < M$, there must be at least two codewords that agree in their first ℓ coordinates. Hence, $\mathsf{d} \leq \mathsf{n} - \ell$. □

For a linear $[\mathsf{n}, \mathsf{k}, \mathsf{d}]$ code over GF(q) the Singleton bound becomes

$$\mathsf{d} \leq \mathsf{n} - \mathsf{k} + 1 \; .$$

This can also be seen from a parity-check matrix of the code: since the rank of a parity-check matrix is $\mathsf{n}-\mathsf{k}$, every set (so at least one set) of $\mathsf{n}-\mathsf{k}+1$ columns in that matrix is linearly dependent.

For linear codes, the Singleton bound can also be obtained by considering a systematic generator matrix of the code: the Hamming weight of each row is at most $\mathsf{n}-\mathsf{k}+1$.

A code is called *maximum distance separable (MDS)* if it attains the Singleton bound with equality. Note that a linear MDS code can correct a certain number, e, of symbols in error within each codeword if its redundancy, $\mathsf{n}-\mathsf{k}$, satisfies $\mathsf{n}-\mathsf{k} = 2e$; in other words, for a linear MDS code, two bytes of redundancy are sufficient (and in fact necessary by the Singleton bound) to correct each error. So, a code with two bytes of redundancy can correct one error in each codeword, and a code with four bytes of redundancy can correct two errors in each codeword, and so on.

The following are simple examples of linear MDS codes over $F = \mathrm{GF}(q)$:

- The whole space F^{n}, which is a linear $[\mathsf{n}, \mathsf{n}, 1]$ code over F.
- The $[\mathsf{n}, \mathsf{n}-1, 2]$ parity code over F.
- The $[\mathsf{n}, 1, \mathsf{n}]$ repetition code over F.

The following family of MDS codes is among the most widely used error-correction codes today.

Let $\alpha_1, \alpha_2, \ldots, \alpha_{\mathsf{n}}$ be distinct elements of $F = \mathrm{GF}(q)$. A *Reed-Solomon code* over F is a linear $[\mathsf{n}, \mathsf{k}, \mathsf{d}]$ code with the parity-check matrix

$$H = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_{\mathsf{n}} \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_{\mathsf{n}}^2 \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{\mathsf{n}-\mathsf{k}-1} & \alpha_2^{\mathsf{n}-\mathsf{k}-1} & \ldots & \alpha_{\mathsf{n}}^{\mathsf{n}-\mathsf{k}-1} \end{pmatrix}.$$

This construction requires $\mathsf{n} \leq q$.

Proposition 8.5 *Every Reed-Solomon code is MDS.*

Proof. Every $(\mathsf{n}-\mathsf{k}) \times (\mathsf{n}-\mathsf{k})$ sub-matrix of H has a *Vandermonde* form

$$B = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \beta_1 & \beta_2 & \ldots & \beta_{\mathsf{n}-\mathsf{k}} \\ \beta_1^2 & \beta_2^2 & \ldots & \beta_{\mathsf{n}-\mathsf{k}}^2 \\ \vdots & \vdots & \vdots & \vdots \\ \beta_1^{\mathsf{n}-\mathsf{k}-1} & \beta_2^{\mathsf{n}-\mathsf{k}-1} & \ldots & \beta_{\mathsf{n}-\mathsf{k}}^{\mathsf{n}-\mathsf{k}-1} \end{pmatrix},$$

where $\beta_1, \beta_2, \ldots, \beta_{\mathsf{n}-\mathsf{k}}$ are distinct elements of the field. Now, the determinant of B is given by

$$\det(B) = \prod_{j>i} (\beta_j - \beta_i)$$

and, therefore, $\det(B) \neq 0$ and B is nonsingular. It follows that every set of $\mathsf{n}-\mathsf{k}$ columns in H is linearly independent and, so, $d \geq \mathsf{n} - \mathsf{k} + 1$. □

Finally, we mention that it is possible to extend Reed-Solomon codes to length $q+1$; such a code is called an *extended Reed-Solomon code*, and these codes are MDS as well [MacS77, Chapter 10].

8.5 Concatenation of ECC and constrained codes

As mentioned in Section 1.3, an ECC encoder and a constrained encoder are usually concatenated as follows: messages are first passed through an error-correction encoder and then a constrained encoder before being transmitted across the channel. At the receiver, data is

decoded via the constrained decoder and then the error-correction decoder. This scheme, illustrated in Figure 8.1(a), is called *standard concatenation*.

Typically, the ECC code is a Reed-Solomon code, owing to their excellent error-correction properties and excellent decoding properties. In addition, in order to provide for protection against bursty errors, the ECC scheme involves *m-way-interleaving* of codewords, which we explain as follows. Suppose that the ECC encoder generates m consecutive codewords:

$$\begin{array}{rcl} \mathbf{c}^1 & = & x_1^1 x_2^1 \dots x_{\mathsf{n}}^1 \\ \dots & = & \dots \\ \mathbf{c}^m & = & x_1^m x_2^m \dots x_{\mathsf{n}}^m \end{array} .$$

Instead of recording these codewords one after another, we first transmit the first symbol of each codeword:

$$x_1^1 x_1^2 \dots x_1^m$$

and then the second symbol of each codeword

$$x_2^1 x_2^2 \dots x_2^m,$$

etc. In this way, the errors in a contiguous burst are spread among several codewords, thereby reducing the likelihood of overwhelming the error correcting capability of the code.

It is natural for the constrained encoder to be nearer the channel since its purpose is to produce sequences that are designed to pass through the channel with little likelihood of corruption. On the other hand, standard concatenation requires that the constrained decoder severely limit error propagation, and this precludes the possibility of constrained encoders based on long block lengths.

In *modified concatenation* (sometimes called reversed concatenation) the order of concatenation is reversed as shown in Figure 8.1(b). This idea is due to Bliss [Bli81] and Mansuripur [Man91] and more recently to Immink [Imm97] and Fan and Calderbank [FC98]; the latter demonstrated some advantages of modified concatenation even with constrained encoders based on relatively short block lengths. A user data sequence $\mathbf{u}$ is first encoded via a high rate encoder E1 into a constrained sequence $\mathbf{w}$. In order to achieve the high rate (very close to capacity), long block lengths must be used. If $\mathbf{w}$ is then transmitted across a noisy (binary) channel, a small burst error or even a single isolated error in the received sequence $\hat{\mathbf{w}}$ could affect much or possibly all of the decoded sequence $\hat{\mathbf{u}}$, yielding enormous error propagation. To avoid this, error correction is incorporated and used to correct all errors before decoding $\hat{\mathbf{w}}$. This is done by computing a sequence $\mathbf{r}$ of parity symbols on $\mathbf{w}$, and then encoding $\mathbf{r}$ into a constrained sequence $\mathbf{y}$ via a second constrained encoder E2 which is less efficient (namely, has a lower rate) than the first encoder; yet it operates on shorter blocks. Both constrained sequences $\mathbf{w}$ and $\mathbf{y}$ are then transmitted across the noisy channel. The sequence of parity symbols should be chosen to allow correction of a prescribed typical channel error event in $\mathbf{w}$, such as bursts of errors up to a certain length. The decoder

attempts to recover $\mathbf{u}$ from the possibly corrupted versions $\hat{\mathbf{w}}$ of $\mathbf{w}$ and $\hat{\mathbf{y}}$ of $\mathbf{y}$. Since the constrained encoder E2 uses short block lengths, it is presumably subject to very little error propagation. Then the decoded version, $\hat{\mathbf{r}}$, of $\hat{\mathbf{y}}$ can be used to correct $\hat{\mathbf{w}}$, without fear of error propagation (i.e., the error events in $\hat{\mathbf{w}}\hat{\mathbf{r}}$ look roughly like the raw channel error events in $\hat{\mathbf{w}}\hat{\mathbf{y}}$). In this way, $\mathbf{w}$ is recovered error-free; decoding $\mathbf{w}$ via the first constrained decoder recovers $\mathbf{u}$ error-free.

One of Immink's key contributions in [Imm97] was the realization that $\mathbf{w}$, being an encoded version of $\mathbf{u}$, is longer than $\mathbf{u}$, so that it may be necessary to increase the number of parity symbols $\mathbf{r}$ for the error-correcting code to achieve the same performance. In addition, for long bursts, the effect of a burst of channel errors is magnified relative to the standard concatenation scheme, since the bursts are not first decoded by the constrained decoder. Immink's solution, shown in Figure 8.1(c), to this problem was to *compress* the sequence $\mathbf{w}$ in a lossless (one-to-one) manner into a sequence $\mathbf{s}$, and then compute the sequence of parity symbols $\mathbf{r}$ based on $\mathbf{s}$; for instance, $\mathbf{s}$ (respectively, $\mathbf{r}$) could be the information (respectively, parity) portions of a Reed-Solomon code. So the parity sequence $\mathbf{r}$, and therefore also the E2-encoded sequence $\mathbf{y}$, can be made shorter, thereby lowering the overhead of the error-correction scheme. At the channel output, the received sequence $\hat{\mathbf{w}}$ is compressed to a sequence $\hat{\mathbf{s}}$, and the ECC decoder recovers $\mathbf{s}$ from $\hat{\mathbf{r}}$ and $\hat{\mathbf{s}}$. Then the decompressor recovers $\mathbf{w}$, and the constrained decoder D1 recovers $\mathbf{u}$.

At one extreme, one could compress $\mathbf{w}$ back to $\mathbf{u}$ (in which case $\mathbf{s}$ would be the same as $\mathbf{u}$). But then a small channel error in $\hat{\mathbf{w}}$ could corrupt all of $\hat{\mathbf{s}}$ *before* error correction. Instead, the compression scheme will guarantee that such a channel error can corrupt only a limited number of bytes in $\hat{\mathbf{s}}$.

In constrained coding, as we have presented it in this text, unconstrained user sequences are encoded, in a lossless manner, into a constrained sequences; after being passed through a channel the constrained sequences are decoded to unconstrained user sequences.

In lossless data compression, the roles of encoder and decoder are reversed: constrained sequences from a source are encoded into unconstrained sequences where no distortion is permitted upon decompression. This duality between constrained coding and lossless data compression has been noted by several authors (see for example [Ari90], [Ker91], [MLT83], [TLM27]). In these works, data compression techniques such as arithmetic coding have been applied to constrained coding. In the remainder of this chapter, we apply constrained coding to lossless data compression, to obtain compressors that can be used in the scheme of Figure 8.1(c).

8.6 Block and sliding-block compressible codes

A *rate $p:q$ block-compressor* for a constrained system S is simply a one-to-one mapping from the set S_q, of words in length q in S, into the set of unconstrained binary words of length p (*the reader should not confuse S_q with the q-th power system, S^q*). Clearly, a necessary and sufficient condition for the existence of a rate $p:q$ block compressor for S is:

$$|S_q| \leq 2^p \ . \tag{8.1}$$

Immink gives in [Imm97] two simple examples: a rate $8:11$ block compressor for the $(1,12)$-RLL constraint and a rate $8:13$ block compressor for the $(2,15)$-RLL constraint. For $p=8$, these values of q are optimal: a simple computation reveals that condition (8.1) would be violated for any rate $8:12$ block compressor for the $(1,12)$-RLL constraint and any rate $8:14$ block compressor for the $(2,15)$-RLL constraint.

Clearly, $p=8$ is a good choice owing to the availability of high performance, high efficiency, off-the-shelf Reed-Solomon codes. But allowing other values of p can give added flexibility in the choice of compression schemes (provided that p and the symbol alphabet of the ECC are somewhat compatible). Clearly, it is desirable to have a small compression rate p/q, and smaller compression rates can be achieved by larger block lengths p and q. But the capacity of the constraint imposes a lower bound on the compression rates, as we show next.

Since $|S_{qm}| \leq |S_q|^m$ for any choice of positive integers q and m, it follows that

$$\mathsf{cap}(S) = \lim_{m\to\infty} (1/(qm)) \cdot \log |S_{qm}| \leq (1/q) \cdot \log |S_q| \ ;$$

that is, the limit in the definition of capacity is taken over elements each of which is an upper bound on $\mathsf{cap}(S)$. Combining this with (8.1) yields

$$\mathsf{cap}(S) \leq (1/q) \cdot \log |S_q| \leq p/q \ . \tag{8.2}$$

Thus, to obtain compression rates p/q close to capacity, we need to take q (and hence p) sufficiently large so that $(1/q)\cdot\log|S_q|$ is close enough to capacity. This approach has several drawbacks. First, such schemes can be rather complex. Secondly, if the typical burst error length is short relative to q, then the compression code may actually expand the burst. Third, even if the typical burst error is of length comparable to q, it may be aligned so as to affect two or more consecutive q-codewords, and therefore two or more consecutive unconstrained p-blocks; this "edge-effect" can counteract the benefits of using compression codes.

The foregoing discussion leads us to consider a more general class of compression codes, in particular *lossless sliding-block compression codes*. Such a code consists of a *compressor* and an *expanding coder* (in short, *excoder*), which acts as a "decompressor". The compressor is a sliding-block code from sequences of q-codewords of S to unconstrained sequences of p-blocks over $\{0,1\}$; that is, a q-codeword $\mathbf{w}$ is compressed into a p-block $\mathbf{s}$ as a time-invariant

function of $\mathbf{w}$ and perhaps some m preceding and a upcoming q-codewords. The excoder, on the other hand, will have the form of a finite-state machine. Just as in conventional constrained coding, the *sliding-block window length* is defined as the sum $\mathsf{m} + \mathsf{a} + 1$.

We next present a precise definition of the model of compressors and excoders considered in this chapter. For the sake of convenience, we start with excoders and then base the definition of compressors on that of the matching excoders. Let S be a constraint over an alphabet Σ, let Φ be a set of size n, and let m and a be nonnegative integers. An (m, a)-*sliding-block compressible* (S, n)-*excoder* is a graph $\mathcal{E}$ in which the edges are labeled by elements of Σ and, in addition, each edge is endowed by a *tag* from Φ so that the following holds:

(X1) the outgoing edges from each state are assigned distinct tags from Φ; in particular, each state will have at most n outgoing edges;

(X2) S is contained in the constraint (over Σ) that is presented by $\mathcal{E}$; and—

(X3) if $\mathbf{w}$ is a word in $S_{\mathsf{m}+\mathsf{a}+1}$, and $e_{-\mathsf{m}}e_{-\mathsf{m}+1}\dots e_0\dots e_{\mathsf{a}}$ and $e'_{-\mathsf{m}}e'_{-\mathsf{m}+1}\dots e'_0\dots e'_{\mathsf{a}}$ are sequences of edges that form two paths in $\mathcal{E}$ both labeled by $\mathbf{w}$, then the tags of e_0 and e'_0 agree.

Often we will apply this definition to a constrained system S whose alphabet consists of q-codewords in another constraint S', in which case $S_{\mathsf{m}+\mathsf{a}+1}$ will consist of words of length $(\mathsf{m} + \mathsf{a} + 1)q$ in S' (see the definition below of a rate $p : q$ excoder).

The definition of an (m, a)-sliding-block compressible (S, n)-excoder $\mathcal{E}$ bears similarity to that of a tagged (m, a)-sliding-block decodable (S, n)-encoder: the main difference is in the containment relationship between S and the constraint presented by $\mathcal{E}$. Here, $\mathcal{E}$ must generate every word in S and, in addition, it may generate words that are not in S; note, however, that condition (X3) applies only to those paths in $\mathcal{E}$ that generate words in S.

One could replace condition (X3) by a weaker condition that would correspond to the lossless condition for encoders given in Section 4.1. But this may not be adequate for the compression desired in the scheme of Figure 8.1(c).

Condition (X3) induces a mapping $\mathcal{C} : S_{\mathsf{m}+\mathsf{a}+1} \to \Phi$, which, in turn, defines the *sliding-block compressor* of $\mathcal{E}$ as follows. For any positive integer ℓ, the compressor maps every word

$$\mathbf{w} = w_{-\mathsf{m}}w_{-\mathsf{m}+1}\dots w_0w_1\dots w_{\ell-1}w_\ell\dots w_{\ell+\mathsf{a}-1}$$

in $S_{\mathsf{m}+\mathsf{a}+\ell}$ into a pair $(v, \mathbf{s})$, where v is an *initial state* in $\mathcal{E}$, which can be the initial state of any path in $\mathcal{E}$ labeled by $w_0w_1\dots w_{\ell+\mathsf{a}-1}$, and

$$\mathbf{s} = s_0s_1\dots s_{\ell-1}$$

is a tag sequence in Φ^ℓ defined by

$$s_i = \mathcal{C}(w_{i-\mathsf{m}} w_{i-\mathsf{m}+1} \dots w_i \dots w_{i+\mathsf{a}}) \ , \quad 0 \le i < \ell \ .$$

Observe that the excoder can recover the sub-word $w_0 w_1 \dots w_{\ell-1}$ of $\mathbf{w}$ by reading the labels along the (unique) path of length ℓ in $\mathcal{E}$ that starts at v and is tagged by $\mathbf{s}$.

In the examples, given later in this chapter, $\mathcal{E}$ will be (m, a)-*definite on* S: if $\mathbf{w}$ is a word in $S_{\mathsf{m}+\mathsf{a}+1}$, and $e_{-\mathsf{m}} e_{-\mathsf{m}+1} \dots e_0 \dots e_{\mathsf{a}}$ and $e'_{-\mathsf{m}} e'_{-\mathsf{m}+1} \dots e'_0 \dots e'_{\mathsf{a}}$ are two paths in $\mathcal{E}$ both generating $\mathbf{w}$, then $e_0 = e'_0$; note that (m, a)-definiteness on S is stronger than condition (X3).

Next, we show how rate $p : q$ compression codes can be described through (S, n)-excoders and compressors. Let S be a constrained system that is presented by a labeled graph G. We now define an (m, a)-*sliding-block compressible excoder for* S *at rate* $p : q$ to be an (m, a)-sliding-block compressible $(S^q, 2^p)$-excoder. The tag set Φ is taken as $\{0,1\}^p$, namely, the set of all possible values of any p-block. So, a rate $p : q$ excoder for S maps p-blocks into q-codewords in a state-dependent manner; the respective compressor, in turn, maps a sequence of q-codewords into a sequence of p-blocks, where the i-th q-codeword is compressed into a p-block through a mapping applied to the i-th q-codeword, as well as m preceding and a upcoming q-codewords. A *block excoder for* S *at rate* $p : q$ is a rate $p : q$ excoder for S with one state. Note that the corresponding sliding-block compressor is simply a block compressor, as defined at the beginning of this section.

We next establish a necessary condition for the existence of (S, n)-excoders.

Proposition 8.6 *Let S be a constraint. There is an (m, a)-sliding-block compressible (S, n)-excoder only if $\mathsf{cap}(S) \le \log n$.*

Proof. Let $\mathcal{E}$ be an (m, a)-sliding-block compressible (S, n)-excoder and let V and Φ be the set of states and the set of tags of $\mathcal{E}$, respectively. Suppose that S' is an irreducible constraint contained in S such that $\mathsf{cap}(S') = \mathsf{cap}(S)$.

Let $\mathbf{w}$ be a word in S'_ℓ (and hence in S_ℓ). Since S' is irreducible, the word $\mathbf{w}$ can be extended to a word $\mathbf{w}'\mathbf{w}\mathbf{w}'' \in S'_{\mathsf{m}+\mathsf{a}+\ell}$. The compressor of $\mathcal{E}$ maps the word $\mathbf{w}'\mathbf{w}\mathbf{w}''$ into a pair $(v, \mathbf{s})$, where $v \in V$ and $\mathbf{s} \in \Phi^\ell$, and the (unique) path in $\mathcal{E}$ that starts at v and is tagged by $\mathbf{s}$ generates the word $\mathbf{w}$. We have thus obtained through the compressor a one-to-one mapping from S'_ℓ into $V \times \Phi^\ell$; so,

$$|S'_\ell| \le |V| \cdot n^\ell \ .$$

The result follows by taking the limit as $\ell \to \infty$ and using the definition of capacity. □

Proposition 8.6 implies that there is a sliding-block compressible excoder for S at rate $p : q$ only if $\mathsf{cap}(S^q) \le p$ or, equivalently, $\mathsf{cap}(S) \le p/q$. The latter inequality is exactly the reverse of Shannon's bound on the rate of (conventional) constrained encoders. The bound (8.2) amounts to the special case of Proposition 8.6 for block excoders.

The next result, which we establish in Section 8.8, states that for finite-type constraints the condition in Proposition 8.6 is not only necessary, but also sufficient for the existence of sliding-block compressible excoders.

Proposition 8.7 *Let S be a finite-type constrained system with memory m, and let n be a positive integer such that $\mathsf{cap}(S) \leq \log n$. Then there is an (m, a)-sliding-block compressible (S, n)-excoder; in fact, this excoder is (m, a)-definite on S.*

Finally, we make some remarks regarding the inclusion of the initial state in the information conveyed from the compressor to the excoder. The cost of transmitting this initial state is quite minimal. Typically, there will be a small number of states and the number of bits required to represent a state is only the logarithm of that number. Also, in the scheme of Figure 8.1(c), one does not really need to expand the entire tag sequence after error correction: since channel decoding takes place after error correction and since the channel decoder has full knowledge of the received (uncompressed) constrained sequence, it need only re-expand the corrected p-blocks in the compressed tag sequence; since a previously corrected portion of the received sequence is very likely to contain state information (for example if the excoder $\mathcal{E}$ is (m, a)-definite on S), no extra state information may be needed at all. Another alternative is to simply compress only those constrained sequences that can be generated from one fixed state of the excoder. When incorporated into Immink's scheme this would entail a loss in capacity, but the loss is very small since the block lengths are so long. A third solution, which is applicable to (m, a)-definite excoders, is to include the first $\mathsf{m}+\mathsf{a}+1$ q-codewords of the (non-compressed) constrained sequence in the bit stream that is protected by the ECC. Thus, the ECC decoder of the receiving end will reconstruct the correct prefix of the constrained sequence, thereby allowing us to recover the state information.

8.7 Application to burst correction

When used in conjunction with the scheme in Figure 8.1(c), there are a number of factors that may affect the choice of a compressor-excoder pair such as the complexity of the compression and decompression and the error-propagation associated with the application of the compression on the receiving end. In particular, we are concerned with how compressors handle raw channel bursts, and their suitability for use with a symbol-based ECC.

The compressor is applied to the channel bit sequence immediately after the channel, so that a benefit of using a low-rate excoder is that the length of a raw channel burst will be roughly decreased by the compression factor p/q, when the length of the burst is long (relative to q). On the other hand, edge effects in the use of a compressor can expand the error length, and this error propagation may dominate for short bursts. In addition, for sliding-block compressible excoders, the sliding-block window length will also extend the

burst. Ultimately, the choice of a compressor-excoder pair involves a balance of these four factors:

1. compression rate $p : q$;
2. edge effects (how many extra p-blocks are affected by the phasing of a burst);
3. effect of the sliding-block window length $\mathsf{m} + \mathsf{a} + 1$ (i.e., how many extra p-blocks may be affected by each error); and —
4. compatibility between the block length p and the symbol alphabet of the ECC.

We consider here the effect of a channel burst of length L bits on the maximum number of affected p-blocks. Hereafter, by a length of a burst in a sequence over a given symbol alphabet we mean the number of symbols between (and including) the first and last erroneous symbols. Our computation will mainly concentrate on the simplified model where any error in the q-codeword will result in an entirely erroneous p-block upon compression, although in practice it might be possible to mitigate this effect by a proper tag assignment to the excoder (see Section 8.8.3).

The maximum number of q-codewords (including the edge effect) that can be affected by a channel burst of length L bits is either $\lfloor (L-1)/q \rfloor + 1$ or $\lceil (L-1)/q \rceil + 1$, depending on the phasing within a q-codeword where the channel burst starts. For (m, a)-sliding-block compressible excoders, the effect of the memory and anticipation is to expand the number of affected p-blocks by $\mathsf{m}+\mathsf{a}$, so that we get a maximum of

$$N = N(L) = \lceil (L-1)/q \rceil + \mathsf{m}+\mathsf{a}+1 \tag{8.3}$$

affected p-blocks.

Next, we need to translate from a number of erroneous p-blocks to a corresponding number of symbol errors for the ECC. Let the symbol alphabet of the ECC in Figure 8.1(c) be the finite field $\mathrm{GF}(2^B)$; namely, the sequence of p-blocks is regarded as a long bit-stream and sub-divided into non-overlapping blocks of length B bits, each such block being a symbol of the ECC and regarded as a "B-bit byte." We make the assumption that the boundaries between p-blocks align with the boundaries between ECC symbols as often as possible, in particular, every $(pB)/\gcd(p, B)$ bits. We can then calculate the maximum number of ECC symbols that are in error due to a channel burst of length L bits.

Consider our basic unit to be of size $\gcd(p, B)$ bits, so that we are starting with a burst of length $(Np)/\gcd(p, B)$, and looking for the maximum number of affected blocks of length $B/\gcd(p, B)$. This is analogous to finding the maximum number of q-codewords affected by a burst of channel bits, and we obtain the following expression for the number of ECC symbols in error as a function of L and B:

$$D(L, B) = \left\lceil \frac{(Np)/\gcd(p, B) - 1}{B/\gcd(p, B)} \right\rceil + 1 = \left\lceil \frac{Np - \gcd(p, B)}{B} \right\rceil + 1 \ .$$

Putting this together with (8.3) yields

$$D(L,B) = \left\lceil \frac{(\lceil (L-1)/q \rceil + \mathsf{m}{+}\mathsf{a}{+}1)p - \gcd(p,B)}{B} \right\rceil + 1 \ .$$

Example 8.12 Consider a $(0,1)$-sliding-block compressible excoder for the $(2,\infty)$-RLL constraint at rate $4:7$ (such as the excoder that we will present in Example 8.16 in Section 8.8.2). Table 8.2 contains the respective values of $D(L,B)$ for $L = 40$ and $B = 4,5,6,7,8$.

B	$D(40,B)$	ν	ρ	κ
4	8	544	64	840
5	8	1,320	(80)	2,170
6	6	2,340	72	3,969
7	6	5,418	(84)	9,334
8	5	10,280	80	17,850

Table 8.2: Parameters of an ECC used for burst lengths of up to 40 bits in conjunction with a $(0,1)$-sliding-block compressible excoder for the $(2,\infty)$-RLL constraint at rate $4:7$.

The last three columns in Table 8.2 show the parameters of an ECC that consists of $D(L,B)$-way interleaving of an extended Reed-Solomon code of length 2^B+1 over GF(2^B). The overall block length (in bits) of this ECC scheme equals $\nu = \nu(L,B) = (2^B+1)\cdot B \cdot D(L,B)$; the values of ν are listed in the third column of the table. Since Reed-Solomon codes are maximum distance separable, each symbol error of GF(2^B) can be corrected using two redundancy symbols. Therefore, the total number of redundant symbols is $2D(L,B)$. The redundancy (in bits) of the coding scheme thus equals $\rho = \rho(L,B) = 2\cdot B\cdot D(L,B)$; this is the length of "Parity" in Figure 8.1(c). The values of ρ are listed in the fourth column of the table, where numbers in parentheses indicate that smaller redundancy values (and larger ECC block lengths) can be obtained by using a larger value of B.

A block of ν bits at the output of the ECC encoder in Figure 8.1(c) corresponds to $\nu-\rho$ bits at the input of that encoder; those bits, in turn, correspond to $\kappa = \kappa(L,B) = \lfloor (\nu-\rho)\cdot q/p \rfloor$ channel bits at the input of the lossless compressor. The values of κ are listed in the fifth column of Table 8.2. Clearly, a smaller value of the ratio ρ/κ means a smaller overhead introduced by the ECC (when combined with the compression). □

Using the ECC scheme and the notations of Example 8.12, we can fix a number, κ_0, of channel bits and compute for each (maximal) burst length L the respective redundancy ρ obtained by optimizing over all values of B for which $\kappa(L,B) \geq \kappa_0$. As an example, consider a message of 512 user bytes (4,096 bits) in Figure 8.1(c). Selecting the rate $256:466$ code

of [Imm97] as the constrained encoder D1, the message is mapped into $7,456$ channel bits. Figure 8.2 shows the best redundancy values attained for $\kappa_0 = 7,456$ and $L \leq 300$ using a $(0,1)$-sliding-block compressible excoder at rate $4:7$ for the $(2,\infty)$-RLL constraint. The figure shows the redundancy values also for two other block excoders for this constraint at rates $8:13$ and $4:6$; note that $8:13$ is the rate of the excoder presented in [Imm97]. Thus we see that for longer bursts, the sliding-block excoder requires less redundancy due to a better compression of the burst length, in spite of the longer sliding-block window. (We point out that Figure 8.2 is the same also for $\kappa_0 = 7,427$, which is the number we get when we divide $4,096$ by the capacity ($\approx .5515$) of the $(2,\infty)$-RLL constraint.)

8.8 Constructing sliding-block compressible excoders

Our construction of excoders (and respective compressors) follows the lines of the state-splitting algorithm in Figure 5.9 for constructing finite-state encoders.

Recall that in the conventional state-splitting algorithm, we begin with a graph presentation G of the given constraint S. Typically, we assume that G is deterministic, such as the graph $G_{2,\infty}$ in Figure 8.3 which presents the $(2,\infty)$-RLL constraint. The state-splitting algorithm generates an encoder for S through a sequence of state splittings, which are guided by approximate eigenvectors.

The algorithm we present here is very similar; the main difference is that instead of approximate eigenvectors, we will use what we call *super-vectors*. Such a vector will lead us through a sequence of state-splitting operations beginning with the graph G and ending with a graph H with out-degree at most n (i.e., each state has at most n outgoing edges). Then one assigns to the edges of H tags taken from the tag alphabet Φ (of size n) such that at each state all outgoing edges have distinct tags; typically, $n = 2^p$ and $\Phi = \{0,1\}^p$. The tagged version of H will be the (S,n)-excoder $\mathcal{E}$.

In order to guarantee the sliding-block compressibility of $\mathcal{E}$, we will assume that S has finite memory m, in which case we take G as a (necessarily deterministic) graph presentation of S that has memory m. When state splitting is applied to this graph, we are guaranteed to end up with an excoder $\mathcal{E}$ which is (m,a)-definite on S, where a is the number of rounds of out-splittings; in particular, $\mathcal{E}$ will be (m,a)-sliding-block compressible.

8.8.1 Super-vectors

As is the case with approximate eigenvectors, super-vectors will be computed using the adjacency matrix A_G of the graph presentation G of S. Recall that for a deterministic presentation, the adjacency matrix can be used to compute the capacity of S; namely $\mathsf{cap}(S) = \log \lambda(A_G)$, where $\lambda(A_G)$ is the largest (absolute value of any) eigenvalue of A_G.

Recall also that $(A_G)^q = A_{G^q}$ and, so, $(\lambda(A_G))^q = \lambda(A_{G^q})$.

Example 8.13 The adjacency matrix of the graph $G_{2,\infty}$ in Figure 8.3 is

$$A_{G_{2,\infty}} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} ,$$

and the capacity of the $(2,\infty)$-RLL constraint is given by $\log \lambda(A_{G_{2,\infty}}) \approx \log 1.4656 \approx .5515$. By Proposition 8.6, we will be able to construct a sliding-block compressible excoder for the $(2,\infty)$-RLL constraint only if its rate $p : q$ satisfies $(\lambda(A_{G_{2,\infty}}))^q \leq 2^p$. In the running example of this section, we will choose $p = 4$ and $q = 7$, in which case $(\lambda(A_G))^q \approx 14.5227 \leq 16 = 2^p$.□

Let A be a nonnegative integer square matrix and let n be a positive integer; e.g., $A = A_G^q$ and $n = 2^p$. An (A, n)-*super-vector* is a nonnegative integer vector $\mathbf{x}$, not identically zero, such that

$$A\mathbf{x} \leq n\mathbf{x} .$$

Note that approximate eigenvectors (from Section 5.2.1) are defined the same way except that the inequality is reversed (with the benefit of hindsight, approximate eigenvectors should probably have been called "sub-vectors").

By a straightforward modification of the corresponding proof for approximate eigenvectors (Theorem 5.4), it follows that for any nonnegative integer square matrix A, there exists an (A, n)-super-vector if and only if $\lambda(A) \leq n$. The proof suggests ways of finding such a vector, but a simpler algorithm is presented in Figure 8.4; this algorithm is the analogue of the Franaszek Algorithm for finding approximate eigenvectors.

Next, we summarize the properties of this algorithm—in particular, the uniqueness of a minimum (A, n)-super-vector $\mathbf{x}^*$ and the fact that the algorithm always converges to $\mathbf{x}^*$. For convenience we assume that A is irreducible.

Proposition 8.8 *Let A be an irreducible integer matrix and let n be a positive integer with $\lambda(A) \leq n$. Then the following holds:*

(a) Any (A, n)-super-vector is strictly positive.

(b) If $\mathbf{x}$ and $\mathbf{x}'$ are (A, n)-super-vectors, then the vector defined by $\mathbf{z} = \min\{\mathbf{x}, \mathbf{x}'\}$ is also an (A, n)-super-vector (here, $\min\{\cdot, \cdot\}$ is applied componentwise).

(c) There is a unique minimum (A, n)-super-vector, i.e., a unique (A, n)-super-vector $\mathbf{x}^*$ such that for any (A, n)-super-vector $\mathbf{x}$ we have $\mathbf{x}^* \leq \mathbf{x}$.

(d) The algorithm in Figure 8.4 eventually halts and returns the vector $\mathbf{x}^*$.

Parts (b) and (d) of the preceding result are analogous to to the corresponding results for approximate eigenvectors (given in Proposition 5.5). On the other hand, for approximate eigenvectors, there is no analogous uniqueness result and there is no clear choice of initial vector for the algorithm.

Proof of Proposition 8.8. (a) Let $\mathbf{x} = (x_u)_u$ be an (A, n)-super-vector. If some entry x_u is 0 then, according to the inequality $A\mathbf{x} \leq n\mathbf{x}$, we have $x_v = 0$ for all v such that $A_{u,v} \neq 0$. By irreducibility of A, this implies that $\mathbf{x}$ is identically zero, contrary to the definition of an (A, n)-super-vector.

(b) Since A is nonnegative, $A\mathbf{z} \leq A\mathbf{x} \leq n\mathbf{x}$ and $A\mathbf{z} \leq A\mathbf{x}' \leq n\mathbf{x}'$; so, $A\mathbf{z} \leq n\mathbf{z}$. Clearly $\mathbf{z}$ has only integer entries. By (a), $\mathbf{x}$ and $\mathbf{x}'$ are strictly positive, and thus so is $\mathbf{z}$; in particular, it is not identically zero.

(c) Let $\mathbf{x}^*$ be obtained by taking the componentwise minimum of all (A, n)-super-vectors. By (b), $\mathbf{x}^*$ is an (A, n)-super-vector, and it is clearly the unique minimum such vector.

(d) Denote by $\mathbf{x}^i$ the value of $\mathbf{x}$ after the i-th iteration of the `while` loop in Figure 8.4, with $\mathbf{x}^0 = (1\ 1\ \ldots\ 1)^\top$. We show by induction on i that $\mathbf{x}^i \leq \mathbf{x}^*$. The induction base $i = 0$ follows from (a). Now, suppose that $\mathbf{x}^i \leq \mathbf{x}^*$ for some i. Since A is nonnegative, we have

$$\frac{1}{n}A\mathbf{x}^i \leq \frac{1}{n}A\mathbf{x}^* \leq \mathbf{x}^* \ .$$

Therefore, $\mathbf{x}^{i+1} = \max\left\{\left\lceil \frac{1}{n}A\mathbf{x}^i \right\rceil, \mathbf{x}^i\right\} \leq \mathbf{x}^*$, thereby establishing the induction step.

Next we verify that the algorithm halts. Observe that $\mathbf{x}^0 \leq \mathbf{x}^1 \leq \mathbf{x}^2 \leq \ldots \leq \mathbf{x}^*$ and that $\mathbf{x}^i$ are integer vectors. Now, if the algorithm did not halt, there had to be an index i for which $\mathbf{x}^{i+1} = \mathbf{x}^i$. However, that would imply $\left\lceil \frac{1}{n}A\mathbf{x}^i \right\rceil \leq \mathbf{x}^i$ (in which case $\mathbf{x}^i$ would in fact be a super-vector), so the algorithm had to halt in the i-th iteration.

We therefore conclude that the algorithm halts and returns an (A, n)-super-vector $\mathbf{x} \leq \mathbf{x}^*$; in fact, we must have $\mathbf{x} = \mathbf{x}^*$, since $\mathbf{x}^*$ is the unique minimum (A, n)-super-vector. □

Example 8.14 The adjacency matrix of the graph $G = G^7_{2,\infty}$ is given by

$$A_G = A^7_{G_{2,\infty}} = \begin{pmatrix} 3 & 2 & 4 \\ 4 & 3 & 6 \\ 6 & 4 & 9 \end{pmatrix} \ .$$

Now,

$$\begin{aligned}
\mathbf{x}^0 &= (1\ 1\ 1)^\top \ , \\
\mathbf{x}^1 &= \max\left\{\left\lceil \tfrac{1}{16}A_G(1\ 1\ 1)^\top \right\rceil, (1\ 1\ 1)^\top\right\} = \max\left\{(1\ 1\ 2)^\top, (1\ 1\ 1)^\top\right\} = (1\ 1\ 2)^\top \ , \\
\mathbf{x}^2 &= \max\left\{\left\lceil \tfrac{1}{16}A_G(1\ 1\ 2)^\top \right\rceil, (1\ 1\ 2)^\top\right\} = \max\left\{(1\ 2\ 2)^\top, (1\ 1\ 2)^\top\right\} = (1\ 2\ 2)^\top \ ,
\end{aligned}$$

and $A_G\mathbf{x}^2 \leq 16\mathbf{x}^2$. Hence, by Proposition 8.8(d), the vector $(1\ 2\ 2)^\top$ is the unique minimum (A_G, n)-super-vector $\mathbf{x}^*$. □

8.8.2 Consistent splittings

As mentioned before, the state-splitting construction of a sliding-block compressible (S, n)-excoder $\mathcal{E}$ starts with a deterministic presentation G of S. Typically, S will be a q-th power of a given constraint S' and G will be the q-th power of a graph presentation of S'; the integer n will be 2^p and $\mathcal{E}$ will thus be a sliding-block compressible excoder for S' at rate $p : q$.

Given S, G, and n, we compute an (A_G, n)-super-vector $\mathbf{x}$ using the algorithm in Figure 8.4. Just as in Section 5.2.1, the entry x_u in $\mathbf{x}$ will be referred to as the *weight of state* u. Using the vector $\mathbf{x}$, we will transform the graph G through out-splitting operations into a graph H such that $(1\ 1\ \ldots\ 1)^\top$ is an (A_H, n)-super-vector (i.e., the weights are all reduced to 1). It is easy to see that this is equivalent to saying that H has out-degree at most n. An (S, n)-excoder $\mathcal{E}$ will then be obtained by assigning tags to the edges of H.

Next we discuss the role of the (A_G, n)-super-vector in more detail. For an edge e in a graph, recall that $\tau_G(e) = \tau(e)$ denotes the terminal state of e.

Given a graph G, a positive integer n, and an (A_G, n)-super-vector $\mathbf{x} = (x_u)_u$, an $\mathbf{x}$-*consistent partition* of G is defined by partitioning the set, E_u, of outgoing edges from each state u in G such that

$$\sum_{e \in E_u^{(r)}} x_{\tau(e)} \leq n x_u^{(r)} \qquad \text{for} \qquad r = 1, 2, \ldots, N = N(u)\ , \tag{8.4}$$

where $x_u^{(r)}$ are nonnegative integers and

$$\sum_{r=1}^{N(u)} x_u^{(r)} = x_u\ . \tag{8.5}$$

The out-splitting based upon such a partition is called an $\mathbf{x}$-*consistent splitting*. The splitting is called *non-trivial* if at least one state u has at least two descendants $u^{(r)}, u^{(t)}$ such that both $x_u^{(r)}$ and $x_u^{(t)}$ are strictly positive. Note that here we have used the same terminology as in Section 5.2.3, but the reader should understand that throughout the remainder of this chapter, the notions of $\mathbf{x}$-*consistent partition* and $\mathbf{x}$-*consistent splitting* are as defined in this paragraph.

Let G' denote the graph after splitting. It is easy to see that the (A_G, n)-super-vector $\mathbf{x}$ gives rise to an *induced* $(A_{G'}, n)$-*super-vector* $\mathbf{x}' = (x'_u)_u$; namely, set $x'_{u^{(r)}} = x_u^{(r)}$. Note that (8.5) asserts that the weights of the descendants of a state u sum to the weight of u.

Example 8.15 Let S be presented by the graph $G = G^7_{2,\infty}$. For each state $u \in \{0, 1, 2\}$ in G, denote by $\mathcal{L}_u$ the set of labels of the outgoing edges from state u in G. Note that the graph G has memory 1, since the label of an edge determines the terminal state of that edge: edges whose labels end with '1' terminate in state 0, edges whose labels end with '10' terminate in state 1, and the remaining edges terminate in state 2. Hence, each set $\mathcal{L}_u$ completely describes the set, E_u, of outgoing edges from state u. We have

$$\begin{aligned}
\mathcal{L}_0 &= \{0000000, 0000001, 0000010, 0000100, 0001000, 0001001, 0010000, 0010001, 0010010\} ; \\
\mathcal{L}_1 &= \{0000000, 0000001, 0000010, 0000100, 0001000, 0001001, 0010000, 0010001, 0010010 \\
&\quad 0100000, 0100001, 0100010, 0100100\} ; \\
\mathcal{L}_2 &= \{0000000, 0000001, 0000010, 0000100, 0001000, 0001001, 0010000, 0010001, 0010010, \\
&\quad 0100000, 0100001, 0100010, 0100100, 1000000, 1000001, 1000010, 1000100, 1001000, 1001001\} .
\end{aligned}$$

We have shown in Example 8.14 that $\mathbf{x} = (1\ 2\ 2)^\top$ is an $(A_G, 16)$-super-vector. We next perform an $\mathbf{x}$-consistent splitting on G which will result in a graph H in which each state has weight 1. That is, up to tagging, the graph H will be an $(S, 16)$-excoder, namely, a rate $4 : 7$ excoder for the $(2, \infty)$-RLL constraint.

Since state 0 has weight 1, it will not be split (or more precisely, we split it trivially into one state, namely itself). Since each of the states 1 and 2 has weight 2, we would like to split each into two states of weight 1. Define the *weight of an edge* to be the weight of its terminal state. Now, $A_G(1\ 2\ 2)^\top = (15\ 22\ 32)^\top$, indicating that the total weights of outgoing edges from states 0, 1, and 2 are 15, 22, and 32, respectively. If we can partition the sets of outgoing edges from state 1 and state 2, each into two subsets of edges of total weight at most 16, then it follows that the weights in the graph H obtained from the corresponding out-splitting will all be 1, as desired. This indeed can be done as follows, where each partition element $E^{(r)}_u$ is represented by the respective label set $\mathcal{L}^{(r)}_u$ (labels that correspond to edges with weight 2 are underlined):

$$\begin{aligned}
\mathcal{L}_0 &= \{\underline{0000000}, 0000001, \underline{0000010}, \underline{0000100}, \underline{0001000}, 0001001, \underline{0010000}, 0010001, \underline{0010010}\} ; \\
\mathcal{L}^{(1)}_1 &= \{\underline{0000000}, 0000001, \underline{0000010}, \underline{0000100}, \underline{0001000}, 0001001, \underline{0010000}, 0010001, \underline{0010010}\} ; \\
\mathcal{L}^{(2)}_1 &= \{\underline{0100000}, 0100001, \underline{0100010}, \underline{0100100}\} ; \\
\mathcal{L}^{(1)}_2 &= \{\underline{0000000}, 0000001, \underline{0000010}, \underline{0000100}, \underline{0001000}, 0001001, \underline{0010000}, 0010001, \underline{0010010}, 1001001\} ; \\
\mathcal{L}^{(2)}_2 &= \{\underline{0100000}, 0100001, \underline{0100010}, \underline{0100100}, \underline{1000000}, 1000001, \underline{1000010}, \underline{1000100}, \underline{1001000}\} .
\end{aligned}$$

The reader can verify that the sets E_0, $E^{(1)}_1$, $E^{(2)}_1$, $E^{(1)}_2$, and $E^{(2)}_2$ have total weights 15, 15, 7, 16, and 16, respectively, as desired (in fact, the weights of $E^{(1)}_2$ and $E^{(2)}_2$ are forced to be 16).

The resulting split graph H will have five states, 0, $1^{(1)}$, $1^{(2)}$, $2^{(1)}$, and $2^{(2)}$, and the induced (A_H, n)-super-vector is $\mathbf{x}' = (1\ 1\ 1\ 1\ 1)^\top$, implying that the row sums of the

adjacency matrix

$$A_H = \begin{pmatrix} 3 & 2 & 2 & 4 & 4 \\ 3 & 2 & 2 & 4 & 4 \\ 1 & 1 & 1 & 2 & 2 \\ 4 & 2 & 2 & 4 & 4 \\ 2 & 2 & 2 & 5 & 5 \end{pmatrix}$$

are all at most 16. □

The following modification of of the corresponding result (Proposition 5.7) for conventional state splitting shows that in general there always is an $\mathbf{x}$-consistent splitting whenever we need one.

Proposition 8.9 *Let G be an irreducible graph which does not have out-degree at most n and let $\mathbf{x}$ be an (A_G, n)-super-vector. Then there is a non-trivial $\mathbf{x}$-consistent splitting of G.*

Proof. By the assumption, some state u in G has out-degree greater than n. By the pigeon-hole principle, there is a subset E of E_u with at most n edges such that n divides $\sum_{e\in E} x_{\tau(e)}$. Partition E_u into two sets $E_u^{(1)} \cup E_u^{(2)}$ where

$$E_u^{(1)} = E \qquad \text{and} \qquad E_u^{(2)} = E_u \setminus E \,,$$

and set

$$x_u^{(1)} = (1/n)\Big(\sum_{e\in E} x_{\tau(e)}\Big) \qquad \text{and} \qquad x_u^{(2)} = x_u - x_u^{(1)} \,.$$

It can be readily verified that the partition $E_u^{(1)} \cup E_u^{(2)}$ indeed implies a non-trivial $\mathbf{x}$-consistent splitting of state u. □

Passage from the (A_G, n)-super-vector $\mathbf{x}$ to the induced $(A_{G'}, n)$-super-vector $\mathbf{x}'$ always preserves the super-vector sum and increases the number of states. Since a super-vector is always a positive integer vector, it follows that repeated applications of Proposition 8.9 beginning with G eventually yield a graph H with an (A_H, n)-super-vector $(1\ 1\ \ldots\ 1)^\top$ and therefore with out-degree at most n. As mentioned earlier, if the original presentation G has finite memory m, then H will be (m, a)-definite on S. Finally, we assign tags to the edges of H, thereby obtaining an (m, a)-sliding-block compressible (S, n)-excoder $\mathcal{E}$. This establishes Proposition 8.7.

It may well be possible to merge states in H, resulting in a simpler excoder. Suppose that u and v are states in H such that every word that can be generated in H by paths starting at u can also be generated by paths starting at v. We can merge state u into state v by redirecting all incoming edges to u into v and then deleting state u with all its outgoing edges. This is the direct analogy of merging in Section 5.5.1.

Merging can add new words to those that are generated by H (and $\mathcal{E}$), and it may also give rise to new paths that present words of S. In particular, it may destroy definiteness on S. However, suppose that there are integers $\mathsf{m}' \geq \mathsf{m}$ and $\mathsf{a}' \geq \mathsf{a}$ such that the following holds: for every word $\mathbf{w} \in S_{\mathsf{m}'}$ that can be generated by a path in H that terminates in u and for every word $\mathbf{w}' \in S_{\mathsf{a}'+1}$ that can be generated in H from v but not from u, we have $\mathbf{w}\mathbf{w}' \notin S_{\mathsf{m}'+\mathsf{a}'+1}$. Under this condition, the merged graph is $(\mathsf{m}', \mathsf{a}')$-definite on S, since definiteness on S involves only words in S, which may be a proper subset of the constraint presented by the merged graph.

Example 8.16 Continuing the discussion in Example 8.15, we observe that $\mathcal{L}_1^{(2)} \subset \mathcal{L}_2^{(2)}$; furthermore, since G has memory 1, edges with the same label in $E_1^{(2)}$ and $E_2^{(2)}$ terminate in the same state of G. It follows that every word that can be generated in H from $1^{(2)}$ can also be generated from $2^{(2)}$.

Let $\mathbf{w}$ be a word that is generated by a path in H that terminates in state $1^{(2)}$. This word is also generated in G by a path that terminates in state 1 and, so, $\mathbf{w}$ ends with '10'. Let $\mathbf{w}'$ be a word that can be generated in H from $2^{(2)}$ but not from $1^{(2)}$. The word $\mathbf{w}'$ necessarily starts with '1' since

$$\mathcal{L}_2^{(2)} \setminus \mathcal{L}_1^{(2)} = \{1000000, 1000001, 1000010, 1000100, 1001000\} .$$

It follows that $\mathbf{w}\mathbf{w}'$ contains the sub-word '101'; as such, it violates the $(2, \infty)$-RLL constraint and, therefore, it does not belong to S. We conclude that the definiteness on S will be preserved upon merging state $1^{(2)}$ into state $2^{(2)}$. Similarly, since $\mathcal{L}_0 = \mathcal{L}_1^{(1)} \subset \mathcal{L}_2^{(1)}$ and $\mathcal{L}_2^{(1)} \setminus \mathcal{L}_0 = \{1001001\}$, we see that states 0 and $1^{(1)}$ can be merged into state $2^{(1)}$ while preserving definiteness on S. Thus, the resulting merged graph, H', has only two states, $a \equiv 2^{(1)}$ and $b \equiv 2^{(2)}$, and it is $(1, 1)$-definite on S. In fact, H' is $(0, 1)$-definite on S: since $\mathcal{L}_2^{(1)} \cap \mathcal{L}_2^{(2)} = \emptyset$, the initial state of the edge that generates the current codeword is uniquely determined by that codeword.

Finally, we assign tags from $\{0, 1\}^4$ to the edges of H' to obtain the $(0, 1)$-sliding-block compressible excoder $\mathcal{E}_{2,\infty}$ whose transition table is shown in Table 8.3. In that table, the rows are indexed by the tags (4-blocks) and the columns by the states of $\mathcal{E}_{2,\infty}$. Entry (s, u) in the table contains the label and terminal state of the outgoing edge from state u that is tagged by s. Observe that the tags have been assigned to the edges of $\mathcal{E}_{2,\infty}$ so that tags of edges that have the same label (and initial state) will differ in only the last bit. Thus, whenever the compression of the current 4-block depends on the upcoming 7-codeword, such dependency is limited only to determining the last bit of the 4-block. □

In Table 8.4, we list for each $p \leq 16$ the largest value of q which allows a block excoder at rate $p : q$ as well as the largest value of q which allows a $(0, 1)$-sliding-block compressible excoder at rate $p : q$ for the $(2, \infty)$-RLL constraint. The values of q for the block excoder

	a	b
0000	0000000, a	0100000, a
0001	0000000, b	0100000, b
0010	0000010, a	0100100, a
0011	0000010, b	0100100, b
0100	0000100, a	1000000, a
0101	0000100, b	1000000, b
0110	0001000, a	1000100, a
0111	0001000, b	1000100, b
1000	0010000, a	1001000, a
1001	0010000, b	1001000, b
1010	0010010, a	0100001, a
1011	0010010, b	1000001, a
1100	0000001, a	0100010, a
1101	0001001, a	0100010, b
1110	0010001, a	1000010, a
1111	1001001, a	1000010, b

Table 8.3: Excoder $\mathcal{E}_{2,\infty}$.

p	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
q (block)	1	3	4	6	8	10	11	13	15	17	19	21	22	24	26	28
q ($(0,1)$-sliding block)	1	3	**5**	**7**	**9**	10	**12**	**14**	**16**	**18**	19	21	**23**	**25**	**27**	**29**

Table 8.4: Maximal values of q for existence of block excoders and $(0,1)$-sliding-block compressible excoders for the $(2,\infty)$-RLL constraint.

case can be obtained by (8.1), where the values of $|S_q|$ can be computed using the formulas in [Imm91, Section 5.2] (it can be verified that the same values of q apply also to $(0,0)$-sliding-block compressible excoders). The values of q for the $(0,1)$-sliding-block compressible case were obtained by actually computing $(A^q_{G_{2,\infty}}, 2^p)$-super-vectors and verifying that excoders can be obtained by one round of out-splitting. In particular, the rate $4:7$ is attained by the excoder $\mathcal{E}_{2,\infty}$ in Example 8.16. It is worthwhile pointing out that a rate $5:9 = .5555\ldots$ is attainable by a $(0,1)$-sliding-block compressible excoder with ten states. This rate is above the capacity ($\approx .5515$) by less than 1%. The boldface numbers in Table 8.4 indicate values of q which are larger (by 1) than those attainable by block codes. The results in Table 8.4 remain unaffected if one were to replace the $(2,\infty)$-RLL constraint with the $(2,15)$-RLL constraint.

We also mention here the existence of the following two $(0,1)$-sliding-block compressible excoders that might be of practical interest: a rate $8:9$ excoder for the $(0,2)$-RLL constraint, and a rate $5:7$ excoder for the $(1,7)$-RLL constraint; there are no block excoders at such rates for those constraints.

8.8.3 Reduction of edge effect in error propagation

Recall that in the analysis of Section 8.7, we made the conservative assumption that a p-block is wholly corrupted even if only one bit in that p-block is in error. Yet, as we know for block codes, and as we have demonstrated in Example 8.16 for sliding-block compressible excoders, special care in the assignment of tags can reduce the dependency of certain bits in p-blocks on certain channel bits, thereby reducing the effect of error. Specifically, when using the excoder $\mathcal{E}_{2,\infty}$ of Example 8.16, a burst of $L = 40$ channel bits, while affecting up to eight 4-blocks, can corrupt only up to 29 bits (and not 32 bits) in those blocks. This, in turn, allows us to modify Table 8.2 to produce Table 8.5 (the modified entries are indicated by boldface numbers). It follows from Table 8.5 that for the range $841 \leq \kappa \leq 7,778$, the actual

B	$D(40, B)$	ν	ρ	κ
4	8	544	64	840
5	**7**	**1,155**	**70**	**1,898**
6	6	2,340	(72)	3,969
7	**5**	**4,515**	**70**	**7,778**
8	5	10,280	80	17,850

Table 8.5: Modified Table 8.2.

redundancy that will be required is strictly less than dictated by Table 8.2. In particular, for $3,970 \leq \kappa \leq 7,778$, the savings amount to reducing the redundancy from 80 to 70 bits.

Additional savings can be obtained by using an excoder with more states, as we demonstrate in the next example.

Example 8.17 Looking closely at Table 8.3, one can see that the compression of the last bit of the current 4-block depends on the first, second, fourth, and seventh bits of the upcoming 7-codeword. The dependency on the seventh bit has a slight disadvantage in case of short bursts—in particular isolated *bit-shift errors*, where an occurrence of '1' in a constrained sequence is shifted by one position, thereby resulting in two adjacent erroneous bits in the constrained sequence. If those two bits cross the boundary of adjacent 7-codewords, the error may propagate through the compression to up to 9 bits in three 4-blocks.

By allowing more states, we are able to present another rate 4 : 7 excoder for the $(2, \infty)$-RLL constraint, $\mathcal{E}'_{2,\infty}$, where the compression of the last bit of the current 4-block depends on the first, second, fourth, and fifth bits of the upcoming 7-codeword (whereas the other bits of the 4-block depend only on the current 7-codeword). Here, one bit-shift error in the constrained sequence may affect only two 4-blocks. The excoder $\mathcal{E}'_{2,\infty}$ is obtained through a different out-splitting of state 2 in G (into states $2^{(1')}$ and $2^{(2')}$), resulting in four states, $\alpha \equiv 0 = 1^{(1)}$, $\beta \equiv 1^{(2)}$, $\gamma \equiv 2^{(1')}$, and $\delta \equiv 2^{(2')}$, with a transition table as shown in Table 8.6. The excoder $\mathcal{E}'_{2,\infty}$ is $(1, 1)$-definite (but not $(0, 1)$-definite) on the constraint; still,

	α	β	γ	δ
0000	0000000, γ	0100010, α	0000000, γ	0100010, α
0001	0000000, δ	0100010, β	0000000, δ	0100010, β
0010	0000010, α	0100000, γ	0000010, α	0100000, γ
0011	0000010, β	0100000, δ	0000010, β	0100000, δ
0100	0000100, γ	–	0000100, γ	1000010, α
0101	0000100, δ	–	0000100, δ	1000010, β
0110	0010000, γ	–	0010000, γ	1000000, γ
0111	0010000, δ	–	0010000, δ	1000000, δ
1000	0010010, α	–	0010010, α	1001001, α
1001	0010010, β	0100001, α	0010010, β	0100001, α
1010	0001000, γ	0100100, γ	0100100, γ	0001000, γ
1011	0001000, δ	0100100, δ	0100100, δ	0001000, δ
1100	–	–	1000100, γ	1000001, α
1101	0001001, α	–	1000100, δ	0001001, α
1111	0000001, α	–	0000001, α	1001000, γ
1111	0010001, α	–	0010001, α	1001000, δ

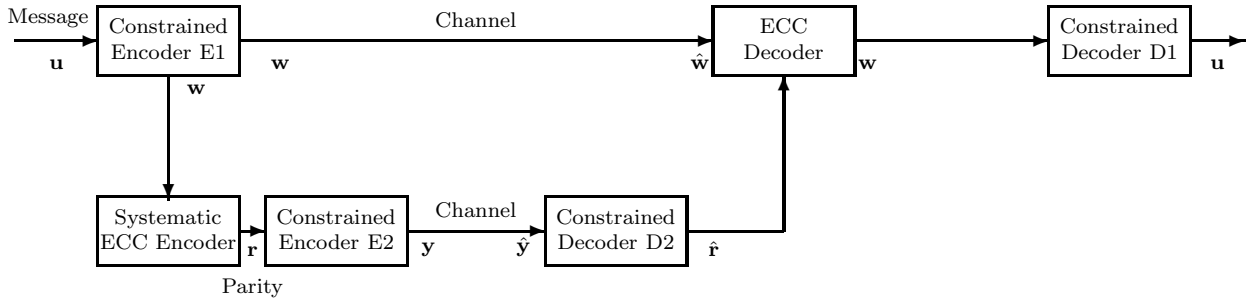
Table 8.6: Excoder $\mathcal{E}'_{2,\infty}$.

the particular tagging of the edges makes it $(0,1)$-sliding-block compressible. Note that states α and β have less than 16 outgoing edges and, so, certain elements of $\{0,1\}^4$ do not tag those edges. □

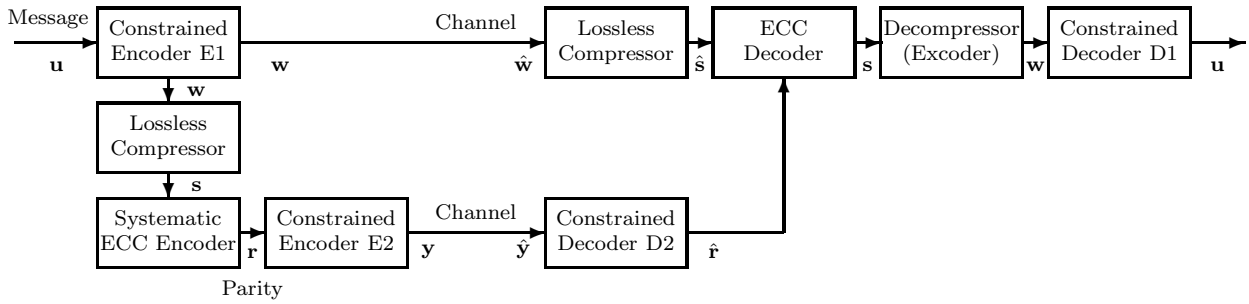
Problems

To be filled in.

(a)



(b)



(c)

Figure 8.1: (a) Standard concatenation. (b) Modified concatenation. (c) Modified concatenation with lossless compression.

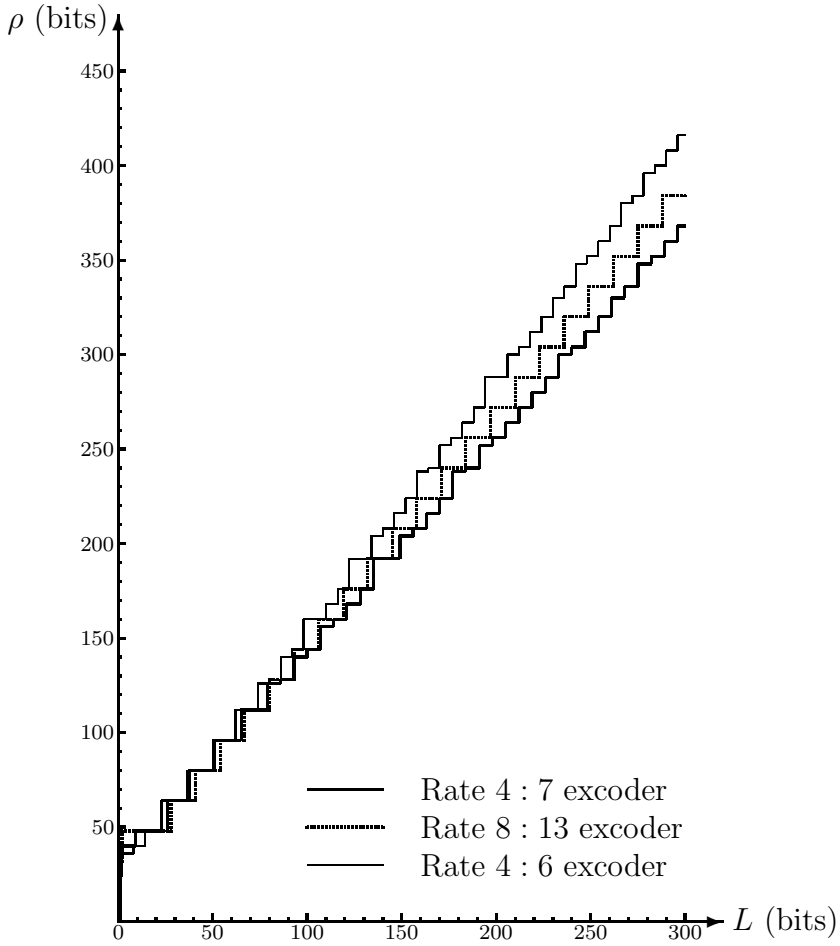


Figure 8.2: Redundancy for $4,096$ user bits and various burst lengths using three different excoders for the $(2, \infty)$-RLL constraint.

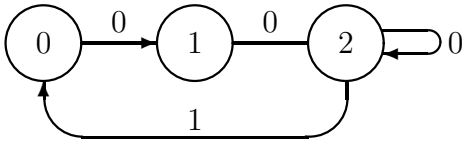


Figure 8.3: Graph presentation $G_{2,\infty}$ of the $(2, \infty)$-RLL constraint.

$\mathbf{x} \leftarrow (1\ 1\ \ldots\ 1)^\top$;
`while` $(A\mathbf{x} \not\leq n\mathbf{x})$
    $\mathbf{x} \leftarrow \max\left\{\left\lceil \frac{1}{n} A\mathbf{x} \right\rceil, \mathbf{x}\right\}$; /* apply $\lceil\cdot\rceil$ and $\max\{\cdot,\cdot\}$ componentwise */
`return` $\mathbf{x}$;

Figure 8.4: Reversed Franaszek algorithm for computing (A, n)-super-vectors.

Chapter 9

Error-Correcting Constrained Coding

In this chapter, we consider codes that have combined error-correction and constrained properties. We begin with a discussion of error mechanisms in recording systems and the corresponding error types observed. We then discuss associated metrics imposed on constrained systems—primarily the Hamming, Lee, and Euclidean metrics—and we survey the literature on bounds and code constructions. In addition, we consider two important classes of combined error-correction/constrained codes: spectral null codes and forbidden list codes.

9.1 Error-mechanisms in recording channels

Magnetic recording systems using peak detection, as described in Chapter 1 of this chapter, are subject to three predominant types of errors at the peak detector output. The most frequently observed error is referred to as a *bitshift error*, where a pair of recorded symbols 01 is detected as 10 (a *left bitshift*), or the pair 10 is detected as 01 (a *right bitshift*). Another commonly occurring error type is called a *drop-out error* or, sometimes, a *missing-bit error*, where a recorded symbol 1 is detected as a 0. Less frequently, a *drop-in error* or *extra-bit error* results in the detection of a recorded 0 as a 1. It is convenient to refer to the drop-in and drop-out errors as *substitution errors*.

Hamming-metric constrained codes are most pertinent in recording channels that behave like a binary symmetric channel, in which drop-in and drop-out errors occur with equal probability. However, there are alternative models of interest that suggest the use of codes designed with other criteria in mind beyond optimization of minimum Hamming distance. Among these models, the two that have received the most attention are: the asymmetric channel—where only drop-in errors or drop-out errors, but not both, are encountered; and the bitshift channel—where a symbol 1 is allowed to shift position by up to a prespecified number of positions.

Another error type we will consider is a *synchronization error*, resulting in an insertion or deletion of a symbol 0 in the detected symbol sequence. In practical digital recording systems on disks and tape, this type of error can have catastrophic consequences with regard to recovery of information that follows the synchronization loss. As a result, recording devices use synchronization and clock generation techniques in conjunction with code constraints, such as the k constraint in RLL codes for peak detection and the $\mathbf{G}$ constraint in PRML $(0, \mathbf{G}/\mathbf{I})$ codes, to effectively preclude such errors. Nevertheless, RLL-constrained synchronization-error-correcting codes have some intrinsic coding-theoretic interest, and we will discuss them below. Codes capable of correcting more than one insertion and deletion error may also be used to protect against bitshift errors, which result from the insertion and deletion of 0's on either side of a 1. The edit distance, or Levenshtein metric, and the Lee metric arise naturally in the context of synchronization errors.

In recording systems using partial-response with some form of sequence detection, exemplified by the PRML system described in Chapter 1, the maximum-likelihood detector tends to generate burst errors whose specific characteristics can be determined from the error events associated with the underlying trellis structure. We will briefly survey various trellis-coded modulation approaches for PRML that yield codes which combine $(0, \mathbf{G}/\mathbf{I})$ constraints with enhanced minimum Euclidean distance.

In practice, constrained codes must limit error propagation. Sliding-block decoders of the most frequently used (d, k)-RLL codes and PRML $(0, \mathbf{G}/\mathbf{I})$ codes typically will propagate a single detector error into a burst of length no more than eight bits. For example, the maximum error propagation of the industry standard $(2, 7)$-RLL and $(1, 7)$-RLL codes are four bits and five bits, respectively, and the PRML $(0, 4/4)$ code limits errors to a single byte. The conventional practice in digital recording devices is to detect and correct such errors by use of an outer error-correcting code, such as a Fire code, interleaved Reed-Solomon code, or a modification of such a code.

9.2 Gilbert-Varshamov-type lower bounds

9.2.1 Classical bound for the Hamming metric

There are several error metrics that arise in the context of digital recording using constrained sequences. For substitution-type errors and bitshift errors, possibly propagated into burst errors by the modulation decoder, it is natural to consider error-correcting codes based upon the *Hamming metric*. It is therefore of interest to investigate Hamming distance properties of constrained sequences.

The Gilbert-Varshamov bound provides for unconstrained sequences over a finite alphabet Σ a lower bound on the size of codes with prespecified minimum Hamming distance. In this

section we present bounds of the Gilbert-Varshamov type and apply them to the class of runlength-limited binary sequences.

Let Σ denote a finite alphabet of size $|\Sigma|$ and denote the Hamming distance between two words $\mathbf{w}, \mathbf{w}' \in \Sigma^q$ by $\Delta_{\text{Hamming}}(\mathbf{w}, \mathbf{w}')$. For a word $\mathbf{w} \in \Sigma^q$, let $\mathcal{B}_{\Sigma^q}(\mathbf{w}; r)$ be the Hamming sphere of radius r in Σ^q centered at $\mathbf{w}$, that is,

$$\mathcal{B}_{\Sigma^q}(\mathbf{w}; r) = \{\mathbf{w}' : \Delta_{\text{Hamming}}(\mathbf{w}, \mathbf{w}') \leq r\} .$$

Let $\mathcal{V}_{\Sigma^q}(\mathbf{w}; r)$ be the cardinality or *volume* of the sphere $\mathcal{B}_{\Sigma^q}(\mathbf{w}; r)$. This quantity is $\sum_{i=0}^{r} \binom{q}{i} (|\Sigma|-1)^i$, independent of the center word $\mathbf{w}$, so we will use the shorthand notation $\mathcal{V}_{\Sigma^q}(r)$.

The Gilbert-Varshamov bound provides a lower bound on the achievable cardinality M of a subset of Σ^q with minimum Hamming distance at least d. We will refer to such a subset as a $(\Sigma^q, M, \mathsf{d})$-code.

Theorem 9.1 *There exists a $(\Sigma^q, M, \mathsf{d})$-code with*

$$M \geq \frac{|\Sigma|^q}{\mathcal{V}_{\Sigma^q}(\mathsf{d}-1)} .$$

For future reference, we recall that the proof of this bound is obtained by iteratively selecting the lth codeword $\mathbf{w}_l$ in the code from the complement of the union of Hamming spheres of radius $q-1$ centered at the previously selected codewords, $\Sigma^q - \cup_{i=1}^{l-1} \mathcal{B}_{\Sigma^q}(\mathbf{w}_i; \mathsf{d}-1)$. Continuing this procedure until the union of spheres exhausts Σ^q, an $(\Sigma^q, M, \mathsf{d})$-code is obtained whose size M satisfies the claimed inequality. □

Let $\delta = \mathsf{d}/q$ denote the *relative minimum distance* and let $H(\delta; z) = -\delta \cdot \log \delta - (1 - \delta) \cdot \log(1-\delta) + \delta \cdot \log(z-1)$, for $0 \leq \delta \leq 1 - (1/z)$, be a z-ary generalization of the entropy function. The Gilbert-Varshamov bound in terms of the rate R of the resulting code can be expressed as

$$R = \frac{\log M}{q} \geq \log |\Sigma| - \frac{\log \mathcal{V}_{\Sigma^q}(\mathsf{d}-1)}{q} \geq \log |\Sigma| - H(\delta; |\Sigma|)$$

(for the last inequality, we refer the reader to [Berl84, pp. 300–301].

9.2.2 Hamming-metric bound for constrained systems

Any generalization of the Gilbert-Varshamov bound to a constrained system S must take into account that the volumes of Hamming spheres in $S \cap \Sigma^q$ are not necessarily independent of the specified centers. Before deriving such bounds, we require a few more definitions. Let

X denote an arbitrary subset of Σ^q. For a word $\mathbf{w} \in X$, we define the Hamming sphere of radius r in X by

$$\mathcal{B}_X(\mathbf{w}; r) = \mathcal{B}_{\Sigma^q}(\mathbf{w}; r) \cap X \ .$$

The maximum volume of the spheres of radius r in X is

$$\mathcal{V}_{X,\max}(r) = \max_{\mathbf{w} \in X} |\mathcal{B}_X(\mathbf{w}; r)| \ ,$$

and the average volume of spheres of radius r in X is given by

$$\mathcal{V}_X(r) = \frac{1}{|X|} \sum_{\mathbf{w} \in X} |\mathcal{B}_X(\mathbf{w}; r)| \ .$$

We also define the set $\mathcal{B}_X(r)$ of pairs $(\mathbf{w}, \mathbf{w}')$ of words in X at distance no greater than r,

$$\mathcal{B}_X(r) = \{(\mathbf{w}, \mathbf{w}') \ : \ \Delta_{\text{Hamming}}(\mathbf{w}, \mathbf{w}') \leq r\} \ .$$

Note that $|\mathcal{B}_X(r)| = \mathcal{V}_X(r) \cdot |X|$. Finally, we define an (X, M, d)-code to be a $(\Sigma^q, M, \mathsf{d})$-code that is a subset of X.

A straightforward application of the Gilbert-Varshamov construction yields the following result.

Lemma 9.2 *Let X be a subset of Σ^q and d be a positive integer. Then, there exists an (X, M, d)-code with*

$$M \geq \frac{|X|}{\mathcal{V}_{X,\max}(\mathsf{d}-1)} \ .$$

The following generalization of the Gilbert-Varshamov bound, first proved by Kolesnik and Krachkovsky [KolK91], is the basis for the more refined bounds derived later in the section. It provides a bound based upon the *average* volume of spheres, rather than the *maximum* volume, as was used in Lemma 9.2.

Lemma 9.3 *Let X be a subset of Σ^q and d be a positive integer. Then, there exists an (X, M, d)-code with*

$$M \geq \frac{|X|}{4\mathcal{V}_X(\mathsf{d}-1)} = \frac{|X|^2}{4|\mathcal{B}_X(\mathsf{d}-1)|} \ .$$

Proof. Consider the subset X' of words $\mathbf{w} \in X$ whose Hamming spheres of radius $\mathsf{d}-1$ satisfy $|\mathcal{B}_X(\mathbf{w}; \mathsf{d}-1)| \leq 2\mathcal{V}_X(\mathsf{d}-1)$. The subset X' must then satisfy $|X'| \geq |X|/2$. If we iteratively select codewords from X', following the procedure used in the derivation of the Gilbert-Varshamov bound, we obtain an (X, M, d)-code, where

$$M \geq \frac{|X'|}{2\mathcal{V}_X(\mathsf{d}-1)} \geq \frac{\frac{1}{2}|X|}{2\mathcal{V}_X(\mathsf{d}-1)} = \frac{|X|^2}{4|\mathcal{B}_X(\mathsf{d}-1)|} \ ,$$

as desired. □

In general, neither of the bounds in the preceding two lemmas is strictly superior to the other, as observed by Gu and Fuja [GuF93]. However, using an analysis of a new code search algorithm—dubbed the "altruistic algorithm" to distinguish it from the "greedy algorithm" that lies at the heart of the standard Gilbert-Varshamov form of bound—they eliminated the factor of 4 in the denominator of the bound in Lemma 9.3. This improved lower bound, stated below as Lemma 9.4, is always at least as good as the bound in Lemma 9.2, and a strict improvement over Lemma 9.3.

The key element of the improved code search algorithm is that, at each codeword selection step, the remaining potential codeword with the largest number of remaining neighbors at distance $\mathsf{d}-1$ or less takes itself out of consideration. As noted in [GuF93], a similar approach was developed independently by Ytrehus [Yt91a], who applied it to compute bounds for runlength-limited codes with various error detection and correction capabilities [Yt91b].

Lemma 9.4 *Let X be a subset of Σ^q and d be a positive integer. Then, there exists an (X, M, d)-code with*

$$M \geq \frac{|X|}{\mathcal{V}_X(\mathsf{d}-1)} = \frac{|X|^2}{|\mathcal{B}_X(\mathsf{d}-1)|} \ .$$

Kolesnik and Krachkovsky [KolK91] applied Lemma 9.3 to sets X consisting of words of length q in runlength-limited and charge constrained systems. Their asymptotic lower bound was based upon an estimate of the average volume of constrained Hamming spheres, whose centers ranged over all of $S \cap \Sigma^q$. Their estimate made use of a generating function for pairwise q-block distances in these families of constrained systems.

9.2.3 Improved Hamming-metric bounds

Marcus and Roth [MR92] found improved bounds by considering subsets X of $S \cap \Sigma^q$ where additional constraints, depending upon the designed relative minimum distance δ, are imposed upon the frequency of occurrence of code symbols $w \in \Sigma$. We now discuss the derivation of these bounds.

Let S be a constrained system over Σ presented by an irreducible deterministic graph $G = (V, E, L)$. Denote by $\Delta(G)$ the set of all stationary Markov chains on G (see Section 3.5). The entropy of $\mathcal{P} \in \Delta(G)$ is denoted by $\mathsf{H}(\mathcal{P})$.

Given a stationary Markov chain $\mathcal{P} \in \Delta(G)$, along with a vector of real-valued functions $\mathbf{f} = (f_1 \ f_2 \ \ldots \ f_t) : E_G \to \mathbb{R}^t$, we denote by $\mathsf{E}_{\mathcal{P}}(\mathbf{f})$ the expected value of $\mathbf{f}$ with respect to $\mathcal{P}$:

$$\mathsf{E}_{\mathcal{P}}(\mathbf{f}) = \sum_{e \in E_G} \mathcal{P}(e)\mathbf{f}(e) \ .$$

For a subset $W = \{w_1, w_2, \ldots, w_t\}$ of Σ, we define the vector indicator function $\mathcal{I}_W : E_G \to \mathbb{R}^t$ by $\mathcal{I}_W = (\mathcal{I}_{w_1}, \mathcal{I}_{w_2}, \ldots, \mathcal{I}_{w_t})$, where $\mathcal{I}_w : E_G \to \mathbb{R}$ is the indicator function for a symbol $w \in \Sigma$:

$$\mathcal{I}_w(e) = \begin{cases} 1 & \text{if } L_G(e) = w \\ 0 & \text{otherwise} \end{cases} .$$

Let $G \times G$ denote the labeled product graph defined by $V_{G\times G} = V_G \times V_G = \{(u, u') : u, u' \in V_G\}$ and $E_{G\times G} = E_G \times E_G$. There is an edge (e, e') in $G \times G$ from state (u, u') to state (v, v') whenever e is an edge in G from state u to state u' and e' is an edge in G from state v to v'. The labeling on $G \times G$ is defined by $L_{G\times G}(e, e') = (L_G(e), L_G(e'))$. We define on $E_{G\times G}$ the coordinate indicator functions $\mathcal{I}_W^{(1)}$ and $\mathcal{I}_W^{(2)}$, given by $\mathcal{I}_W^{(1)}((e, e')) = \mathcal{I}_W(e)$ and $\mathcal{I}_W^{(2)}((e, e')) = \mathcal{I}_W(e')$. Finally, we define the coordinate distance function $\mathcal{D} : E_{G\times G} \to \mathbb{R}$ by

$$\mathcal{D}((e, e')) = \begin{cases} 1 & \text{if } L_G(e) \neq L_G(e') \\ 0 & \text{otherwise} \end{cases} .$$

For a given symbol subset W of size t and a vector $\mathbf{p} \in [0, 1]^t$, we now define the quantities

$$\mathcal{S}_W(\mathbf{p}) = \sup_{\substack{\mathcal{P} \in \Delta(G) \\ \mathsf{E}_P(\mathcal{I}_W) = \mathbf{p}}} \mathsf{H}(\mathcal{P})$$

and

$$\mathcal{T}_W(\mathbf{p}, \delta) = \sup_{\substack{\mathcal{P}' \in \Delta(G \times G) \\ \mathsf{E}_{\mathcal{P}'}(\mathcal{I}_W^{(i)}) = \mathbf{p}\,,\ i = 1,2 \\ \mathsf{E}_{\mathcal{P}'}(\mathcal{D}) \in [0, \delta]}} \mathsf{H}(\mathcal{P}') .$$

Finally, to concisely state the bounds, we introduce the following function of the relative designed distance δ:

$$R_W(\delta) = \sup_{\mathbf{p} \in [0,1]^t} \{\, 2\mathcal{S}_W(\mathbf{p}) - \mathcal{T}_W(\mathbf{p}, \delta) \,\} .$$

The following theorem is proved in [MR92]. It is obtained by application of Lemma 9.3 to the words in $S \cap \Sigma^q$ generated by cycles in G starting and ending at a specified state $u \in G$, with frequency of occurrence of the symbols $w_i \in W$ given approximately by p_i, for $i = 1, 2, \ldots, t$.

Theorem 9.5 *Let S be a constrained system over Σ presented by a primitive deterministic graph, let $\delta > 0$, and let W be a subset of Σ of size t. Then there exist $(S \cap \Sigma^q, M, \delta q)$-codes satisfying*

$$\frac{\log M}{q} \geq R_W(\delta) - o(1) ,$$

where $o(1)$ stands for a term that goes to zero as q tends to infinity.

Computation of the quantities $\mathcal{S}_W(\mathbf{p})$ and $\mathcal{T}_W(\mathbf{p}, \delta)$ requires the solution of a constrained optimization problem in which the objective function $\mathcal{P} \to \mathsf{H}(\mathcal{P})$ is concave, and the constraints are linear. The theory of convex duality based upon Lagrange multipliers provides a method to translate the problem into an unconstrained optimization with a convex objective function [MR92].

In order to reformulate the problem, we need to introduce a vector-valued matrix function that generalizes the adjacency matrix A_G. For a function $\mathbf{f} : E_G \to \mathbb{R}^t$ and $\mathbf{x} \in \mathbb{R}^t$, let $A_{G;\mathbf{f}}(\mathbf{x})$ be the matrix defined by

$$\Big(A_{G;\mathbf{f}}(\mathbf{x}) \Big)_{u,v} = \sum_{e\,:\,\sigma(e)=u,\,\tau(e)=v} 2^{-\mathbf{x}\cdot\mathbf{f}(e)}$$

We remark that for any function $\mathbf{f}$, the matrix $A_{G;\mathbf{f}}(\mathbf{0})$ is precisely the adjacency matrix of G.

The following lemma is the main tool in translating the constrained optimization problem to a more tractable form. It is a consequence of standard results in the theory of convex duality.

Lemma 9.6 *Let G and $\mathbf{f}$ be as above. Let $\mathbf{g} : E_G \to \mathbb{R}^l$, and define $\boldsymbol{\psi} = [\mathbf{f}, \mathbf{g}] : E_G \to \mathbb{R}^{t+l}$. Then, for any $\mathbf{r} \in \mathbb{R}^t$ and $\mathbf{s} \in \mathbb{R}^l$,*

$$\sup_{\substack{\mathcal{P} \in \Delta(G) \\ \mathsf{E}_{\mathcal{P}}(\mathbf{f}) = \mathbf{r} \\ \mathsf{E}_{\mathcal{P}}(\mathbf{g}) \le \mathbf{s}}} \mathsf{H}(\mathcal{P}) = \inf_{\substack{\mathbf{x} \in \mathbb{R}^t \\ \mathbf{z} \in (\mathbb{R}^+)^l}} \Big\{ \mathbf{x}\cdot\mathbf{r} + \mathbf{z}\cdot\mathbf{s} + \log \lambda(A_{G;\boldsymbol{\psi}}(\mathbf{x},\mathbf{z})) \Big\} \ .$$

Applying Lemma 9.6 to Theorem 9.5, we can derive dual formulas for the lower bounds $R_W(\delta)$, for a specified symbol set W and relative minimum distance δ. For the case where W consists of a single symbol $w \in \Sigma$, the resulting formula is particularly tractable. To express it succinctly, we define

$$\mathcal{J}_w = \Big[\mathcal{I}^{(1)}_{\{w\}} + \mathcal{I}^{(2)}_{\{w\}}, \mathcal{D} \Big] : E_{G\times G} \to \mathbb{R}^2$$

and, to simplify notation, $\mathcal{S}_w(p) = \mathcal{S}_{\{w\}}\Big((p)\Big)$, $\mathcal{T}_w(p,\delta) = \mathcal{T}_{\{w\}}\Big((p),\delta\Big)$, and $R_w(\delta) = R_{\{w\}}(\delta)$.

The lower bound on attainable rates follows from the following theorem.

Theorem 9.7 *Let S be a constrained system over Σ presented by a primitive graph G, let $\delta > 0$, and let $w \in \Sigma$. Then*

a)

$$\mathcal{S}_w(p) = \inf_{x \in \mathbb{R}} \{ px + \log \lambda(A_{G;\mathcal{I}_w}(x)) \} \ ;$$

b)

$$\mathcal{T}_w(p,\delta) \;=\; \inf_{x\in\mathbb{R},z\in\mathbb{R}^+}\{2px \;+\; \delta z \;+\; \log\lambda(A_{G\times G;\mathcal{J}_w}(x,z))\}\;;$$

c)

$$R_w(\delta) = \sup_{p\in[0,1]}\Big\{2\inf_{x\in\mathbb{R}}\Big\{px+\log\lambda(A_{G;\mathcal{J}_w}(x))\Big\} - \inf_{x\in\mathbb{R},\,z\in\mathbb{R}^+}\Big\{2px+\delta z+\log\lambda(A_{G\times G;\mathcal{J}_w}(x,z))\Big\}\Big\}\;.$$

In particular, if $\mathcal{P}\in\Delta(G)$ has maximal entropy rate

$$\mathsf{H}(\mathcal{P}) = \sup_{\mathcal{P}'\in\Delta(G)}\mathsf{H}(\mathcal{P}')\;,$$

and the symbol probability p equals $\mathsf{E}_{\mathcal{P}}(\mathcal{I}_w)$, then

$$\mathcal{S}_w(p) = \log\lambda(A_G)$$

and, setting $x=0$ in part b) of Theorem 9.7,

$$\mathcal{T}_w(p,\delta)\le\inf_{z\in\mathbb{R}^+}\{\delta z+\log\lambda(A_{G\times G;\mathcal{J}_w}(0,z))\}\;.$$

From Theorem 9.5 and part c) of Theorem 9.7, we recover the lower bound of Kolesnik and Krachkovsky.

Corollary 9.8

$$\frac{\log M}{q}\ge R_{KK}(\delta)-o(1)\;,$$

where

$$R_{KK}(\delta) = 2\log\lambda(A_G) - \inf_{z\in\mathbb{R}^+}\{\delta z+\log\lambda(A_{G\times G;\mathcal{J}_w}(0,z))\}\;.$$

Better lower bounds can be obtained by prescribing the frequency of occurrence of words $\mathbf{w}$ of arbitrary length, rather than only symbols. See [MR92] for more details.

Example 9.1 For the $(0,1)$-RLL constrained system, consider the cases where $W=\{11\}$ and $W=\{111\}$, with corresponding lower bounds R_{11} and R_{111}. It is not difficult to see that $R_{11}(\delta)$ must equal $R_1(\delta)$. Table 9.1 from [MR92] gives the values of $R_{KK}(\delta)$, $R_1(\delta)$, and $R_{111}(\delta)$ for selected values of δ. □

We remark that, in some circumstances, one might assign to each edge $e\in E_G$ a *cost* associated to its use in a path generating a sequence in S. Lower bounds on the rate of codes into S with specified relative minimum distance δ and *average cost constraint* have been derived by Winick and Yang [WY93] and Khayrallah and Neuhoff [KN96].

δ	$R_{KK}(\delta)$	$R_1(\delta)$	$R_{111}(\delta)$
0.00	0.6942	0.6942	0.6942
0.05	0.4492	0.4504	0.4507
0.10	0.3055	0.3096	0.3109
0.15	0.2014	0.2094	0.2119
0.20	0.1241	0.1361	0.1399
0.25	0.0679	0.0831	0.0877
0.30	0.0295	0.0461	0.0506
0.35	0.0073	0.0218	0.0254
0.40	0	0.0077	0.0097
0.45	0	0.0013	0.0016
0.50	0	0	0

Table 9.1: Attainable rates for $(0, 1)$-RLL constrained system.

9.3 Towards sphere-packing upper bounds

In comparison to lower bounds, much less is known about upper bounds on the size of block codes for constrained systems. We describe here a general technique introduced by Abdel-Ghaffar and Weber in [AW91]. Let S be a constrained system over Σ and let X be a nonempty subset of Σ^q. For a word $\mathbf{w} \in \Sigma^q$, denote by $\mathcal{B}_X(\mathbf{w}; t)$ the set of words $\mathbf{w}' \in X$ which are at distance t or less from $\mathbf{w}$ according to some distance measure $\Delta(\mathbf{w}, \mathbf{w}')$. If $\mathcal{C}$ is an $(S \cap \Sigma^q, M, \mathsf{d} = 2t+1)$ code, then, by the sphere-packing bound, we must have

$$\sum_{\mathbf{w} \in \mathcal{C}} |\mathcal{B}_X(\mathbf{w}; t)| \leq |X| \tag{9.1}$$

for any nonempty subset $X \subseteq \Sigma^q$. In the conventional sphere-packing bound, the subset X is taken to be the whole set Σ^q. Improved bounds may be obtained by taking X to be a proper subset of Σ^q. Specifically, define

$$N(S, X; i) = |\{\mathbf{w} \in S \cap \Sigma^q \, : \, |\mathcal{B}_X(\mathbf{w}; t)| = i\}| \, .$$

Now, if X is contained in $\bigcup_{\mathbf{w} \in S \cap \Sigma^q} \mathcal{B}_{\Sigma^q}(\mathbf{w}; t)$, then

$$\sum_{i=0}^{|X|} iN(S, X; i) \geq |X| \, ,$$

so there exists an integer j, $1 \leq j \leq |X|$, such that

$$\sum_{i=0}^{j-1} iN(S, X; i) < |X| \, ,$$

and

$$\sum_{i=0}^{j} iN(S, X; i) \geq |X| \, .$$

Abdel-Ghaffar and Weber [AW91] used these inequalities to establish the following upper bound on the code cardinality.

Theorem 9.9 *Let $\mathcal{C}$ be an $(S \cap \Sigma^q, M, \mathsf{d} = 2t+1)$ code and let X be a nonempty subset of $\bigcup_{\mathbf{w} \in S \cap \Sigma^q} \mathcal{B}_{\Sigma^q}(\mathbf{w}; t)$. Then*

$$|\mathcal{C}| \leq \sum_{i=0}^{j-1} N(S, X; i) + \left\lfloor \frac{|X| - \sum_{i=0}^{j-1} iN(S, X; i)}{j} \right\rfloor .$$

Proof. If $|\mathcal{C}| \leq \sum_{i=0}^{j-1} N(S, X; i)$, then we are done already. So, we may assume that $|\mathcal{C}| > \sum_{i=0}^{j-1} N(S, X; i)$. Divide $\mathcal{C}$ into two subsets $\mathcal{C}_1, \mathcal{C}_2$ where $\mathcal{C}_1$ consists of the $\sum_{i=0}^{j-1} N(S, X; i)$ elements $\mathbf{w}$ of $\mathcal{C}$ with the smallest $|\mathcal{B}_X(\mathbf{w}; t)|$. Then

$$\sum_{i=0}^{j-1} iN(S, X; i) \leq \sum_{\mathbf{w} \in \mathcal{C}_1} |\mathcal{B}_X(\mathbf{w}; t)|,$$

and

$$j(|\mathcal{C}| - \sum_{i=0}^{j-1} N(S, X; i)) \leq \sum_{\mathbf{w} \in \mathcal{C}_2} |\mathcal{B}_X(\mathbf{w}; t)|.$$

Now, use the preceding two inequalities to lower bound the left-hand side of inequality (9.1):

$$\sum_{i=0}^{j-1} iN(S, X; i) + j(|\mathcal{C}| - \sum_{i=0}^{j-1} N(S, X; i)) \leq \sum_{\mathbf{w} \in \mathcal{C}} |\mathcal{B}_X(\mathbf{w}; t)| \leq |X|.$$

The theorem follows from this. □

For the special case of bitshift errors, Abdel-Ghaffar and Weber obtain in [AW91] upper bounds on single-bitshift correcting codes $\mathcal{C}$ for (d, k)-RLL constrained systems S as follows. First, partition every code $\mathcal{C} \subseteq S \cap \Sigma^q$ into constant-weight subsets $\mathcal{C} = \cup_{\mathsf{w}} \mathcal{C}_{\mathsf{w}}$, such that each element of $\mathcal{C}_{\mathsf{w}}$ has Hamming weight w; then apply Theorem 9.9 to the subsets $\mathcal{C}_{\mathsf{w}}$, for suitably chosen sets X. Table 9.2 shows results for selected RLL constraints and codeword lengths.

Constructions of codes for channels with substitution, asymmetric, and bitshift errors, as well as bounds on the *maximum* cardinality of such codes of fixed length, have been addressed by numerous other authors, for example Blaum [Blaum91]; Ferreira and Lin [FL91]; Fredrickson and Wolf [FW64]; Immink [Imm91]; Kolesnik and Krachkovsky [KolK94]; Kuznetsov and Vinck [KuV93a], [KuV93b]; Lee and Wolf [Lee88], [LW87], [LW89]; Patapoutian and Kumar [PK92]; Shamai and Zehavi [SZ91]; and Ytrehus [Yt91a], [Yt91b].

q	$(d,k)=(2,7)$	$(d,k)=(3,10)$	$(d,k)=(4,12)$	$(d,k)=(5,15)$
3	1			
4	1	1		
5	1	1	1	
6	2	1	1	1
7	2	2	1	1
8	3	3	2	1
9	4	2	2	2
10	5	3	3	2
11	8	5	2	2
12	10	6	3	3
13	14	7	5	3
14	18	9	5	3
15	26	13	7	5
16	35	16	8	6
17	48	21	11	7
18	68	29	14	9
19	91	38	18	11
20	126	49	22	13
21	176	63	28	16
22	239	84	36	21
23	329	110	46	25
24	455	147	57	32
25	627	194	73	40
26	877	255	93	49
27	1204	335	117	61
28	1670	440	151	75
29	2302	581	193	95
30	3206	774	244	117
31	4464	1024	311	143
32	6182	1356	396	179

Table 9.2: Upper bounds on sizes of (d,k)-RLL constrained single shift-error correcting codes of length $3 \leq q \leq 32$.

9.4 Distance properties of spectral-null codes

Finally, we mention that spectral-null constrained codes—in particular, dc-free codes—with Hamming error-correction capability have received considerable attention. See, for example, Barg and Litsyn [BL91]; Blaum and van Tilborg [TiBl89]; Blaum, Litsyn, Buskens, and van Tilborg [BLBT93]; Calderbank, Herro, and Telang [CHT89]; Cohen and Litsyn [CL91]; Etzion [Etz90]; Ferreira [Fe84]; Roth [Roth93]; Roth, Siegel, and Vardy [RSV94]; Waldman and Nisenbaum [WN95]. Spectral-null codes also have inherent Hamming-distance properties, as shown by Immink and Beenker [ImmB87]. They considered codes over the alphabet $\{+1, -1\}$ in which the order-m moment of every codeword $\mathbf{x} = (x_1\ x_2\ \ldots\ x_{\mathsf{n}})$ vanishes for $m = 0, 1, \ldots, K-1$, i.e.,

$$\sum_{i=1}^{\mathsf{n}} i^m x_i = 0, \qquad m = 0, 1, \ldots, K-1 \ .$$

They referred to a code with this property as a code with order-$(K-1)$ zero-disparity. For each codeword $\mathbf{x}$, the discrete Fourier transform, given by $\Phi_{\mathbf{x}}(f) = \sum_{\ell=1}^{\mathsf{n}} x_\ell e^{-\jmath 2\pi f \ell}$, where $\jmath = \sqrt{-1}$, satisfies

$$\left.\frac{d^m \Phi_{\mathbf{x}}(f)}{df^m}\right|_{f=0} = 0 \qquad \text{for} \quad m = 0, 1, \ldots, K-1 \ .$$

This implies by part 8 of Problem 3.35 that the power spectral density of the ensemble of sequences generated by randomly concatenating codewords vanishes at $f = 0$, along with its order-ℓ derivatives for $\ell = 1, 2, \ldots, 2K-1$. A code, or more generally a constraint, with this property is said to have an *order-K spectral-null at $f = 0$*.

The following theorem, from [ImmB87], provides a lower bound on the minimum Hamming distance of a code with spectral null at $f = 0$.

Theorem 9.10 *Let $\mathcal{C}$ be a code with order-K spectral null at $f = 0$. Let $\mathbf{x}, \mathbf{y}$ be distinct codewords in $\mathcal{C}$. Then, their Hamming distance satisfies*

$$\Delta_{\text{Hamming}}(\mathbf{x}, \mathbf{y}) \geq 2K \ .$$

This result will play a role in the subsequent discussion of codes for the Lee and Euclidean metrics.

9.5 Synchronization/bitshift error correction

Synchronization errors, resulting from the insertion or deletion of symbols, and coding methods for protection against such errors have been the subject of numerous investigations. The

edit distance, introduced by Levenshtein and often referred to as the *Levenshtein metric*, is particularly appropriate in this setting, as it measures the minimum number of symbol insertions and deletions required to derive one finite-length sequence from another. The reader interested in codes based upon the Levenshtein metric is referred to Bours [Bours94]; Iizuka, Hasahara, and Namahawa [IKN80]; Kruskal [Krusk83]; Levenshtein [Lev65], [Lev67], [Lev71], [Lev91]; Levenshtein and Vinck [LV93]; Tanaka and Kasai [TK76]; Tenengolts [Ten76], [Ten84]; and Ullman [U66], [U67].

When dealing with synchronization errors (insertions and deletions of 0's) in (d, k)-RLL constrained systems, it is convenient to represent a constrained sequence as a sequence of "runs," where a run corresponds to a symbol 1 along with the subsequent string of contiguous symbols 0 preceding the next consecutive symbol 1. We associate to each run a positive integer called the "runlength" representing the number of symbols in the run. As an example, the $(1, 7)$-RLL sequence 10100000001000100 corresponds to the sequence of runs with runlengths 2, 8, 4, 3.

Let $\mathbf{w}$ be a (d, k)-constrained sequence with n runs and corresponding runlength sequence $\mathbf{s} = s_1, s_2, \ldots, s_{\mathsf{n}}$. Insertion of e symbols 0 in the jth run of $\mathbf{w}$ generates the sequence with runlengths $\mathbf{s}' = s_1, \ldots, s_j + e, s_{j+1}, \ldots, s_{\mathsf{n}}$, while deletion of e symbols 0 from run j generates the sequence of runlengths $\mathbf{s}' = s_1, \ldots, s_j - e, s_{j+1}, \ldots, s_{\mathsf{n}}$. (In the latter, e cannot exceed s_j.) An *e-synchronization error* denotes such a pattern of e insertions or deletions occurring within a single run. Note, also, that a bitshift error, or more generally, an e-bitshift error consisting of e left-bitshift errors or e right-bitshift errors occurring at the boundary between two adjacent runs, may be viewed as a pair of e-synchronization errors in consecutive runs, one being an insertion error, the other a deletion error.

This "runlength"-oriented viewpoint has been used in the design of RLL codes capable of detecting and correcting bitshift and synchronization errors. Hilden, Howe, and Weldon [Hild91] constructed a class of variable length codes, named Shift-Error-Correcting Modulation (SECM) codes, capable of correcting up to some prespecified number of random e-bitshift errors, for a preselected shift-error size e. The runlengths are regarded as elements of a finite alphabet F whose size, usually taken to be an odd prime integer, satisfies $k - d + 1 \geq |F| \geq 2e + 1$. The binary information string is viewed as a sequence of k runs $\mathbf{r} = r_1, r_2, \ldots, r_{\mathsf{k}}$, satisfying (d, k) constraints, with runlengths $\mathbf{s} = s_1, s_2, \ldots, s_{\mathsf{k}}$. The sequence of transition positions $\mathbf{t} = t_1, t_2, \ldots, t_{\mathsf{k}}$ is then defined by:

$$t_j = \sum_{i=1}^{j} s_i \pmod{|F|}, \quad \text{for} \quad j = 1, 2, \ldots, \mathsf{k}\,.$$

These values are then applied to a systematic encoder for an $[\mathsf{n}, \mathsf{k}, \mathsf{d}]$ BCH code over a finite field F of prime size, yielding parity symbols $t_{\mathsf{k}+1}, t_{\mathsf{k}+2}, \ldots, t_{\mathsf{n}}$. A (d, k)-constrained binary codeword is then generated by appending to the original information runs the sequence of parity runs $r_{\mathsf{k}+1}, r_{\mathsf{k}+2}, \ldots, r_{\mathsf{n}}$ whose runlengths $s_{\mathsf{k}+1}, s_{\mathsf{k}+2}, \ldots, s_{\mathsf{n}}$ satisfy

$$d + 1 \leq s_j < d + 1 + |F|\,, \quad \text{for} \quad j = \mathsf{k}{+}1, \mathsf{k}{+}2, \ldots, \mathsf{n}$$

and

$$t_j = \sum_{i=1}^{j} s_i \pmod{|F|}, \quad \text{for} \quad j = \mathsf{k}{+}1, \mathsf{k}{+}2, \ldots, \mathsf{n}\ .$$

In particular, for $|F| = 2t{+}1$, the resulting code may be used to correct up to t random 1-bitshift errors, where t is the designed error-correcting capability of the BCH code. Note that a similar construction provides for correction of random e-synchronization errors by encoding the runlengths themselves, rather than the transition positions. The interpretation of bitshift and, more generally, synchronization errors in terms of their effect on runlengths leads naturally to the consideration of another metric, the *Lee metric*.

The *Lee distance* $\Delta_{\rm Lee}(x, y)$ of two symbols x, y in a finite field F of prime size is the smallest absolute value of any integer congruent to the difference $x - y$ modulo $|F|$. For vectors $\mathbf{x}$, $\mathbf{y}$ in F^{n}, the Lee distance $\Delta_{\rm Lee}(\mathbf{x}, \mathbf{y})$ is the sum of the component-wise Lee distances. The *Lee weight* $w_{\rm Lee}(\mathbf{x})$ of a vector $\mathbf{x}$ is simply $\Delta_{\rm Lee}(\mathbf{x}, \mathbf{0})$, where $\mathbf{0}$ denotes the all-zero vector of length n.

Among the families of codes for the Lee-metric are the well-known negacyclic codes introduced by Berlekamp [Berl84, Ch. 9], the family of cyclic codes devised by Chiang and Wolf [CW71], and the Lee-metric BCH codes investigated by Roth and Siegel [RS92], [RS94].

All of these Lee-metric code constructions have the property that the redundancy required for correction of a Lee-metric error vector of weight t is approximately t symbols. In contrast, codes designed for the Hamming metric require approximately $2t$ check symbols to correct t random Hamming errors. In a recording channel subject to e-synchronization errors and e-bitshift errors, where the predominant errors correspond to small values of e, one might anticipate reduced overhead using a Lee-metric coding solution. This observation was made independently by Roth and Siegel [RS94], Saitoh [Sai93a], [Sai93b], and Bours [Bours94] (see also Davydov [Dav93] and Kabatiansky, Davydov, and Vinck [KDV92]), who have proposed a variety of constrained code constructions based on the Lee-metric, and have derived bounds on the efficiency of these constructions, as we now describe.

For bitshift-error correction, Saitoh proposed a construction yielding codes with fixed binary symbol length. He showed that the construction is asymptotically optimal with respect to a Hamming bound on the redundancy for single-bitshift error-correcting (d, k)-RLL codes.

The construction of Saitoh requires that the codewords begin with a symbol 1 and end with at least d symbols 0. The codewords will have a fixed number of runs and, consequently, a variable length in terms of binary symbols. The codewords are defined as follows. If the runlengths are denoted s_i, $i = 0, 1, \ldots, \mathsf{n}$, the sequence of runlengths s_i, for even values of i, comprise a codeword in a single-error correcting code over the Lee metric. The sequence of runlengths s_i, for $i \equiv 3 \pmod 4$, comprise a codeword in a single error-detecting code for the Lee metric. It is evident that, in the presence of a single bitshift error, the Lee-metric single error-correcting code will ensure correct determination of the runlengths s_i for even values of

i, indicating if the erroneous runlength, say s_{2j}, suffered an insertion or deletion of a symbol 0. The Lee-metric error-detecting code will then complete the decoding by determining if the corresponding deletion or insertion applies to runlength s_{2j-1} or s_{2j+1}.

In the broader context of synchronization errors, Roth and Siegel described and analyzed a construction of (d,k)-RLL codes for detection and correction of such errors as an application of a class of Lee-metric BCH codes [RS92]. The shortened BCH code of length n over a finite prime field F, denoted $\mathcal{C}(\mathsf{n},r;F)$, is characterized by the parity-check matrix

$$H(\mathsf{n},r;F) \equiv \begin{pmatrix} 1 & 1 & \dots & 1 \\ \beta_1 & \beta_2 & \dots & \beta_{\mathsf{n}} \\ \beta_1^2 & \beta_2^2 & \dots & \beta_{\mathsf{n}}^2 \\ \vdots & \vdots & \dots & \vdots \\ \beta_1^{r-1} & \beta_2^{r-1} & \dots & \beta_{\mathsf{n}}^{r-1} \end{pmatrix} ,$$

where $(\beta_1\ \beta_2\ \dots\ \beta_{\mathsf{n}})$ is the *locator vector*, consisting of distinct nonzero elements of the smallest h-dimensional extension field F_h of F of size greater than n.

Hence, a word $\mathbf{x} = (x_1\ x_2\ \dots\ x_{\mathsf{n}}) \in F^{\mathsf{n}}$ is in $\mathcal{C}(\mathsf{n},r;F)$ if and only if it satisfies the following r parity-check equations over F_h:

$$\sum_{i=1}^{\mathsf{n}} x_i \beta_i^m = 0 , \quad \text{for} \quad m = 0,1,\dots,r-1 .$$

The following theorem provides a lower bound on the minimum Lee distance of $\mathcal{C}(\mathsf{n},r;F)$, denoted $\mathsf{d}_{\mathrm{Lee}}(\mathsf{n},r;F)$.

Theorem 9.11

$$\mathsf{d}_{\mathrm{Lee}}(\mathsf{n},r;F) \geq \begin{cases} 2r & \text{for } r \leq (|F|-1)/2 \\ |F| & \text{for } (|F|+1)/2 \leq r < |F| \end{cases} .$$

This bound follows from Newton's identities [ImmB87],[KS91a] and can be regarded, in a way, as the analogue of the BCH lower bound $r+1$ on the minimum Hamming distance of $\mathcal{C}(\mathsf{n},r;F)$, although the proof of the $2r$ lower bound is slightly more complicated. For $r \geq |F|$ we can bound $\mathsf{d}_{\mathrm{Lee}}(\mathsf{n},r;F)$ from below by the minimum Hamming distance $r+1$.

The $2r$ lower bound does not hold in general for all values of r; however, it does hold for all r in the base-field case $\mathsf{n} \leq |F|-1$. The $2r$ lower bound for the base-field case takes the following form.

Theorem 9.12 *For $r \leq n \leq |F|-1$,*

$$\mathsf{d}_{\mathrm{Lee}}(\mathsf{n},r;F) \geq 2r .$$

The *primitive* case corresponds to codes $\mathcal{C}(\mathsf{n}, r; F)$ for which $\mathsf{n} = |F|^h - 1$. The redundancy of such codes is known to be bounded from above by $1 + (r-1)h$. This bound, along with the following lower bound derived by a sphere-packing argument, combine to show that the primitive codes are near-optimal for sufficiently small values of r.

Lemma 9.13 (Sphere-packing bound, Golomb and Welch [GoW68], [GoW70]) *A code over a finite prime field F of length n, size M, and minimum Lee distance $\geq 2r-1$ for some $r \leq (|F|+1)/2$ must satisfy the inequality*

$$M \cdot \sum_{i=0}^{r-1} 2^i \binom{n}{i} \binom{r-1}{i} \leq |F|^{\mathsf{n}} \ .$$

Theorem 9.14 *A code over a finite prime field F of length n, size M, and minimum Lee distance $\geq 2r-1$ for some $r \leq (|F|+1)/2$ must satisfy the inequality*

$$(r-1)\left(\log_{|F|}(\mathsf{n}-r+2) - \log_{|F|}(r-1)\right) \leq \mathsf{n} \ - \ \log_{|F|} M \ .$$

Proof. By Lemma 9.13 we have

$$\frac{(\mathsf{n}-r+2)^{r-1}}{(r-1)^{r-1}} \cdot 2^{r-1} \leq |F|^{\mathsf{n}}/M \ .$$

The theorem now follows by taking the logarithm to base $|F|$ of both sides of this inequality. □

The construction of synchronization-error correcting codes based upon the Lee-metric BCH codes is as follows. Given constraints (d, k), we choose $|F| \leq k-d+1$. We regard every run of length s in the (d, k)-constrained information sequence as an element $(s-d-1) \pmod{|F|}$ of F, and use a systematic encoder for $\mathcal{C}(\mathsf{n}, r; F)$ to compute the corresponding check symbols in F. Each check symbol a, in turn, is associated with a run of length $\overline{a} + d + 1$, where $\overline{a}$ is the smallest nonnegative integer such that $a = \overline{a} \cdot 1$, where 1 stands for the multiplicative unity in F. The code $\mathcal{C}(\mathsf{n}, r; F)$, with $r \leq (|F|-1)/2$ and $\mathsf{n} \leq |F|^h - 1$ can simultaneously correct b bitshift errors and s non-bitshift synchronization errors whenever $2b + s < r$. (Observe that, when counting errors, an e-bitshift error is counted as e bitshift errors; this applies respectively also to synchronization errors. Also, bitshift or synchronization errors may create runlengths that violate the (d, k)-constraint. In such a case we can mark the illegal runlength as an erasure rather than an error.) The redundancy required will be no more than $1 + (r-1)h$ symbols from the alphabet F, and we recall that Theorem 9.14 indicates the near-optimality of the Lee-metric primitive BCH codes $\mathcal{C}(|F|^h - 1, r; F)$, for values $r \ll |F|^h - 1$.

Example 9.2 Two typical choices for parameters (d, k) are $(1, 7)$ and $(2, 8)$, both satisfying $k - d + 1 = 7$. Setting $|F| = 7$ and $r = 3$, we obtain a family of codes for these

constraints, based upon $\mathcal{C}(\mathsf{n}, 3; 7)$, that can correct any error pattern of Lee weight 2 (and detect error patterns of Lee weight 3). In particular, the codes will correct one single-bitshift (1-bitshift) error or any other combination of two insertions/deletions of symbols 0. For $\mathsf{n} \leq |F|^h - 1$, the required redundancy is no more than $1 + 2h$ symbols. □

As mentioned above, the class of Hamming-metric SECM codes are directed primarily toward the situation when only bitshift-type errors occur. The constructions based upon Lee-metric codes can be modified to improve their efficiency in this type of error environment by recording, instead of the nominal codeword $\mathbf{x} = (x_1\ x_2\ \ldots\ x_{\mathsf{n}})$, the differentially precoded word $\mathbf{y} = (y_1\ y_2\ \ldots\ y_{\mathsf{n}})$ defined by $y_1 = x_1$ and $y_i = x_i - x_{i-1}$ for $2 \leq i \leq \mathsf{n}$, where all operations are taken modulo $|F|$. If $\mathbf{y}$ is recorded, and no bitshift errors occur, the original word $\mathbf{x}$ is reconstructed by an "integration" operation:

$$x_i = \sum_{j=1}^{i} y_j \ .$$

If, however, an e-bitshift error occurs at the boundary between runs j and $j{+}1$ of $\mathbf{y}$, the integration operation converts the error into an e-synchronization error in run j of $\mathbf{x}$. In other words, the original bitshift error pattern of Lee weight $2e$ is converted into a synchronization error pattern of Lee weight e. In order to ensure the correctness of the first run y_1, it suffices to require that the code contain the all-one word $(1\ 1\ \ldots\ 1)$ and all of its multiples.

For the Lee-metric BCH codes, this construction provides the capability to correct up to $r{-}1$ bitshift errors and detect up to r bitshift errors, when $2r < |F| \leq k{-}d{+}1$. The construction can be extended to the base-field case as well.

Example 9.3 Let $|F| = 7$ and $r = 3$ as in the previous example. The construction above will generate codes with length n a multiple of 7. For $\mathsf{n} = 7$, the redundancy is $1+(r{-}1) = 3$ runs; for $\mathsf{n} = 14, 21, \ldots, 49$ the redundancy is $1+2(r{-}1) = 5$ runs; for $\mathsf{n} = 56, 63, \ldots, 343$ the redundancy is $1 + 3(r{-}1) = 7$ runs. All of these codes will correct up to two single-bitshift errors or one double-bitshift (2-bitshift) error. By way of comparison, in [Hild91] Hilden et al. describe SECM codes of lengths 26, 80, and 242 for correcting two single-bitshift errors, requiring redundancy of 7, 9, and 11 runs, respectively. These SECM codes do not handle double-bitshift errors. □

Example 9.4 As $|F|$ increases, so does the discrepancy in the number of check symbols (runs) compared to the SECM codes in [Hild91]. For $|F| = 11$, suitable for representing $(d, k) = (1, 11)$ for example, and $r = 5$, the Lee-metric BCH code with $\mathsf{n} = 11$ requires 5 check symbols; for $\mathsf{n} = 22, 33, \ldots, 121$, the redundancy is 9 symbols; for $\mathsf{n} = 132, 143, \ldots, 1331$ the redundancy will be 13 symbols. These codes will correct up to four single-bitshift errors; two single-bitshift and one double-bitshift errors; or two double-bitshift errors. The codes presented in [Hild91] for correcting up to four single-bitshift errors have lengths 26, 80, and 242 and require redundancy of 16, 21, and 26, respectively. □

Bours [Bours94] provided a construction of synchronization-error correcting RLL codes with *fixed length* over the binary alphabet that also relies on an underlying Lee-metric code. He did not require the underlying code to be a Lee-metric BCH code, however, and thereby avoided having the error-correction capability limited by the code alphabet size.

The definition of the Lee metric can also be generalized in a straightforward manner to integer rings. Orlitsky described in [Or93] a nonlinear construction of codes over the ring of integers modulo 2^h for correcting any prescribed number of Lee errors. His construction is based on dividing a codeword of a binary BCH code into nonoverlapping h-tuples and regarding the latter as the Gray-code representations of the integers between 0 and $2^h - 1$.

It is also worth remarking that all of the Lee-metric codes mentioned above can be efficiently decoded algebraically.

We close the discussion of Lee-metric codes by noting that the definition of the class of Lee-metric BCH codes was motivated by a Lee-metric generalization of the result of Immink and Beenker in Theorem 9.10 to integer-valued spectral-null constraints [KS91a], [EC91].

Theorem 9.15 *Let S be a constrained system over an integer alphabet with order-K spectral null at $f = 0$, presented by a labeled graph G. Let $\mathbf{x}, \mathbf{y}$ be distinct sequences in S generated by paths in G, both of which start at a common state u and end at a common state v. Then, the Lee distance satisfies*

$$\Delta_{\mathrm{Lee}}(\mathbf{x}, \mathbf{y}) \geq 2K \ .$$

This result will play an important role in the next section in the context of Euclidean-metric codes for PRML.

When combining bitshift and synchronization errors, any bitshift error can obviously be regarded as two consecutive synchronization errors in opposite directions – one e-insertion, one e-deletion – thus reducing to the synchronization-only model of errors. However, such an approach is not optimal, and better constructions have been obtained to handle a limited number of bitshift and synchronization errors (combined). See Hod [Hod95], Kløve [Kl95], and Kuznetsov and Vinck [KuV93a], [KuV93b].

9.6 Soft-decision decoding through Euclidean metric

Let $\mathbf{x}$ and $\mathbf{y}$ be sequences of length n over the real numbers. The *squared-Euclidean distance* between these sequences, denoted $\Delta^2_{\mathrm{Euclid}}(\mathbf{x}, \mathbf{y})$ is given by

$$\Delta^2_{\mathrm{Euclid}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{\mathsf{n}} (x_i - y_i)^2 \ .$$

The Euclidean metric is most relevant in channels with additive white Gaussian noise (AWGN). In particular, it is of interest in connection with the model of the magnetic recording channel as a binary input, partial-response system with AWGN. The success of trellis-coded modulation, as pioneered by Ungerboeck, in improving the reliability of memoryless channels with AWGN provided the impetus to design coding schemes for channels with memory, such as partial-response channels, in AWGN. For binary input-restricted partial-response channels suitable as models for recording channels, such as the Class-4 channel characterized by the input-output relation $y_i = x_i - x_{i-2}$, several approaches have been proposed that make use of binary convolutional codes. These approaches typically require a computer search of some kind to determine the codes that are optimal with respect to rate, Euclidean distance, and maximum-likelihood detector complexity. See, for example, Wolf and Ungerboeck [WU86]; Calderbank, Heegard, and Lee [CHL86], Hole [Hole91]; and Hole and Ytrehus [HoY94].

There is another approach, however, that relies upon the concepts and code construction techniques that have been developed in the previous chapters. The underlying idea is as follows. First, find a constrained system S, presented by a labeled graph G, that ensures a certain minimum Euclidean distance between the partial-response channel output sequences generated when channel inputs are restricted to words in S. Then, apply state-splitting (or other) methods to construct an efficient encoder from binary sequences to S. Since the graph structure $\mathcal{E}$ underlying the encoder is often more complex (in terms of number of states and interconnections) than the original graph G, use G rather than $\mathcal{E}$ as the starting point for the trellis-based Viterbi detector of the coded channel.

Karabed and Siegel [KS91a] showed that this approach can be applied to the family of constrained systems S whose spectral null frequencies coincide with those of the partial-response channel transfer function. The resulting codes are referred to as *matched-spectral-null codes*. We conclude this section with a brief summary of the results that pertain to the application of this technique to the Class-4 and related partial-response systems. We will refer to the *dicode* channel, which is characterized by the input-output relation $y_i = x_i - x_{i-1}$, and has a first-order spectral null at $f = 0$, and we remark that the Class-4 partial-response channel may be interpreted as a pair of interleaved dicode channels, one operating on inputs with even indices, the other on inputs with odd indices.

Lemma 9.16 *Let S be a constrained system over an integer alphabet with order-K spectral null at zero frequency. Let S' be the constrained system of sequences at the output of a cascade of N dicode channels, with inputs restricted to words in S. Then, S' has an order-$(K+N)$ spectral null at zero frequency.*

Noting that any lower bound on Lee distance provides a lower bound on squared-Euclidean distance for sequences over integer alphabets, we obtain from the preceding lemma and Theorem 9.15 the following lower bound on the minimum squared-Euclidean distance of a binary, matched-spectral-null coded, partial-response channel with spectral null at $f = 0$.

Theorem 9.17 *Let S be a constrained system over the alphabet $\{+1, -1\}$, with order-K spectral null at zero frequency. Let $\mathbf{x}$ and $\mathbf{y}$ be distinct sequences in S, differing in a finite number of positions. If $\mathbf{x}'$ and $\mathbf{y}'$ are the corresponding output sequences of a partial-response channel consisting of a cascade of N dicode channels, then*

$$\Delta^2_{\text{Euclid}}(\mathbf{x}, \mathbf{y}) \geq 8(K + N) \ .$$

It is easy to see that the lower bound of Theorem 9.17 remains valid in the presence of J-way, symbol-wise interleaving of the constrained sequences and the partial-response channel. In particular, for the Class-4 partial-response channel (i.e., the 2-way interleaved dicode channel), the application of sequences obtained by 2-way interleaving a code having a first-order spectral null at zero frequency doubles the minimum squared-Euclidean distance at the channel output, relative to the uncoded system.

We remark that J-way interleaving of sequences with an order-K spectral null at $f = 0$ generates sequences with order-K spectral nulls at frequencies $f = r/J$, for $r = 0, 1, \ldots, J-1$ [MS87]. Thus, a 2-way interleaved, dc-free code has spectral nulls at zero frequency and at frequency $f = 1/2$, corresponding to the spectral null frequencies of the Class-4 partial-response channel.

Graph presentations for spectral null sequences are provided by *canonical diagrams*, introduced by Marcus and Siegel [MS87] for first-order spectral null constraints and then extended to high-order constraints by, among others, Monti and Pierobon [MPi89]; Karabed and Siegel [KS91a]; Eleftheriou and Cideciyan [EC91]; and Kamabe [Kam94].

Discussion of the canonical diagrams requires the notion of a labeled graph with an infinite number of states. Specifically, a *countable-state labeled graph* $G_\infty = (V, E, L)$ consists of a countably-infinite set of states V; a set of edges E, where each edge has an initial state and a terminal state, both in V, and the states in V have bounded out-degree and in-degree; and an edge labeling $L : E \to \Sigma$, where Σ is a finite alphabet.

We say that a countable-state graph G_∞ is a *period-*p *canonical diagram* for a spectral null constraint if:

1. Every finite subgraph $H \subset G_\infty$ generates a set of sequences with the prescribed spectral null constraint.

2. For any period-p graph G'_∞ that presents a system with the specified spectral null constraint, there is a label-preserving graph homomorphism of G'_∞ into G_∞, meaning a map from the edges of G'_∞ to the edges of G_∞ that preserves initial states, terminal states, and labels.

The canonical diagram G_∞ for a first-order spectral null constraint at zero frequency, with symbol alphabet $\{+1, -1\}$, is shown in Figure 9.1. As mentioned in Example 3.4,

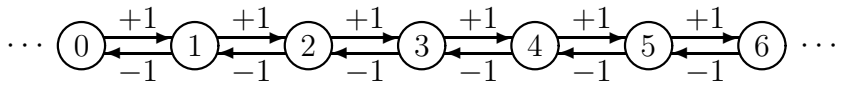


Figure 9.1: Canonical diagram for first-order spectral null at $f = 0$.

the capacity of the constrained system generated by a subgraph G_B, consisting of $B+1$ consecutive states and the edges with beginning and ending states among these, is given by

$$\mathsf{cap}(S(G_B)) = \log\left(2\cos\frac{\pi}{B+2}\right) .$$

From this expression, it follows that

$$\lim_{B\to\infty} \mathsf{cap}(S(G_B)) = 1 .$$

We pointed out in Chapter 2 that the constrained system generated by any finite subgraph of G_∞ is almost-finite-type. Applying Theorem 4.12, we see that by choosing B large enough, we can construct a sliding-block decodable finite-state encoder for the constrained system $S(G_B)$ at any prespecified rate $p : q$ with $p/q < 1$.

From the structure of the canonical diagram, it is clear that any constrained system with first-order spectral null at $f = 0$ limits the number of consecutive zero samples at the output of the dicode channel. When the constrained sequences are twice-interleaved and applied to the Class-4 partial-response channel, the number of zero samples at the output is limited "globally" as well as in each of the even/odd interleaved subsequences. This condition is analogous to that achieved by the $(0, \mathbf{G/I})$ constraints for the baseline PRML system.

The subgraph G_B chosen for the code construction may be augmented to incorporate the dicode channel memory, as shown in Figure 9.2 for the case $M = 6$, providing the basis for a dynamic programming (Viterbi) detection algorithm for the coded-dicode channel with AWGN. Each state in the trellis has a label of the form v^m, where v is the state in Figure 9.1 from which it is derived, and the superscript m denotes the sign of the dicode channel memory. Just as does the uncoded dicode detector graph, represented by the trellis in Figure 1.20 of Chapter 1, the coded-dicode detector graph supports sequences that can cause potentially unbounded delays in the merging of survivor sequences and, therefore, in decoding. The spectral-null code sequences that generate these output sequences are called *quasicatastrophic sequences*, and they are characterized in the following proposition.

Proposition 9.18 *The quasicatastrophic sequences in the constrained system presented by G_B are those generated by more than one path in G_B.*

To limit the merging delay, the matched-spectral-null code is designed to avoid these sequences, and it is shown in [KS91a] that this is always possible without incurring a rate loss for any G_B, with $B \geq 3$.

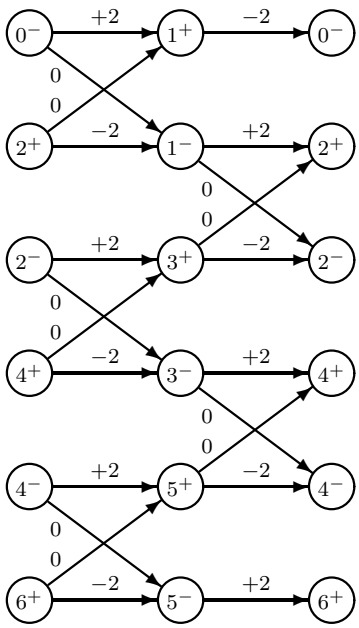


Figure 9.2: Graph underlying coded-dicode channel Viterbi detector for G_6.

Further details and developments regarding the design and application of matched-spectral-null codes to PRML systems may be found in [Shung91], [Thap92], [Thap93], [Fred94], and [Rae94].

9.7 Forbidden list codes for targeted error events

This section (which is yet to be written) will be based on results taken from Karabed-Siegel-Soljanin [KSS00].

Problems

Problem 9.1 A graph G is called *binary* if its labels are over the alphabet $\{0, 1\}$. The *(Hamming) weight* of a word $\mathbf{w}$ generated by a binary graph G is the number of 1's in $\mathbf{w}$. The weight of a path in a binary graph G is the weight of the word generated by that path.

Given a binary graph G and states u and v in G, denote by $\tau^{(\ell)}_{u,v,k}$ the number of paths of length

ℓ and weight k in G that originate in u and terminate in v. For states u and v in G, define the *length-ℓ weight-distribution polynomial* (of paths from u to v), in the indeterminate z, by

$$P_{u,v}^{(\ell)}(z) = \sum_{k=0}^{\ell} \tau_{u,v,k}^{(\ell)} z^k \ .$$

As an example, for the graph H in Figure 9.3, $P_{A,C}^{(4)} = z^2 + 2z^3$, since there are three paths of length 4 that originate in A and terminate in C: one path has weight 2, and the other paths each has weight 3.

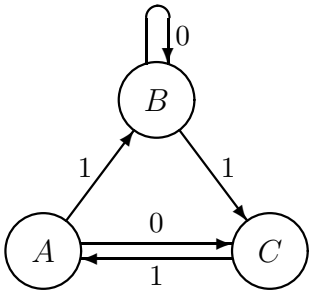


Figure 9.3: Graph H for Problem 9.1.

For a binary graph $G = (V, E, L)$, let $B_G(z)$ be the $|V| \times |V|$ matrix in the indeterminate z, where

$$(B_G(z))_{u,v} = P_{u,v}^{(1)}(z)$$

for every $u, v \in V$. Each entry in $B_G(z)$ is therefore a polynomial in z of degree at most 1.

1. Compute $B_H(z)$ for the graph in Figure 9.3.
2. For the matrix $B_H(z)$ found in 1, compute $(B_H(z))^2$ and $(B_H(z))^4$.
3. Let $B_G(z)$ be the matrix associated with a binary graph G, and let u and v be states in G. Given a positive integer ℓ, obtain an expression for the polynomial $P_{u,v}^{(\ell)}(z)$ in terms of $B_G(z)$.
4. Identify the matrix $B_G(1)$ associated with a binary graph G.
5. Identify the matrix $B_G(0)$.
6. Let G be a binary graph and let z_0 be a positive real number. Show that G is irreducible if and only if the matrix $B_G(z_0)$ is irreducible. Does this hold also when $z_0 = 0$?

Problem 9.2 Recall the definitions from Problem 9.1. Let G be a binary *lossless* graph and let u and v be states in G. For positive integers ℓ and d, denote by $J_{u,v,d}^{(\ell)}$ the number of words of length ℓ and weight $\leq d$ that can be generated in G by paths that originate in u and terminate in v.

1. Show that for every $0 \le d \le \ell$,
$$J^{(\ell)}_{u,v,d} = \sum_{k=0}^{d} \tau^{(\ell)}_{u,v,k} \ .$$

2. Based on 1, show that for every real z in the range $0 < z \le 1$,
$$J^{(\ell)}_{u,v,d} \le \sum_{k=0}^{\ell} \tau^{(\ell)}_{u,v,k} z^{k-d}$$
and, therefore,
$$J^{(\ell)}_{u,v,d} \le \min_{0 \le z \le 1} z^{-d} P^{(\ell)}_{u,v}(z) \ .$$

3. Based on 2 and 3, derive an upper bound on the number of words of length ℓ and weight $\le d$ in $S(G)$, as a function of $B_G(z)$, ℓ, and d.

Problem 9.3 Recall the definitions from Problem 9.1. Let S be a constrained system presented by a deterministic binary graph G with finite memory $\mathcal{M}$. For nonnegative integers ℓ and k, denote by $Y^{(\ell)}_k$ the number of (ordered) pairs $(\mathbf{w}, \mathbf{w}')$ of words of length ℓ in S such that $\mathbf{w}$ and $\mathbf{w}'$ are at Hamming distance k; i.e., they differ on exactly k locations. Define the polynomial $Y^{(\ell)}(z)$ by
$$Y^{(\ell)}(z) = \sum_{k=0}^{\ell} Y^{(\ell)}_k z^k \ .$$

1. Show that $Y^{(\ell)}(0) = |S \cap \{0,1\}^\ell|$.

2. Show that $Y^{(\ell)}(1) = (Y^{(\ell)}(0))^2$.

3. Let $S_{0,1}$ denote the $(0,1)$-RLL constrained system. Show that when $S = S_{0,1}$, the polynomial $Y^{(\ell)}(z)$ can be written as
$$Y^{(\ell)}(z) = (1 \ z \ z \ 1) \left(B_{G*G}(z)\right)^{\ell-1} \mathbf{1} \ ,$$
where $\mathbf{1}$ is the all-one column vector and
$$B_{G*G}(z) = \begin{pmatrix} 1 & z & z & 1 \\ 1 & 0 & z & 0 \\ 1 & z & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \ .$$

4. Compute $Y_4(z)$ explicitly for $S = S_{0,1}$.

5. Find the largest integer n for which there exists a block $(S^4_{0,1}, n)$-encoder whose codewords are at Hamming distance at least 2 from each other.

6. Generalize 3 for any constrained system S over $\{0,1\}$ with finite memory $\mathcal{M}$.

Bibliography

[AW91] K.A.S. Abdel-Ghaffar, J.H. Weber, *Bounds and constructions for run-length limited error-control block codes, IEEE Trans. Inform. Theory,* 37 (1991), 789–800.

[AbS65] M. Abramowitz, I.A. Stegun, *Handbook of Mathematical Functions,* Dover Publications, New York, 1965.

[Ad87] R.L. Adler, *The torus and the disk, IBM J. Res. Develop.,* 31 (1987), 224–234.

[ACH83] R.L. Adler, D. Coppersmith, M. Hassner, *Algorithms for sliding block codes — an application of symbolic dynamics to information theory, IEEE Trans. Inform. Theory,* 29 (1983), 5–22.

[AFKM86] R.L. Adler, J. Friedman, B. Kitchens, B.H. Marcus, *State splitting for variable-length graphs, IEEE Trans. Inform. Theory,* 32 (1986), 108–113.

[AGW77] R.L. Adler, L.W. Goodwyn, B. Weiss, *Equivalence of topological Markov shifts, Israel J. Math.,* 27 (1977), 49–63.

[AHM82] R.L. Adler, M. Hassner, J. Moussouris, *Method and apparatus for generating a noiseless sliding block code for a* $(1,7)$ *channel with rate 2/3,* US patent 4,413,251 (1982).

[AHU74] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, Reading, Massachusetts, 1974.

[Ari90] E. Arikan, *An implementation of Elias coding for input-restricted channel, IEEE Trans. Inform. Theory,* 36 (1990), 162–165.

[Ash87a] J.J. Ashley, *On the Perron-Frobenius eigenvector for non-negative integral matrices whose largest eigenvalue is integral, Linear Algebra Appl.,* 94 (1987), 103–108.

[Ash87b] J.J. Ashley, *Performance Bounds in Constrained Sequence Coding,* Ph.D. Thesis, University of California, Santa Cruz, 1987.

[Ash88] J.J. Ashley, *A linear bound for sliding block decoder window size, IEEE Trans. Inform. Theory,* 34 (1988), 389–399.

[Ash92] J.J. Ashley, *LR conjugacies of shifts of finite type are uniquely so, Contemp. Math.,* 135 (1992), 57–84.

[Ash93] J.J. Ashley, *An extension theorem for closing maps of shifts of finite type, Trans. AMS,* 336 (1993), 389–420.

[Ash96] J.J. Ashley, *A linear bound for sliding block decoder window size, II, IEEE Trans. Inform. Theory,* 42 (1996), 1913–1924.

[AB94] J.J. Ashley, M.-P. Béal, *A note on the method of poles for code construction, IEEE Trans. Inform. Theory,* 40 (1994).

[AJMS99] J.J. Ashley, G. Jaquette, B. Marcus, P. Seger, *Run length limited encoding/decoding with robust resync,* U.S. Patent 5,969,649 (1999).

[AKS96] J.J. Ashley, R. Karabed, P.H. Siegel, *Complexity and sliding block decodability, IEEE Trans. Inform. Theory,* 42 (1996), 1925–1947.

[AM95] J.J. Ashley, B.H. Marcus, *Canonical encoders for sliding block decoders, SIAM J. Discrete Math.,* 8 (1995), 555-605.

[AM98] J. J. Ashley, B. H. Marcus, *Two-dimensional lowpass filtering codes for holographic storage, IEEE Trans. Commun.,* 46 (1998), 724–727.

[AM97] J. Ashley and B. Marcus, *A generalized state splitting algorithm, IEEE Trans. Inform. Theory,* 43 (1997), 1326–1338.

[AM00] J. Ashley and B. Marcus, *Time-varying encoders for constrained systems: an approach to limiting error propagation, IEEE Trans. Inform. Theory,* 46 (2000), 1038–1043.

[AMR95] J.J. Ashley, B.H. Marcus, R.M. Roth, *Construction of encoders with small decoding look-ahead for input-constrained channels, IEEE Trans. Inform. Theory,* 41 (1995), 55–76.

[AMR96] J.J. Ashley, B.H. Marcus, R.M. Roth, *On the decoding delay of encoders for input-constrained channels, IEEE Trans. Inform. Theory,* 42 (1996), 1948–1956.

[AS87] J.J. Ashley, P.H. Siegel, *A note on the Shannon capacity of runlength-limited codes, IEEE Trans. Inform. Theory,* 33 (1987), 601–605.

[AHPS93] J.J. Ashley, M. Hilden, P. Perry, P.H. Siegel, *Correction to 'A note on the Shannon capacity of runlength-limited codes', IEEE Trans. Inform. Theory,* 39 (1993), 1110–1112.

[BL91] A.M. Barg, S.N. Litsyn, *DC-constrained codes from Hadamard matrices, IEEE Trans. Inform. Theory,* 37 (1991), 801–807.

[Béal90a] M.-P. Béal, *The method of poles: a coding method for constrained channels, IEEE Trans. Inform. Theory,* IT-36 (1990), 763–772.

[Béal90b] M.-P. Béal, *La méthode des pôles dans le cas des systèmes sofiques,* preprint, LITP Report, 29 (1990).

[Béal93a] M.-P. Béal, *Codage Symbolique,* Masson, Paris, 1993.

[Béal93b] M.-P. Béal, *A new optimization condition in the method of poles for code construction,* preprint, 1993.

[BI83] G.F.M. Beenker, K.A.S. Immink, *A generalized method for encoding and decoding run-length-limited binary sequences, IEEE Trans. Inform. Theory,* 29 (1983), 751–754.

[Berg96] J.W.M. Bergmans, *Digital Baseband Transmission and Recording,* Kluwer Academic Publishers, The Netherlands, 1996.

[Berl80] E.R. Berlekamp, *Technology of error-correcting codes, Proc. IEEE,* 68 (1980), 564–593.

[Berl84] E.R. Berlekamp, *Algebraic Coding Theory,* Revised Edition, Aegean Park Press, Laguna Hills, California, 1984.

[Blah83] R.E. Blahut, *Theory and Practice of Error-Control Codes,* Addison-Wesley, Reading, Massachusetts, 1983.

[BM75] I.F. Blake, R.C. Mullin, *The Mathematical Theory of Coding,* Academic Press, New York, 1975.

[Blaum91] M. Blaum, *Combining ECC with modulation: performance comparisons, IEEE Trans. Inform. Theory,* 37 (1991), 945–949.

[BLBT93] M. Blaum, S. Litsyn, V. Buskens, H.C.A. van Tilborg, *Error-correcting codes with bounded running digital sum, IEEE Trans. Inform. Theory,* 39 (1993), 216–227.

[Bli81] W.G. Bliss, *Circuitry for performing error correction calculations on baseband encoded data to eliminate error propagation, IBM Tech. Discl. Bull.,* 23 (1981), 4633–4634.

[BLG97] E. Boltt, Y-C Lai, and C. Grebogi, *Coding, channel capacity and noise resistance in communicating with chaos", Physics Review Letters,* 79 (1997), 3787–3790.

[Bours94] P.A.H. Bours, *Codes for correcting insertion and deletion errors,* Ph.D. dissertation, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands, June 1994.

[Bouw85] G. Bouwhuis, J. Braat, A. Huijser, J. Pasman, G. van Rosmalen, K.A.S. Immink, *Principles of Optical Disc Systems,* Adam Hilger, Bristol and Boston, 1985.

[BKM85] M. Boyle, B. Kitchens, B.H. Marcus, *A note on minimal covers for sofic systems, Proc. AMS,* 95 (1985), 403–411.

[BMT87] M. Boyle, B.H. Marcus, P. Trow, *Resolving maps and the dimension group for shifts of finite type, Memoirs AMS,* 377 (1987).

[BM99] G. W. Burr and B. H. Marcus, *Coding tradeoffs for high-density holographic data storage, Proc. SPIE*, Vol. 3802-06 (1999), 18–29.

[CHL86] A.R. Calderbank, C. Heegard, T.-A. Lee, *Binary convolutional codes with application to magnetic recording, IEEE Trans. Inform. Theory,* 32 (1986), 797–815.

[CHT89] A.R. Calderbank, M.A. Herro, V. Telang, *A multilevel approach to the design of dc-free codes, IEEE Trans. Inform. Theory,* 35 (1989), 579–583.

[CW71] J.C.-Y. Chiang, J.K. Wolf, *On channels and codes for the Lee metric, Inform. Control,* 19 (1971), 159–173.

[C70] T.M. Chien, *Upper bound on the efficiency of DC-constrained codes, Bell Syst. Tech. J.,* (1970), 2267–2287.

[Cid92] R. Cideciyan, F. Dolivo, R. Hermann, W. Hirt, W. Schott, *A PRML system for digital magnetic recording, IEEE J. Sel. Areas Commun.,* 10 (1992), 38–56.

[CL91] G.D. Cohen, S.N. Litsyn, *DC-constrained error-correcting codes with small running digital sum, IEEE Trans. Inform. Theory,* 37 (1991), 801–807.

[Cov73] T.M. Cover, *Enumerative source encoding, IEEE Trans. Inform. Theory,* IT-19 (1973), 73–77.

[Dav93] V.A. Davydov, *Error correcting codes in module metric, Lee metric, and operator errors, Problemy Perdachi Informatssii,* 29 (1993) 10–20 (in Russian).

[Dol89] F. Dolivo, *Signal processing for high density digital magnetic recoding,* in *Proc. COMPEURO'89,* Hamburg, Germany, 1989.

[DMU79] F. Dolivo, D. Maiwald, G. Ungerboeck, *Partial-response class-IV signaling with Viterbi decoding versus conventional modified frequency modulation in magnetic recording,* IBM Res. Zurich Lab., IBM Res. Rep. RZ 973–33865 (1979).

[EH78] J. Eggenberger, P. Hodges, *Sequential encoding and decoding of variable length, fixed rate data codes,* US patent 4,115,768 (1978).

[EC91] E. Eleftheriou, R. Cideciyan, *On codes satisfying Mth order running digital sum constraints, IEEE Trans. Inform. Theory,* 37 (1991), 1294–1313.

[Etz90] T. Etzion, *Constructions of error-correcting DC-free block codes, IEEE Trans. Inform. Theory,* 36 (1990), 899–905.

[Even65] S. Even, *On information lossless automata of finite order, IEEE Trans. Elect. Comput.,* 14 (1965), 561–569.

[Even79] S. Even, *Graph Algorithms,* Computer Science Press, Potomac, Maryland, 1979.

[FC98] J. Fan and R. Calderbank, *A modified concatenated coding scheme, with applications to magnetic data storage, IEEE Trans. on Inform. Theory,* 44 (1998), 1565–1574.

[FMR00] J. Fan, B. Marcus and R. Roth, *Lossles sliding-block compression of constrained systems, IEEE Trans. Inform. Theory,* 46 (2000), 624–633.

[Ferg72] M.J. Ferguson, *Optimal reception for binary partial response channels, Bell Sys. Tech. J.,* 51 (1972), 493–505.

[Fe84] H.C. Ferreira, *Lower bounds on the minimum-Hamming distance achievable with runlength constrained or DC-free block codes and the synthesis of a* $(16, 8)$ $d_{\min} = 4$ *DC-free block code, IEEE Trans. Magnetics,* 20 (1984), 881–883.

[FL91] H.C. Ferreira, S. Lin, *Error and erasure control* (d, k) *block codes, IEEE Trans. Inform. Theory,* 37 (1991), 1399–1408.

[Fi75a] R. Fischer, *Sofic systems and graphs, Monats. fur Math.* 80 (1975), 179–186.

[Fi75b] R. Fischer, *Graphs and symbolic dynamics, Colloq. Math. Soc. János Bólyai, Topics in Information Theory,* 16 (1975), 229–243

[For72] G.D. Forney, Jr., *Maximum likelihood sequence detection in the presence of intersymbol interference, IEEE Trans. Inform. Theory,* 18 (1972), 363–378.

[FC89] G.D. Forney, Jr., A.R. Calderbank, *Coset codes for partial response channels; or, cosets codes with spectral nulls, IEEE Trans. Inform. Theory,* 35 (1989), 925–943.

[ForsB88] K. Forsberg, I.F. Blake, *The enumeration of (d, k) sequences, Proc. 26th Allerton Conference on Communications, Control, and Computing,* Urbana-Champaign, Illinois (1988), 471–472.

[Fra68] P.A. Franaszek, *Sequence-state coding for digital transmission, Bell Sys. Tech. J.,* 47 (1968), 143–155.

[Fra69] P.A. Franaszek, *On synchronous variable length coding for discrete noiseless channels, Inform. Control,* 15 (1969), 155–164.

[Fra70] P.A. Franaszek, *Sequence-state methods for run-length-limited coding, IBM J. Res. Develop.,* 14 (1970), 376–383.

[Fra72] P.A. Franaszek, *Run-length-limited variable length coding with error propagation limitation,* US patent 3,689,899 (1972).

[Fra79] P.A. Franaszek, *On future-dependent block coding for input-restricted channels, IBM J. Res. Develop.,* 23 (1979), 75–81.

[Fra80a] P.A. Franaszek, *Synchronous bounded delay coding for input restricted channels, IBM J. Res. Develop.,* 24 (1980), 43–48.

[Fra80b] P.A. Franaszek, *A general method for channel coding, IBM J. Res. Develop.,* 24 (1980), 638–641.

[Fra82] P.A. Franaszek, *Construction of bounded delay codes for discrete noiseless channels, IBM J. Res. Develop.,* 26 (1982), 506–514.

[Fra89] P.A. Franaszek, *Coding for constrained channels: a comparison of two approaches, IBM J. Res. Dev.,* 33 (1989), 602–607.

[FT93] P.A. Franaszek, J.A. Thomas, *On the optimization of constrained channel codes,* IBM Research Report 19303 (1993). See also *Proc. 1993 IEEE Int'l Symp. Inform. Theory,* San Antonio, Texas (1993), p. 3.

[Fred89] L.J. Fredrickson, J.K. Wolf, *Error-detecting multiple block (d, k) codes, IEEE Trans. Magn.,* 25 (1989), 4096–4098.

[Fred94] L. Fredrickson, R. Karabed, P. Siegel, H. Thapar, R. Wood, *Improved trellis-coding for partial-response channels, Proc. IEEE Magn. Rec. Conf.,* San Diego, California (1994), *IEEE Trans. Magn.*, 31 (1995), 1141–1148.

[FW64] C. Freiman, A. Wyner, *Optimum block codes for noiseless input restricted channels, Inform. Control,* 7 (1964), 398–415.

[Fri84] J. Friedman, *A note on state splitting, Proc. AMS,* 92 (1984), 206–208.

[Fri90] J. Friedman, *On the road coloring problem, Proc. Amer. Math. Soc.,* 110 (1990), 1133–1135.

[Funk82] P. Funk, *Run-length-limited codes with multiple spacing, IEEE Trans. Magnetics,* 18 (1982), 772–775.

[Gant60] F.R. Gantmacher, *Matrix Theory, Volume II,* Chelsea Publishing Company, New York, 1960.

[GHW92] R.D. Gitlin, J.F. Hayes and S.B. Weinstein, *Data Communications Principles,* Plenum Press, New York, 1992.

[GoW68] S.W. Golomb, L.R. Welch, *Algebraic coding and the Lee metric,* in: *Error Correcting Codes,* H.B. Mann (Editor), John Wiley, 1968, pp. 175–194.

[GoW70] S.W. Golomb, L.R. Welch, *Perfect codes in the Lee metric and the packing of polyominoes, SIAM J. Appl. Math.,* 18 (1970), 302–317.

[GuF93] J. Gu, T. Fuja, *A generalized Gilbert-Varshamov bound derived via analysis of a code-search algorithm, IEEE Trans. Inform. Theory,* 39 (1993), 1089–1093.

[GuF94] J. Gu, T. Fuja, *A new approach to constructing optimal block codes for runlength-limited channels, IEEE Trans. Inform. Theory,* 40 (1994), 774–785.

[HRuHC] R. Haeb, D. Rugar, T. Howell and P.Coleman, *Coding and signal processing for a magnetooptic resonant bias coil overwrite experiment,* IBM Research Report, RJ 6962 (66499), 1989.

[HGO93] S. Hayes, C. Grebogi and E. Ott, *Communicationg with chaos, Physics Review Letters,* 70 (1993), 3031–3034.

[HHH00] W. Hirt, M. Hassner, N. Heise, *IrDA-VFIr (16 Mb/s): Modulation Code and System Design, IEEE Personal Communications,* to appear.

[HBH94] J. F. Heanue, M. C. Bashaw, and L. Hesselink, *Volume holographic storage and retrieval of digital data, Science,* 265 (1994), 749–752.

[Heeg91] C.D. Heegard, B.H. Marcus, P.H. Siegel, *Variable-length state splitting with applications to average runlength-constrained (ARC) codes, IEEE Trans. Inform. Theory,* 37 (1991), 759–777.

[Heem82] J.P.J. Heemskerk, K.A.S. Immink, *Compact disc: system aspects and modulation, Philips Techn. Review,* 40 (1982), 157–164.

[Hild91] H.M. Hilden, D.G. Howe, E.J. Weldon, Jr., *Shift error correcting modulation codes, IEEE Trans. Magn.,* 27 (1991), 4600–4605.

[Hod95] R. Hod, *Coding Methods for Input-Constrained Channels,* M.Sc. Thesis (in Hebrew), Technion, Haifa, Israel, 1995.

[Hole91] K.J. Hole, *Punctured convolutional codes for the* $1-D$ *partial-response channel, IEEE Trans. Inform. Theory,* 37 (1991), 808–817.

[HoY94] K.J. Hole, Ø. Ytrehus, *Improved coding techniques for partial-response channels, IEEE Trans. Inform. Theory,* 40 (1994), 482–493.

[Holl94] H.D.L. Hollmann, *A block-decodable* $(1, 8)$ *runlength-limited rate* $8/12$ *code, IEEE Trans. Inform. Theory,* 40 (1994), 1292–1296.

[Holl95] H.D.L. Hollmann, *On the construction of bounded-delay encodable codes for constrained systems, IEEE Trans. Inform. Theory,* 41 (1995), 1354–1378.

[Holl96] H.D.L. Hollmann, *Bounded-delay-encodable, block-decodable codes for constrained systems, IEEE Trans. Inform. Theory,* 42 (1996), 1957–1970.

[Holl97] H.D.L. Hollmann, *On an approximate eigenvector associated with a modulation code, IEEE Trans. Inform. Theory,* 43 1997), 1672–1678.

[Hopc79] J.E. Hopcroft, J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation,* Addison-Wesley, Reading, Massachusetts, 1979.

[HorM76] J.E. T. Horiguchi, K. Morita, *An optimization of modulation codes in magnetic recording, IEEE Trans. Magnetics,* 12 (1976), 740–746.

[How84] T.D. Howell, *Analysis of correctable errors in the IBM 3380 disk file, IBM J. Res. Develop.,* 28 (1984), 206–211.

[How89] T.D. Howell, *Statistical properties of selected recording codes, IBM J. Res. Dev.,* 32 (1989), 60–73.

[Huff54] D.A. Huffman, *The synthesis of sequential switching circuits, J. Franklin Inst.,* 257 (1954), 161–190 and 275–303.

[Huff59] D.A. Huffman, *Canonical forms for information lossless finite-state machine, IRE Trans. Circuit Theory,* 6, (1959, Special Supplement), 41–59.

[IKN80] I. Iizuka, M. Kasahara, T. Namekawa, *Block codes capable of correcting both additive and timing errors, IEEE Trans. Inform. Theory,* 26 (1980), 393–400.

[Imm90] K.A.S. Immink, *Runlength-limited sequences, Proc. IEEE,* 78 (1990), 1745–1759.

[Imm91] K.A.S. Immink, *Coding Techniques for Digital Recorders,* Prentice Hall, New York, 1991.

[Imm92] K.A.S. Immink, *Block-decodable runlength-limited codes via look-ahead technique, Philips J. Res.,* 46 (1992), 293–310.

[Imm95a] K.A.S. Immink, *Constructions of almost block-decodable runlength-limited codes, IEEE Trans. Inform. Theory,* 41 (1995), 284–287.

[Imm95b] K.A.S. Immink, *EFMPlus: The coding format of the multimedia compact disc, IEEE Trans. Consum. Electron.,* 41 (1995), 491–497.

[Imm99] K.A.S. Immink, *Codes for Mass Data Storage Systems,* Shannon Foundation Publishers, The Netherlands, 1999.

[Imm97] K.A.S. Immink, *A practical method for approaching the channel capacity of constrained channels, IEEE Trans. Inform. Theory,* 43 (1997), 1389–1399.

[ImmB87] K.A.S. Immink, G.F.M. Beenker, *Binary transmission codes with higher order spectral zeros at zero frequency, IEEE Trans. Inform. Theory,* 33 (1987), 452–454.

[IO85] K.A.S. Immink, H. Ogawa, *Method for encoding binary data,* US patent 4,501,000 (1985).

[Jon95] N. Jonoska, *Sofic shifts with synchronizing presentations, Theoretical Computer Science,* 158 (1996), 81–115.

[Jus82] J. Justesen, *Information rates and power spectra of digital codes, IEEE Trans. Inform. Theory,* 28 (1982), 457–472.

[JusH84] J. Justesen, T. Høholdt, *Maxentropic Markov chains, IEEE Trans. Inform. Theory,* 30 (1984), 665–667.

[KDV92] G.A. Kabatiansky, V.A. Davydov, A.J.H. Vinck, *On error correcting codes for three types of channels, Proc. Int'l Workshop Algebraic Combin. and Coding Theory,* Bulgaria (1992), 101–103.

[Kam89] H. Kamabe, *Minimum scope for sliding block decoder mappings, IEEE Trans. Inform. Theory,* 35 (1989), 1335–1340.

[Kam94] H. Kamabe, *Irreducible components of canonical diagrams for spectral nulls, IEEE Trans. Inform. Theory,* 40 (1994), 1375–1391.

[KarM88] R. Karabed, B.H. Marcus, *Sliding-block coding for input-restricted channels, IEEE Trans. Inform. Theory,* 34 (1988), 2–26.

[KS91a] R. Karabed, P.H. Siegel, *Matched spectral null codes for partial response channels, IEEE Trans. Inform. Theory,* 37 (1991), 818–855.

[KS91b] R. Karabed, P.H. Siegel, *A 100% efficient sliding-block code for the charge-constrained, runlength-limited channel with parameters* $(d, k; c) = (1, 3; 3)$, *Proc. 1991 IEEE Int'l Symp. Inform. Theory,* Budapest, Hungary (1991), p. 229.

[KSS00] R. Karabed, P.H. Siegel, E. Soljanin, *Constrained coding for binary channels with high intersymbol interference, IEEE Trans. Inform. Theory,* 45 (1999), 1777–1797.

[KZ98] A. Kato, K. Zeger, *On the capacity of two dimensional run length constrained channels, IEEE Trans. Inform. Theory,* 45 (1999), 1527–1540.

[Ker91] K.J. Kerpez, *Runlength codes from source codes, IEEE Trans. Inform. Theory,* 37 (1991), 682–687.

[Khay89] Z.-A. Khayrallah, *Finite-State Codes and Input-Constrained Channels,* Ph.D. Thesis, University of Michigan, 1989.

[KN90] Z.-A. Khayrallah, D. Neuhoff, *Subshift models and finite-state codes for input-constrained noiseless channels: a tutorial, Udel-EE Technical Report Number 90-9-1,* 1990.

[KN96] Z.-A. Khayrallah, D. Neuhoff, *Coding for channels with cost constraints, IEEE Trans. Inform. Theory,* 42 (1996), 854–867.

[KN99] B. King and M. Neifeld, *Unequal a priori probabilities for holographic storage, Proc. SPIE,* Vol. 3802-08 (1999), 40–43.

[Kit81] B. Kitchens, *Continuity Properties of Factor Maps in Ergodic Theory,* Ph.D. Thesis, University of North Carolina, Chapel Hill, 1981.

[Kl95] T. Kløve, *Codes correcting a single insertion/deletion of a zero or a single peak-shift, IEEE Trans. Inform. Theory,* 41 (1995), 279–283.

[Koba71] H. Kobayashi, *Application of probabilistic decoding to digital magnetic recording systems, IBM J. Res. Develop.,* 15 (1971), 64-74.

[Koba72] H. Kobayashi, *Correlative level coding and maximum-likelihood decoding, IEEE Trans. Inform. Theory,* 18 (1972), 363–378.

[KobT70] H. Kobayashi, D.T. Tang, *Application of partial-response channel coding to magnetic recording systems, IBM J. Res. Develop.,* 14 (1970), 368–374.

[Koh78] Z. Kohavi, *Switching and Finite Automata Theory,* Second Edition, Tata McGraw-Hill, New Delhi, 1978.

[KolK91] V.D. Kolesnik, V.Yu. Krachkovsky, *Generating functions and lower bounds on rates for limited error-correcting codes, IEEE Trans. Inform. Theory,* 37 (1991), 778–788.

[KolK94] V.D. Kolesnik, V.Yu. Krachkovsky, *Lower bounds on the achievable rates for limited bitshift-correcting codes, IEEE Trans. Inform. Theory,* 40 (1994), 1443–1458.

[Kretz67] E.R. Kretzmer, *Generalization of a technique for binary data transmission, IEEE Trans. Commun. Technol.,* 14 (1967), 67.

[Krusk83] J.B. Kruskal, *An overview of sequence comparison: Time warps, string edits, and macromolecules, Siam Review,* 25 (1983), 201–237.

[KuV93a] A.V. Kuznetsov, A.J.H. Vinck, *A coding scheme for single peak-shift correction in (d, k)-constrained channels, IEEE Trans. Inform. Theory,* 39 (1993), 1444–1450.

[KuV93b] A.V. Kuznetsov, A.J.H. Vinck, *The application of q-ary codes for the correction of single peak-shifts, deletions and insertions of zeros, Proc. 1993 IEEE Int'l Symp. Inform. Theory,* San Antonio, Texas (1993), p. 128.

[Lee88] P. Lee, *Combined error-correcting/modulation codes,* Ph.D. dissertation, University of California, San Diego, USA, 1988.

[LW87] P. Lee, J.K. Wolf, *Combined error-correction/modulation codes, IEEE Trans. Magnetics,* 23 (1987), 3681–3683.

[LW89] P. Lee, J.K. Wolf, *A general error-correcting code construction for run-length limited binary channels, IEEE Trans. Inform. Theory,* 35 (1989), 1330–1335.

[LemCo82] A. Lempel, M. Cohn, *Look-ahead coding for input-restricted channels, IEEE Trans. Inform. Theory,* 28 (1982), 933–937.

[Lev65] V.I. Levenshtein, *Binary codes capable of correcting deletions, insertions, and reversals,* (Russian), *Doklady Akademii Nauk SSSR,* 163 (1965), 845–848. (English), *Soviet Physics Doklady,* 10 (1966), 707–710.

[Lev67] V.I. Levenshtein, *Asymptotically optimum binary code with correction for losses of one or two ajacent bits, Problems of Cybernetics,* 19 (1967), 298–304.

[Lev71] V.I. Levenshtein, *One method of constructing quasilinear codes providing synchronization in the presence of errors,* (Russian), *Problemy Peredachi Informatsii,* 7 (1971), 30–40. (English), *Problems of Information Transmission,* 7 (1971), 215–222.

[Lev91] V.I. Levenshtein, *On perfect codes in deletion and insertion metric,* (Russian), *Discretnaya. Mathematika,* 3 (1991), 3–20. (English), *Discrete Mathematics and Applications,* 2 (1992), 241–258.

[LV93] V.I. Levenshtein, A.J.H. Vinck, *Perfect (d, k)-codes capable of correcting single peak-shifts, IEEE Trans. Inform. Theory,* 39 (1993), 656–662.

[LinCo83] S. Lin, D.J. Costello, Jr., *Error Control Coding, Fundamentals and Applications,* Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[LM95] D. Lind, B. Marcus, *An Introduction to Symbolic Dynamics and Coding,* Cambridge University Press, 1995.

[MacS77] F.J. MacWilliams, N.J.A. Sloane, *The Theory of Error-Correcting Codes,* North-Holland, Amsterdam, 1977.

[Man91] M. Mansuripur, *Enumerative modulation coding with arbitrary constraints and post-modulation error correction coding and data storage systems, Proc. SPIE,* Vol. 1499 (1991), 72–86.

[MM77] J.C. Mallinson, J.W. Miller, *Optimal codes for digital magnetic recording, Radio and Elec. Eng.,* 47 (1977), 172–176.

[Mar85] B.H. Marcus, *Sofic systems and encoding data, IEEE Trans. Inform. Theory,* 31 (1985), 366–377.

[MR91] B.H. Marcus, R.M. Roth, *Bounds on the number of states in encoder graphs for input-constrained channels, IEEE Trans. Inform. Theory,* IT-37 (1991), 742–758.

[MR92] B.H. Marcus, R.M. Roth, *Improved Gilbert-Varshamov bound for constrained systems, IEEE Trans. Inform. Theory,* 38 (1992), 1213–1221.

[MS87] B.H. Marcus, P.H. Siegel, *On codes with spectral nulls at rational submultiples of the symbol frequency, IEEE Trans. Inform. Theory,* 33 (1987), 557–568.

[MS88] B. Marcus, P.H. Siegel, *Constrained codes for partial response channels, Proc. Beijing Int'l Workshop on Information Theory* (1988), DI-1.1–1.4.

[MSW92] B.H. Marcus, P.H. Siegel, J.K. Wolf, *Finite-state modulation codes for data storage, IEEE J. Sel. Areas Comm.,* 10 (1992), 5–37.

[MLT83] G.N.N. Martin, G.G. Langdon, S.J.P. Todd, *Arithmetic codes for constrained channels, IBM J. Res. Develop.,* 27 (1983), 94–106.

[Mcl77] R.J. McEliece, *The Theory of Information and Coding,* Addison-Wesley, Reading, Massachusetts, 1977.

[Mill63] A. Miller, *Transmission system,* US patent 3,108,261 (1963).

[Mill77] J.W. Miller, *DC free encoding for data transmission system,* US patent 4,027,335 (1977).

[Minc88] H. Minc, *Nonnegative Matrices,* Wiley, New York, 1988.

[MoMa01] D.S. Modha, B.H. Marcus, *Art of constructing low complexity encoders/decoders for constrained block codes, IEEE J. Sel. Areas in Comm.,* (2001), to appear.

[MPi89] C.M. Monti, G.L. Pierobon, *Codes with a multiple spectral null at zero frequency, IEEE Trans. Inform. Theory,* 35 (1989), 463–472.

[Moore56] E.F. Moore, *Gedanken-experiments on sequential machines, Automata Studies,* Princeton University Press, Princeton, New Jersey, 1956, 129–153.

[NB81] K. Norris, D.S. Bloomberg, *Channel capacity of charge-constrained run-length limited systems, IEEE Trans. Magnetics,* 17 (1981), 3452–3455.

[Obr81] G.L. O'Brien, *The road coloring problem, Israel J. Math.,* 39 (1981), 145–154

[Or93] A. Orlitsky, *Interactive communication of balanced distributions and of correlated files, SIAM J. Discr. Math.,* 6 (1993), 548–564.

[Ott93] E. Ott, *Chaos in Dynamical Systems,* Cambridge Univ. Press, 1993.

[OGY90] E. Ott, C. Grebogi, J. York, *Controlling Chaos, Physics Review Letters,* 64 (1990), 1196–1199.

[Par64] W. Parry, *Intrinsic Markov chains, Transactions AMS,* 112 (1964) 55-66.

[PT82] W. Parry, S. Tuncel *Classification Problems in Ergodic Theory,* Cambridge University Press, 1982.

[PK92] A. Patapoutian, P. V. Kumar, *The (d, k) subcode of a linear block code, IEEE Trans. Inform. Theory,* 38 (1992), 1375–1382.

[Patel75] A.M. Patel, *Zero-modulation encoding in magnetic recording, IBM J. Res. Develop.,* 19 (1975), 366–378.

[PRS63] M. Perles, M.O. Rabin, E. Shamir, *The theory of definite automata, IEEE. Trans. Electron. Computers,* 12 (1963), 233–243.

[PS92] D. Perrin, M.-P. Schutzenberger, *Synchronizing prefix codes and automata and the road coloring problem,* in *Symbolic Dynamics and Its Applications,* Contemporary Mathematics 135 (1992), P. Walters (Editor), 295–318.

[PW72] W.W. Peterson, E.J. Weldon, Jr., *Error-Correcting Codes,* Second Edition, MIT Press, Cambridge, Massachusetts, 1972.

[Pie84] G.L. Pierobon, *Codes for zero spectral density at zero frequency, IEEE Trans. Inform. Theory,* 30 (1984), 435–439.

[Pl89] V. Pless, *Introduction to the Theory of Error Correcting Codes,* Second Edition, John Wiley, New York, 1989.

[PH98] V.S Pless, W.C. Huffman *Handbook of Coding Theory,* Elsevier, Amsterdam, 1998.

[Pohl92] K.C. Pohlmann, *The Compact Disc Handbook,* Second Edition, A–R Editions, Madison, Wisconsin, 1992.

[Rae94] J.W. Rae, G.S. Christiansen, P. Siegel, R. Karabed, H. Thapar, S. Shih, *Design and performance of a VLSI 120 Mb/s trellis-coded partial response channel, Proc. IEEE Magn. Rec. Conf.,* San Diego, California (1994), *IEEE Trans. Magn.*, 31 (1995), 1208–1214.

[Roth00] R.M. Roth, *On runlength-limited coding with DC control, IEEE Trans. Commun.,* (2000), 351–358.

[RS92] R.M. Roth, P.H. Siegel, *A family of BCH codes for the Lee metric, Proc. Thirtieth Annual Allerton Conf. on Communication, Control, and Computing,* Urbana-Champaign, Illinois, September 1992, 1–10.

[RS94] R.M. Roth, P.H. Siegel, *Lee-metric BCH codes and their application to constrained and partial-response channels, IEEE Trans. Inform. Theory,* 40 (1994), 1083–1096.

[RSV94] R.M. Roth, P.H. Siegel, A. Vardy, *High-order spectral-null codes: constructions and bounds, IEEE Trans. Inform. Theory,* 40 (1994), 1826–1840.

[RSW00] R.M. Roth, P.H. Siegel, J.K. Wolf, *Efficient coding of two-dimensional runlength-limited constraints, Proc. SPIE,* Vol. 3802 (1999), 8–17.

[Roth93] R.M. Roth, *Spectral-null codes and null spaces of Hadamard submatrices, Designs, Codes, and Cryptography,* 9 (1996), 177–191. See also *Proc. First French–Israeli Workshop on Algebraic Coding,* Paris (1993), G. Cohen, S. Litsyn, A. Lobstein, G. Zémor (Editors), Springer (LNCS 781, 1994), 141–153.

[Ru96] G. Ruckenstein (Sadeh), *Encoding for Input-Constrained Channels,* M.Sc. Thesis, Technion, Haifa, Israel, 1996.

[RuR01] G. Ruckenstein (Sadeh) and R. Roth, *Lower bounds on the anticipation of encoders for input-constrained channels, IEEE Trans. Inform. Theory,* 47 (2001), 1796–1812.

[RuS89] D. Rugar, P.H. Siegel, *Recording results and coding considerations for the resonant bias coil overwrite technique, Optical Data Storage Topical Meeting,* G.R. Knight, C.N. Kurtz (Editors), *Proc. SPIE,* Vol. 1078 (1989), 265–270.

[Sai93a] Y. Saitoh, *Theory and design of error-control codes for byte-organized/ input-restricted storage devices where unidirectional/peak-shift errors are predominant,* Ph.D. dissertation, Division of Electrical and Computer Engineering, Yokohama National University, Yokohama, Japan, February 1993.

[Sai93b] Y. Saitoh, T. Ohno, H. Imai, *Construction techniques for error-control runlength-limited block codes, IEICE Trans. Fundamentals,* E76-A (1993), 453–458.

[Sen80] E. Seneta, *Non-negative Matrices and Markov Chains,* Second Edition, Springer, New York, 1980.

[SZ91] S. Shamai, E. Zehavi, *Bounds on the capacity of the bit-shift magnetic recording channel, IEEE Trans. Inform. Theory,* 37 (1991), 863–871.

[Sha48] C.E. Shannon, *The mathematical theory of communication, Bell Sys. Tech. J.,* 27 (1948), 379–423.

[ST93] D.B. Shmoys, É. Tardos, *Computational complexity,* in *The Handbook of Combinatorics,* L. Lovász, R.L. Graham, M. Grötschel (Editors), North Holland, Amsterdam (to appear).

[Shung91] C. Shung, P. Siegel, H. Thapar, R. Karabed, *A 30 MHz trellis codec chip for partial-response channels, IEEE J. Solid-State Circ.,* 26 (1991), 1981–1987. San Francisco, February 1991, pp. 132–133.

[Sie85a] P.H. Siegel, *Recording codes for digital magnetic storage, IEEE Trans. Magnetics,* 21 (1985), 1344–1349.

[Sie85b] P.H. Siegel, *On the complexity of limiting error propagation in sliding block codes, Proc. 2nd IBM Symp. on Coding and Error Control,* San Jose, California, January 1985.

[SW91] P.H. Siegel, J.K. Wolf, *Modulation and coding for information storage, IEEE Commun. Magazine,* 29 (1991), 68–86.

[Sklar88] B. Sklar, *Digital Communications: Fundamentals and Applications,* Prentice-Hall, 1988.

[SC95] N. Swenson, J. M. Cioffi, *Sliding block line codes to increase dispersion-limited distance of optical fiber channels, IEEE J. Select. Areas Commun.,* 13 (1995), 485–498.

[TK76] E. Tanaka, T. Kasai, *Synchronization and substitution error-correcting codes for the Levenshtein metric, IEEE Trans. Inform. Theory,* 22 (1976), 156–162.

[TB70] D.T. Tang, L.R. Bahl, *Block codes for a class of constrained noiseless channels, Inform. Control,* 17 (1970), 436–461.

[Ten76] G.M. Tenengolts, *Class of codes correcting bit loss and errors in the preceding bit,* (Russian), *Avtomatika i Telemekhanika,* 37 (1976), 174–179. (English), *Automation and Remote Control,* 37 (1976), 797–802.

[Ten84] G.M. Tenengolts, *Nonbinary codes, correcting single deletion or insertion, IEEE Trans. Inform. Theory,* 30 (1984), 766–769.

[TiBl89] H.C.A. van Tilborg, M. Blaum *On error-correcting balanced codes, IEEE Trans. Inform. Theory,* 35 (1989), 1091–1093.

[Tja94] T.J. Tjalkens, *On the principal state method for runlength limited sequences, IEEE Trans. Inform. Theory,* 40 (1994), 934–941.

[Thap92] H.K. Thapar, J. Rae, C.B. Shung, R. Karabed, P.H. Siegel, *Performance evaluation of a rate 8/10 matched spectral null code for class-4 partial response, IEEE Trans. Magn.,* 28 (1992), 2884–2889.

[Thap93] H. Thapar, C. Shung, J. Rae, R. Karabed, P.H. Siegel, *Real-time recording results for a trellis-coded partial response (TCPR) system, IEEE Trans. Magn.,* 29 (1993), 4009–4011.

[TLM27] S.J.P. Todd, G.N.N. Martin, G.G. Langdon, *A general fixed rate arithmetic coding method for constrained channels, IBM J. Res. Develop.,* 27 (1983), 107–115.

[U66] J.D. Ullman, *Near-optimal, single-synchronization error-correcting code, IEEE Trans. Inform. Theory,* 12 (1966), 418–424.

[U67] J.D. Ullman, *On the capabilities of codes to correct synchronization errors, IEEE Trans. Inform. Theory,* 13 (1967), 95–105.

[Var62] R.S. Varga, *Matrix Iterative Analysis,* Prentice-Hall, Englewood Cliffs, New Jersey, 1962.

[WN95] H. Waldman, E. Nisenbaum, *Upper bounds and Hamming spheres under the DC constraint, IEEE Trans. Inform. Theory,* 41 (1995), 1138–1145.

[WT99] S.X. Wang, A. M. Taratorin, *Magnetic Information Storage Technology,* Academic Press, 1999.

[WW91] A.D. Weathers, J.K. Wolf, *A new 2/3 sliding block code for the (1,7) runlength constraint with the minimal number of encoder states, IEEE Trans. Inform. Theory,* 37 (1991), 908–913.

[Weig88] T. Weigandt, *Magneto-optic recording using a (2,18,2) runlength limited code,* S.M. Thesis, MIT, Cambridge, MA, 1991.

[Wic95] S.B. Wicker, *Error Control Coding in Digital Communication and Storage,* Prentice-Hall, 1995.

[WF83] A. Widmer and P. Franaszek, *A DC-balanced, partitioned-block 8b/10b transmission code, IBM J. Res. Develop.,* 27 (1983), 440–451.

[Will73] R.F. Williams, *Classification of subshifts of finite type, Annals Math.,* 98 (1973), 120–153; errata: *Annals Math.,* 99 (1974), 380–381.

[Will88] S. Williams, *Covers of non-almost-finite-type systems, Proc. AMS,* 104 (1988), 245–252.

[WY93] K. Winick, S.-H. Yang, *Bounds on the size of error correcting runlength-limited codes,* preprint, 1993.

[WU86] J.K. Wolf, G. Ungerboeck, *Trellis coding for partial-response channels, IEEE Trans. Commun.,* 34 (1986), 765–773.

[Wood90] R. Wood, *Denser magnetic memory, IEEE Spectrum*, 27, No. 5 (May 1990), 32–39.

[WoodP86] R. Wood, D. Peterson, *Viterbi detection of class IV partial response on a magnetic recoding channel, IEEE Trans. Commun.,* 34 (1986), 454–461.

[YY76] S. Yoshida, S. Yajima, *On the relation between an encoding automaton and the power spectrum of its output sequence, Trans. IECE Japan,* 59 (1976), 1–7.

[Yt91a] Ø. Ytrehus, *Upper bounds on error-correcting runlength-limited block codes, IEEE Trans. Inform. Theory,* 37 (1991), 941–945.

[Yt91b] Ø. Ytrehus, *Runlength-limited codes for mixed-error channels, IEEE Trans. Inform. Theory,* 37 (1991), 1577–1585.

[Ze87] E. Zehavi, *Coding for Magnetic Recording,* Ph.D. Thesis, University of California, San Diego, 1987.

[ZW88] E. Zehavi, J.K. Wolf, *On runlength codes, IEEE Trans. Inform. Theory,* 34 (1988), 45–54.