Coding for Two Dimensional Constrained Fields

Laursen, Torben Vaarbye

Publication date:
2006

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Laursen, T. V. (2006). *Coding for Two Dimensional Constrained Fields*. Technical University of Denmark.

Coding for Two Dimensional Constrained Fields

Torben Vaarby Laursen

May 26, 2006

COM•DTU

Research Center COM
Technical University of Denmark
Building 343
2800 Kgs. Lyngby
DENMARK

Kodning for todimensionelle skiftrum

Som en del af mit Ph.D.-projekt ved COM institutet på DTU under vejledning af lektor Søren Forchhammer har jeg udarbejdet denne rapport. Mit projekt omhandler kodning for todimensionelle skiftrum.

Jeg vil i dette danske resumé beskrive baggrunden for projektet, samt opremse de væsenligste resultater. For en mere udførlig gennemgang henvises til selve afhandlingen, der er forfattet på engelsk.

Overførsel af data, det være sig i netværk eller i forbindelse med datalagring, modelleres normalt som transmission over en støjfyldt kanal. Visse datasekvenser vil være mere udsat for støj end andre på grund af særlige fysiske karakteristika ved kanalen. Det giver derfor mening at forbyde disse særlig fejlbehæftede signaler og fjerne dem ved hjælp af en kode inden transmission.

Teorien for symbolske dynamiske systemer tilbyder en generel ramme til at afdække konstruktion af koder, der er fri for visse forbudte ord. En samling af sekvenser, hvori der ikke optræder forbudte ord kaldes et skiftrum. Disse skiftrum er yderst velforståede, og det er muligt at beregne den maksimalt opnåelige koderate, entropien, for en given liste af forbudte ord. Der eksisterer også værktøjer til effektivt at frembringe gode koder, der har koderater under (men tæt på) entropien.

Nye typer lagermedier, såsom holografisk lagring er todimensionelle og rejser behovet for at kunne undgå visse mønstre. Teorien for todimensionelle skiftrum er langt vanskeligere. Det er ikke muligt eksplicit at beregne entropien. Teknikkerne for kodekonstruktion kan heller ikke umiddelbart generaliseres til to dimensioner.

I dette arbejde forsøger vi at afdække noget af teorien for todimensionelle skiftrum. Vi indfører en række eksempler på skiftrum, der kan modellere visse aspekter af datalagring. Eksempelvis kan skiftrummet NIB modellere visse af de særlig fejlbehæftede mønstre, der optræder under holografisk lagring. Vi giver en række øvre og nedre grænser for entropien af disse skiftrum baseret på endimensionelle teknikker.

Vi diskuterer brugen af stokastiske modeller for skiftrum. Vi anvender en første ordens Pickard model til at modellere højere ordens skiftrum ved hjælp af en alfabetudvidelse. Dette leder til modeller med et meget stort antal parametre. Vi præsenterer en iterativ metode, der med udgangspunkt i et sæt betingede sandsynligheder kan hjælpe med at vælge parametre for en stationær model.

Tanken med modellerne er at modellere opførslen af forskellige kodningsteknikker. Vi giver et eksempel, der tager afsæt i kodningsmetoden bit-stuffing for det konkrete skiftrum NIB. Vi angiver parametrene for en stationær model, hvis sandsynligheder er induceret af bit-stuffing indkoderen og beregner dens entropi.

Endelig præsenterer vi en variant af bitstuffingkodningsmetoden, hvor det er muligt at udregne koderaten. Metoden kan anvendes på alle checkerboard skiftrum og gør det muligt at finde gode nedre grænser for entropien af disse skiftrum. Konkret gives der grænser for en række Run-Length-Limited skiftrum, der er blandt de hidtige bedste grænser for disse skiftrum. Metoden demonstreres også for det såkaldte diamond skiftrum.

Abstract

This report deals with constrained coding in two dimensions. We describe the theory of constrained fields as a framework for addressing some of the challenges of code construction for advanced data storage devices that treat the recording media as a surface, rather than a series of tracks. The important concept of entropy is introduced. In general, the entropy of a constrained field is not readily computable, but we give a series of upper and lower bounds based on one dimensional techniques.

We discuss the use of a Pickard probability model for constrained fields. The novelty lies in using a first order model to model higher order constraints by the use of an alphabet extension. We present an iterative method that based on a set of conditional probabilities can help in choosing the large numbers of parameters of the model in order to obtain a stationary model. Explicit results are given for the No Isolated Bits constraint.

Finally we present a variation of the encoding scheme of bit-stuffing that is applicable to the class of checkerboard constrained fields. It is possible to calculate the entropy of the coding scheme thus obtaining lower bounds on the entropy of the fields considered. These lower bounds are very tight for the Run-Length limited fields. Explicit bounds are given for the diamond constrained field as well.

Preface

This report is part of the requirement for the fulfillment of Ph.D. program at the Technical University of Denmark. The work was carried out at the Department of Communications, Optics and Materials under the supervision of lektor Søren Forchhammer.

Some of the software developed as part of this work was made while I was visiting the Center for Magnetic Recording Research, at the University of California, San Diego in the Fall of 2003. I am grateful to the director professor Paul Siegel for all his time and his willingness to aid while I visited CMRR.

Also, I would like to thank my supervisor Søren Forchhammer for his patience and good spirits.

Most of the results in this work are joint work with my supervisor Søren Forchhammer and have been submitted in another form. In particular the contents of Section 2.7 appeared as [12]. Furthermore much of Chapter 3 has been submitted as [14] whereas Chapter 4 is a revised version of the conference paper [17] and has been submitted in a slightly different version as the article [13].

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In many communication systems be they optical networks or data storage devices, some sequences of signals are more prone to error than others. For instance in an optical network a light pulse often signifies a one and no pulse is interpreted as zero. But this means that long sequences of zeroes will be difficult to distinguish from one another.

Hence in order to enhance reliable communication it seems prudent to impose restrictions on the sequences that are allowed to be transmitted (or recorded). One could say that we constrain the number of allowed sequences. This raises the need for a theory of constrained coding. How do we encode the sequences such that they obey certain constraints and what rates are achievable doing this.

We will use the celebrated information theoretic model of communication as proposed by Shannon [44]. As Berlekamp has put it:

> Communication links transmit information from here to there.
> Computer memories transmit information from now to then.

Consider Figure 1.1. The data source generates data. A general error correcting code is applied to the data. Then a modulation code is applied that shapes the data to the characteristics of the channel, by imposing certain constraints on the sequences. Our main point of interest is this constrained coding step.

There is a large body of work dealing with constrained coding. A recent survey article is [25]. Much of the theory has been developed for and applied to digital storage devices. In classical storage devices

$$\begin{tikzcd}
\text{Data Source} \arrow[r] & \text{ECC} \arrow[r] & \text{Modulation} \arrow[d] \\
& & \text{Noisy channel} \arrow[d] \\
\text{Destination} & \text{ECC decoding} \arrow[l] & \text{Demodulation} \arrow[l]
\end{tikzcd}$$

Figure 1.1: A model of digital storage

such as magnetic tape, hard disks and optical discs data is written along tracks. From a high level point of view, ones are typically detected as changes (shift between magnetic orientations for tape and magnetic discs or between pits and lands in an optical disc) and zeros as no change. Hence the amount of zeros read is determined by reading speed and a clock. Usually the clock is reset each time a one is detected. But if a large sequence of zeros occur, we might miss detection of one of them, due to inaccuracies in the clock and/or variation in the symbol speed. This problem is known as *clock drift*. Neighboring symbols tend to influence each other. This pheanomen is known as Inter-symbol Interference (ISI).

Most popular codes for storage devices have obeyed certain *run-length-limited* RLL(d, k) constraints where the number of 0s between successive 1s has to be at least d and at most k. The purpose of the k constraint is to limit clock drift whereas the d constraint serves to limit ISI.

There is a very well-developed theory for constrained coding that not only offers the maximum achievable coding rate for a given constraint, but also offers an algorithm (the State-Splitting Algorithm [1], [34]) for constructing codes with a given code rate below this maximum.

1.1 Advanced data storage devices

In the search for ever greater data densities researchers have considered devices that treat the storage medium as a *surface* rather than being essentially one dimensional.

Holographic storage, while an old idea [22], has very recently been brought past prototyping [46]. The basic functionality of a holographic storage system can be outlined as follows. A laser is directed towards an array of shutters, each of which can be either shut or open, allowing light to pass through in the latter case. The array is programmable (i.e. a pattern of ones and zeros). When the light beams pass through the open shutters an object beam is created. The object beam is focused through a lens and exposed to a reference beam. The two beams interfere with each other and cause a pattern on the recording medium. This pattern is the hologram.

If the hologram is illuminated by the reference beam, the object beam (i.e. the data) is recreated and can be focused and detected with a charge coupled device.

When a zero is surrounded by ones, the system tend to be prone to inter-pixel interference, i.e. the *pattern*

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

is especially error-prone.

Hence, it makes sense to eliminate these patterns from the data before recording.

As another example consider the millipede project [48], [9]. Here an array of small tips usually employed in atomic force microscopy is used to melt tiny depressions in a polymer medium. The tips can both read and write, and it is possible to erase previously written data.

The read proces is based on a thermo-mechanical sensing. The tip is heated up and if it is above a pit, it cools more quickly (detected by measuring the resistance of the tip, by having a small voltage pass through it).

In order to ensure reliable read-back the channel bits have to be spaced sufficiently far apart that neighboring holes do not interfere with each other.

However, by employing a two dimensional code where each one (hole) has to be surrounded by zeros, we may be able to pack the channel bits more closely thereby increasing the storage density of the medium without sacrificing reliability. That is, we might wish to forbid patterns such as:

$$\left\{ 11, 101, \begin{matrix} 01 \\ 10 \end{matrix}, \begin{matrix} 10 \\ 01 \end{matrix}, \begin{matrix} 1 \\ 1 \end{matrix}, \begin{matrix} 1 \\ 0 \\ 1 \end{matrix} \right\}.$$

As can be seen from these two examples, it appears interesting to try to investigate coding for *constrained fields*. This is the theme of the present text.

1.2 Outline of the text

In Chapter 2 we introduce constrained fields. We give several examples of concrete fields and give estimates for their entropy based on a one dimensional approximation by a finite state machine. We introduce the coding scheme of bit-stuffing. Finally in Section 2.7 we discuss a method for designing two dimensional constrained codes by cascading finite width arrays using predefined finite width periodic merging arrays. We give concrete examples for a density constraint and a symmetric run-length-limited constraint.

In Chapter 3 we discuss probability measures that agree with constraints. It general it appears to be difficult to find stationary models of constrained fields. We explore the use of the first order Pickard model to model higher order constraints by using an alphabet extension. We offer an iterative method that finds a stationary model for the No Isolated Bit constraint. In Section 3.6.4 we present the parameters for a model determined in this fashion that sports a high entropy.

Then in Chapter 4 we present a variation of bit-stuffing and demonstrates how to calculate the entropy of the induced measures. This offers some very tight lower bounds on the entropy of the run-length-limited(d, ∞) constraint for $d = 2, 3, 4$. The method is applicable to all checkerboard constraints and as a further example we use the method of the diamond(3) constraint.

Finally, we sum up our work in the conclusion where we collect the best bounds on the entropy of the constrained fields considered in this text.

Chapter 2

Constrained fields

In this chapter we define constrained fields by a list of forbidden patterns. We define the important concept of entropy and discuss how to estimate the entropy of a constrained field. Furthermore we present two examples of coding schemes for some particular constrained fields.

2.1 Basic definitions

Let $\mathcal{A}$ denote a finite alphabet, usually $\mathcal{A} = \{0, 1\}$. Consider some finite subset, C, of the integer lattice $\mathbb{Z}^2$. We are interested in different *labelings* of C, that is, different mappings $L : C \longrightarrow \mathcal{A}$ that obeys certain characteristics, for instance

$$\forall x, y \in C : L(x) = 1, \|x - y\| = 1 \Rightarrow L(y) = 0.$$

Or expressed in a different fashion: Neighboring ones are not allowed.

We will refer to the pair (C, L) as a *configuration* and usually we will omit reference to the labeling itself. Instead of specifying the labeling explicitly, we instead list the patterns we are not interested in. One can think of these as the especially error-prone patterns of the introduction.

Let $\mathcal{F}$ denote a finite list of *forbidden blocks* made of symbols from $\mathcal{A}$. We define the 2D constrained field $C(\mathcal{F})$ to be the set of all configurations in the plane where no blocks from $\mathcal{F}$ occur. Sometimes we will refer to a constrained field as a *constraint*. When dealing with a finite configuration C and a list of forbidden words $\mathcal{F}$, we will say that the configuration does not violate the constraint, if no blocks from $\mathcal{F}$ occur in C. Equivalently, we say that C is an *admissible* configuration.

Let N, respectively M denote the largest height, respectively width of a block in $\mathcal{F}$. The *extent* of a constrained field $C(\mathcal{F})$ is $N \times M$. If either N or M equals 1, the constrained field is in fact one-dimensional. Hence the smallest extent we will consider is 2×2, which we call a *first order constraint.*

Example 2.1. Let $\mathcal{A} = \{0, 1\}$. Let

$$\mathcal{F} = \left\{ 11, \begin{matrix} 1 \\ 1 \end{matrix} \right\}.$$

The first order field $C(\mathcal{F})$ is called the *Hard Square constraint.*

Example 2.2. Let $\mathcal{A} = \{0, 1\}$. Let

$$\mathcal{F} = \left\{ \begin{matrix} & 1 & \\ 1 & 0 & 1 \\ & 1 & \end{matrix} \right\}.$$

The field $C(\mathcal{F})$ is called the NIZ constraint (No Isolated Zero). The extent of NIZ is 3×3 and thus NIZ is a higher order constraint.

If we make the field symmetric with regards to ones and zeros, we get the NIB constraint (No Isolated Bit). The forbidden patterns of NIB are

$$\mathcal{F} = \left\{ \begin{matrix} & 0 & \\ 0 & 1 & 0 \\ & 0 & \end{matrix}, \begin{matrix} & 1 & \\ 1 & 0 & 1 \\ & 1 & \end{matrix} \right\}.$$

In the one dimensional case the capacity of a discrete noise-free channel gives the maximum achievable coding rate for transmission over the channel. We now introduce a similar characteristic of constrained fields.

Definition 2.3. Let $E(n, m)$ be the set of admissible configurations on an n by m rectangle for a given field F. The *combinatorial entropy*, H, of F is defined as

$$H(F) = \lim_{n,m \to \infty} \frac{\log_2 |E(n, m)|}{nm}. \tag{2.1}$$

No code from an unconstrained binary source to F can have a code rate higher than $H(F)$. In many works the combinatorial entropy is referred to as the *capacity*.

One could wonder whether the limit is well-defined. Two similar proofs of this can be found in [19], [28].

Example 2.4. Consider the Hard Square constraint. In [6] the entropy of this particular constrained field has been determined to be approximately 0.5879.

2.1.1 Some examples of constrained fields

Let d, k be natural numbers and let $k > d$. The RLL(d, k) field is defined over the binary alphabet in the following manner. Both horizontally and vertically it is required that between any two "ones" the run-length of "zeroes" is at least d and at most k long. Hence the name "run-length-limited" (RLL).

Example 2.5. Consider the case where $d = 1, k = 3$. We have RLL$(d, k) = C(\mathcal{F})$, where

$$\mathcal{F} = \left\{ 11, 0000, \begin{matrix} 1 \\ 1 \end{matrix}, \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \end{matrix} \right\}.$$

In the previous example we only needed to check along rows and columns in order to see whether a constraint is violated or not. When this is the case we say that the constraint is *splitting* along rows and columns.

A variation of the RLL constraints are the symmetric run-length-limited constraints where the run-length applies to both symbols.

Definition 2.6. Let d, k be integers, $k \geq d$. The SRLL(d, k) constraint over the binary alphabet is defined as the set of configurations where the horizontal and vertical run-length of any symbol is at least d and at most k.

Example 2.7. Below we give two examples of configurations. The left is a valid SRLL(1,2) configuration, but it violates the SRLL(2,3) constraint. The right configuration is admissible with regards to the

SRLL(2,3) constraint, but violates the SRLL(1,2) constraint. Note that we do not require the constraint to be fulfilled at the borders at the configuration, i.e. we allow run-lengths of 1 for the SRLL(2,3) constraint, if it is the last symbol in a row or column.

100101100110	000111001110
110110110110	110110011000
011001001001	111000111001
100110110101	001000110001
110011001010	000111001110

Another class of constraints which is not splitting is given by imposing constraints on the local average value or density of given values of the alphabet. This constraint has some resemblance to half-toning, i.e. rendering gray scale images by binary images.

Let x_{ij} be the symbols within any $N \times M$ rectangle.

Definition 2.8. Given N and M, the binary density constraint is defined by

$$d_{min} \leq \sum_{i=1}^{N} \sum_{j=1}^{M} x_{ij} \leq d_{max} \qquad (2.2)$$

where $0 \leq d_{min} \leq d_{max} \leq NM$ and $x_{ij} \in \{0, 1\}$.

Example 2.9. Let $N = M = 3$ and $d = 3, k = 6$. An example of a valid density(d, k) constraint is then

100101100100
110110110110
011001011001
110110110101
010011001011

2.2 Representations of a field

A natural representation of a constraint in 2D is to have two labeled graphs G and H with the same labeling function such that the rows of a configuration C corresponds to paths along G whereas columns of C corresponds to paths along H [21] [20]. This representation can readily handle the splitting constraints such as 2D RLL(d, k).

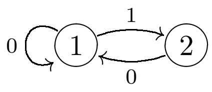


Figure 2.1: A representation of the allowed vertical and horizontal transitions for the Hard Square constraint.

Example 2.10. Consider the labeled graph in Figure 2.1. This describes the allowed horizontal and vertical transitions for the Hard Square constraint.

However, when one deals with more general constraints where the forbidden words are larger blocks one has to enlarge the alphabet. As an example consider the NIB constraint from Example 2.2 which is a higher order constraint that isn't splitting. The extent of the constraint is $N = M = 3$.

The states of the horizontal representation will then be blocks of size $N \times (M-1)$ whereas the states of the vertical representation will be blocks of size $(N-1) \times M$.

For the vertical representation there is a transition from state i to state j if there exists a configuration in $E(N, M)$, for which state i is identical to the top $N-1$ rows and state j to the bottom $N-1$ rows. State i and j have an overlap of $N-2$ rows. The label of the transition is the last row of j. The transitions for the horizontal representation is defined in a similar manner.

Example 2.11. Consider the NIB constraint from Example 2.2. NIB has extent 3×3, so a representation of the vertical transitions will have states of size 2×3 whereas the representation of horizontal transitions will have states of size 3×2.

The 3×3 configuration, G depicted below, is a valid NIB configuration.

$$G = \begin{matrix} 000 \\ 110 \\ 010 \end{matrix}$$

A vertical transition generating G will look like

$$\begin{array}{ccc} \boxed{\begin{matrix}000\\110\end{matrix}} & \longrightarrow & \boxed{\begin{matrix}110\\010\end{matrix}} \\ \\ i & & j \end{array}$$

whereas a horizontal transition generating G will look like

$$\begin{array}{ccc} \boxed{\begin{matrix}00\\11\\01\end{matrix}} & \longrightarrow & \boxed{\begin{matrix}00\\10\\10\end{matrix}} \\ \\ i & & j \end{array}$$

Unfortunately this representation of a 2D constraint is not operational in the sense that it offers no means of computing the entropy of the constraint. This is not a fault of this particular representation, however. In Section 2.5 we will consider the problem of actually computing the entropy.

2.3 Checkerboard constraints

In this section we will introduce our main example of a constrained field, the so-called checkerboard constraint.

Consider binary constraints where each 1 has to be surrounded by an arbitrary, but specific neighborhood of 0s. These *checkerboard constraints* were introduced in [26] and further investigated in [38]. In the latter article the neighborhood could be any measurable, bounded subset of the plane, but we will restrict ourselves to neighborhoods defined on the integer lattice $\mathbb{Z}^2$. The RLL(d, ∞) constraints are examples of this. Our other main example is the diamond constraint or minimum distance M between ones with regards to the 1-norm. We will refer to this constraint as $\diamond(M)$ (in [26] this constraint is referred to as the Diamond $M-1$ constraint). If we refer to the Diamond constraint without mentioning M, it is understood that $M = 3$.

The extent of the $\diamond(M)$ constraint is $M \times M$. However, unlike the RLL(d, k), the forbidden words do not split along rows and columns.

$$\mathcal{N}_{\diamond(3)} : \begin{array}{ccccc} & & 0 & & \\ & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 \\ & 0 & 0 & 0 & \\ & & 0 & & \end{array} \qquad \mathcal{N}_{2,\infty} : \begin{array}{ccccc} & & 0 & & \\ & & 0 & & \\ 0 & 0 & 1 & 0 & 0. \\ & & 0 & & \\ & & 0 & & \end{array}$$

Figure 2.2: The all-0 neighborhood, $\mathcal{N}$ for the constraints $\diamond(3)$ and RLL$(2,\infty)$.

Compare the all-0 neighborhood, $\mathcal{N}$ required around a 1 for each of the two constraints $\diamond(3)$ and RLL$(2,\infty)$ in Figure 2.2.

Given the all 0-neighborhood the extent, $N \times M$, of the checkerboard constraint is given by the smallest values of N and M, for which the $(2N-1)$ by $(2M-1)$ rectangle centered at the 1 contains the all 0-neighborhood.

2.4 Coding

Let F be some constrained field. As stated in the introduction we are interested in coding unconstrained binary data into F. Let κ be some code for F. The *rate* of a code is the ratio

$$R(\kappa) = \frac{\text{\#bits in source signal}}{\text{\#bits in coded signal}}.$$

Unlike the one dimensional case where there exists a large and powerful framework for designing constrained codes [25], [33] there is no general theory for developing codes for constrained fields.

Indeed, there have not been reported many actual codes in the literature. Examples are [47], [21], [39], [2], [41], [45], [10].

Example 2.12. Consider the following intuitive coding scheme for the Hard Square constraint. Imagine that the field F is a checkerboard. We then simply write the source bits to the "black" squares, whereas the white squares are padded with zeroes.

Decoding is easy. We just read from the black squares (or every other symbol). The coding rate of this scheme is 1/2.

This code has a lot of things going for it. If we define the efficiency of a code as the ratio between its code rate and the entropy of the constraint, the simple “checkerboard code” has an efficiency of 86%. It has limited error propagation (assuming a binary symmetric channel). It is extremely simple. So of course, we have to have this very simple code in mind for comparison when considering more advanced codes.

2.4.1 A two pass code for the RLL(d, ∞) constraint

In this section we describe a scheme for encoding a user bit stream S which is assumed to be Bernoulli(1/2) onto a finite $n \times m$ lattice such that the RLL(d, ∞) constraint is obeyed.

It is easiest to describe the coding scheme in the case where $d = 1$, i.e. the Hard Square constraint, so we will begin by dealing with that case. As we saw the simplest possible scheme is to regard the lattice as a checkerboard and write the bit stream to the black squares and then use the white squares for zero padding. This achieves a coding rate of a 1/2.

Now if we add a second coding pass to this, we will achieve a higher rate. The idea is to scan over the white squares. Each time we find a white square with four black 'zeros' as neighbors, we can write to this square without violating the constraint. If a bit is on the boundary, we will relax the constraint accordingly.

Decoding is simple. We scan over the black squares reading the data bits. After all the black squares have been processed, we scan the white squares. If a white square is surrounded by four zeros, then it contains a user bit. We have to decide on an ordering of the checkerboard. We will scan the checkerboard left to right, top to bottom, black squares first, then white squares.

Let us determine the coding rate of this scheme. Since the scheme is variable rate we will find the average code rate. Each time a white square is surrounded by four zeros we can encode an extra user bit. On average this will happen 1 out of 16 times, since each of the 2^4 different configurations of neighbors are equally likely (we assumed that the user bits were Bernoulli(1/2)). That is the average rate R is

$$R = \frac{1}{2} + \frac{1}{16}\frac{1}{2} = \frac{17}{16}\frac{1}{2}.$$

Now, let $d > 1$. We will extend the idea of a checkerboard slightly. We still have black squares, but following each black squares are now d white squares. The black squares are still arranged diagonally. As in the previous case, we begin by writing the user bits to the black squares. This achieves a rate of $\frac{1}{d+1}$. Assume for the moment that we only write to the white diagonals to the right of a black diagonal. At first glance a free position will only occur once in $2^{4(d+1)}$. However, because we write to diagonals, the neighbors of a particular white square will either be black (there are four of these) or belong to a different white diagonal (which are already zero). Hence free positions will occur one time out of 16. See below for the case $d = 2$.

$$\begin{array}{ccccc} & & W_2 & & \\ & & B & & \\ W_2 & B & \cdot & W_2 & B \\ & & W_2 & & \\ & & B & & \end{array}$$

All in all, we get the average coding rate of

$$R = \frac{17}{16(d+1)}.$$

We can increase the rate by writing to any free positions in the remaining white diagonals, but it is not clear how to analyze the gain, since the number of free positions depend on the values stored in previous free positions.

2.4.2 Bit-stuffing

We will now discuss a coding scheme which is very suitable for the hard square constraint. Instead of padding every user bit with a zero, i.e. the checkerboard code of the previous section, the idea of bit-stuffing is to pad with zeros only whenever it is necessary.

We need to define an order of writing to a rectangle. We will use normal writing order: that is left to right, top to bottom one row at a time. One could write along diagonals as well or use more sophisticated orderings.

Bit-Stuffing works as follows. Let x denote some previously written symbol whereas y is used to illustrate the idea of "next available position":

- '0' is written as a '0' at the next available position.

$$0 \mapsto \begin{array}{|c|c|c|}\hline x & 0 & y \\ \hline & & \\ \hline\end{array}$$

- '1' is written as a '1' at the next available position. However, the position next to (to the right of) the written '1' and below it is immediately "stuffed" with '0'.

$$1 \mapsto \begin{array}{|c|c|c|}\hline x & 1 & 0 \\ \hline y & 0 & \\ \hline\end{array}$$

  This stuffing does not wrap around. If a '1' is written at a right border position, there is no stuffed '0' next to it. Only below. Similarly for '1's on the bottom border.

Example 2.13. Consider writing the data sequence 1011101001 to a 5×4 rectangle.

$$1011101001 \mapsto \begin{array}{|c|c|c|c|c|}\hline 1 & 0 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 0 & 1 \\ \hline 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 \\ \hline\end{array}$$

Note that bit-stuffing is a variable rate coding scheme. Hence we talk about its *average* coding rate. But it is difficult to calculate this analytically. One easy alternative is to find an average coding rate by simulation.

The bit-stuffer described can easily be extended to any checkerboard constraint. Each time we write a '1', we immediately stuff with zeros filling out the 0-neighborhood around the written '1'. We only stuff in those positions that can occur as next available positions, however. For instance, in the case of RLL(d, ∞) we stuff with d zeros to the right and below each '1' where applicable.

Example 2.14. Consider a bit-stuffer for the Diamond constraint. A '0' is simply written as a '0' at the next available position, whereas a '1' is written at the next available position and the 0-neighborhood is stuffed with ’0’s. This is depicted below. x denotes the previously written symbol and y denotes the next available position.

$$1 \mapsto \begin{array}{|c|c|c|c|c|c|} \hline x & 1 & 0 & 0 & y & \\ \hline 0 & 0 & 0 & & & \\ \hline & 0 & & & & \\ \hline \end{array}$$

2.4.3 Biased bit-stuffing

We will present a modification to bit-stuffing based on the concept of a biased stream. Since we pad with ’0’s whenever we write a ’1’, it makes sense that we would use less space if the data contained less ’1’s. Normally, we assume that the data is unbiased, that is the output of a Bernoulli(1/2) source. We now introduce a precoding step which transforms the unbiased data into a biased stream.

Let p_1 denote the probability of a ’1’ in the biased stream. If $p_1 < 1/2$ the average coding rate of the bit-stuffer R_b will increase towards one. However, this come at a cost, namely the rate of the precoder. If the bias is severe, the rate of the precoder will be very low, eliminating the gain in rate of the bit-stuffer. We view the biased bit-stuffer as the composition of the mappings of the precoder and the ordinary bit-stuffer.

$$\text{Data } X \longrightarrow \text{Precoder}(p_1) \longrightarrow \text{Bitstuffer}$$

There exists precoders which can transform unbiased streams into p_1-biased ones at the (asymptotic) rate of $H(p_1)$, where H is the binary entropy function. One can think of such a transformation as the inverse to an arithmetic encoder for a p_1-biased stream.

The coding rate R_{mb} of a biased bit-stuffer is then

$$R_{mb} = H(p_1)R_b. \tag{2.3}$$

We have investigated (2.3) by simulation for the Hard Square constraint. By optimizing over p_1 we were able to get a coding rate around 0.58 which is within 2% of the entropy of the Hard Square constraint.

The idea of using biased streams and precoding can be extended further by having more than one biased sequence and choose between them depending on past data besides what is prescribed by the constraint. This was utilized by Roth et al. [41] to determine and optimize the entropy of the bit-stuffing scheme for the Hard Square constraint. fields. They carried out a detailed analysis for performing bit-stuffing along the diagonals rather than row-by-row as we have done. It should be noted that they derived analytical bounds on their scheme showing that its coding rate was within 1% of the entropy.

For the higher order constraints 2D RLL (d, ∞) lower bounds on the entropy of the bit-stuffing scheme are presented in [20].

In Chapter 4 we will present a variation of the basic bit-stuffing scheme applicable to all checkerboard constrained fields, where we can calculate the coding rate explicitly.

2.5 Computability of the entropy

In the one dimensional case the combinatorial entropy of a constraint $\mathcal{C}$ is easily obtained. Recall, that sequences satisfying a constraint on N consecutive symbols such as run-length-limited sequences may be described by finite state sources, where a state is characterized by $N-1$ symbols. The transfer (or adjacency) matrix $\mathbf{T}$ of the source indicates the possible transitions between two states. The largest eigenvalue λ of the transfer matrix $\mathbf{T}$ determines the growth rate of the number of configurations [32]. Taking the logarithm gives the maximum entropy [44]:

$$H = \log_2 \lambda. \tag{2.4}$$

Unfortunately, in the two dimensional case things are not as simple. Recall from the discussion in Section 2.2 that we do not have an operational representation of a constrained field. Only in very few cases do we have analytical expressions for the entropy.

What is worse: This appears to be a general problem. Indeed, the problem of whether $H(F) > 0$ for a general constrained field F is undecidable [4]. In [32] it is shown that the symbolic dynamics concept of topological entropy is what we have defined as the combinatorial entropy. Unfortunately the topological entropy for general 2D shift spaces is uncomputable [24].

In order to make precise the notion of uncomputable, we will need a computational model. We will use an abstract computer: A RAM machine [23]. However, we do not wish to get into the details of the model at this point. Suffice to say that a RAM machine is very close to a modern day computer (as opposed to a Turing machine).

Definition 2.15. A number x is *computable* if given $\epsilon > 0$ there exists a natural number N_ϵ and an algorithm such that a RAM machine running the algorithm in N_ϵ steps will output a rational number r with $|x-r| < \epsilon$.

Even though the topological entropy for general 2D shift spaces is uncomputable, we can still hope that the combinatorial entropy is computable for some class of constrained fields. Indeed, the following options present themselves.

- Search for a more restricted class of fields, where it is possible to compute the entropy.
- Search for good upper and lower bounds on the entropy.

While the first approach has been successful for some classes of two-dimensional symbolic dynamical systems [30], these have additionally algebraic structure that isn't suitable for modeling communication aspects. In Section 2.6.3 we show that the entropy of a checkerboard constraint is in fact computable, but unfortunately not in a tractable way. Hence we turn to the second possibility.

This approach might seem counter-intuitive. How can we on one hand have that the topological entropy is uncomputable and on the other hand look for bounds that converge to the true value. It turns out that for the bounds we do find we do not know the order of convergence, so we cannot in advance know how many times we must run the algorithm to obtain the desired accuracy. Thus there is no conflict with the uncomputability of the entropy. Indeed, the whole purpose of Definition 2.15 was to resolve what at first glance appeared to be a paradox.

In the next section we introduce some upper and lower bounds on the combinatorial entropy based on one dimensional techniques. The reminder of the thesis deals with refining these bounds and see how they relate to coding schemes.

Finally we remark that the coding rate of an actual code for a constrained field will provide us with a constructive lower bound for the entropy of that field. This means that it may be worthwhile to investigate coding schemes that have severe practical drawbacks because their coding rate can provide an estimate (lower bound) on the entropy. For instance the method of bit-stuffing is variable rate and has unlimited error-propagation, which are serious drawbacks in a practical setting. Nevertheless, the method achieves very high coding rates for checkerboard constraints. This has been utilized to obtain very tight lower bounds on the entropy of the RLL(d, ∞) constrained fields [20]. We will present a variation of the bit-stuffing scheme in Chapter 4 which yields even tighter bounds on the entropy of the RLL(d, ∞) fields for $d = 2, 3$ and 4.

2.6 Bounding the entropy with a 1D method

In this section we will introduce a 1D process that "behaves" like a constrained field. The entropy of the process can be calculated with 1D methods and by utilizing a sequence of processes that expands to the field we can approximate the entropy of the field.

The idea of using a 1D approximation is natural and have been explored in [6], [2] and [15], [16].

Consider a *band*, that is a 2D array of finite (horizontal) width m and arbitrary (vertical) height n. We think of the band as being generated one row at a time by a finite state source and growing in the vertical direction.

The admissible configurations of an array of width m may for all n be described by a finite state source. For a constraint of extent $N \times M$, the states of the source are given by the symbols on the $(N-1) \times m$ segment which appear as the first or last $N-1$ rows of an admissible configuration on a $N \times m$ rectangle, i.e. a configuration of $E(N, m)$. A transition from state i to state j is admissible if there is a configuration in $E(N, m)$, for which state i is identical to the top $N-1$ rows and state j to the bottom $N-1$ rows. State i and j have an overlap of $N-2$ rows. The last row of j is generated by the transition from i to j and appended to the previous rows of the output. Any admissible configuration of $E(n, m)$ with fixed m and n $(> N-1)$ rows may be generated as an output by starting the source in the state specified by the first $N-1$ rows and making $n-N+1$ transitions appending one row to the output in each transition.

The transfer matrix $\mathbf{T}_m = (t_{ij})$ indicates which transitions that satisfy the constraint by defining the elements $t_{ij} = 1$ if the transition from state i to j is admissible and $t_{ij} = 0$ if it is not admissible. According to the Perron-Frobenius theorem [43] this non-negative matrix has a unique positive largest eigenvalue, λ_m and the (per row) entropy, H_B, of the band source is therefore given by Equation (2.4):

$$H_B(m) = \log_2 \lambda_m.$$

Hence the per symbol entropy of the source on an array of width m ($n \to \infty$) is given by,

$$\frac{H_B(m)}{m} = \frac{\log_2 \lambda_m}{m}. \tag{2.5}$$

Since we are not considering constraints beyond the width of the band (there might be cases where we are not able to expand a valid band to an admissible configuration of greater width) the per symbol entropy of the band will be greater than the entropy of the field. However, it is clear that as the width of the band $w \to \infty$ the per symbol entropy of the band source will converge to the entropy of the constrained field. Hence Equation (2.5) provides a sequence of upper bounds on the entropy H defined by (2.1).

In Appendix B we have collected the entropy of a large number of band sources for different constrained fields.

2.6.1 Some upper bounds from band sources

While it appears easy to utilize the upper bound (2.5) there quickly arises a number of practical problems. We refer the reader to Appendix A where we discuss some of these in greater detail. Here we merely note that the number of states in the transfer matrix grows exponentially in the width of the band. This makes it necessary to use some sort of iterative method for computing the largest eigenvalue of the matrices as they quickly become very large. We have used the power method [8]. Again we refer the reader to Appendix A where we offer more detail.

In Table 2.1 we have collected some upper bounds for the RLL(d, ∞) constrained fields for $d = 2, 3, 4$. As can be seen from the table, the rate of convergence is very slow. Nevertheless the method of band sources provides a relatively easy means of obtaining upper bounds for any constrained field.

RLL$(2,\infty)$		RLL$(3,\infty)$		RLL$(4,\infty)$	
w	H_U	w	H_U	w	H_U
12	0.4574	10	0.3849	9	0.3387
13	0.4564	11	0.3833	10	0.3366
14	0.4557	12	0.3820	11	0.3347
15	0.4550	13	0.3809	12	0.3332
16	0.4544	14	0.3800	13	0.3320
17	0.4539	15	0.3791	14	0.3309
18	0.4534	16	0.3784	15	0.3299

Table 2.1: Upper bounds H_U based on a band source of width w for the constrained fields RLL(d,∞) for $d = 2, 3, 4$.

Table 2.2 and Table 2.3 give similar upper bounds for the NIB and NIZ constraints and the SRLL(1,2) and SRLL(2,3) constraints, respectively. Again, we note that the rate of convergence is very slow.

w	NIB	NIZ
6	0.9475554710	0.9732
7	0.9441683911	0.9716
8	0.9415444907	0.9703
9	0.9396501864	0.9694
10	0.9380681117	0.9686

Table 2.2: Upper bounds H_U based on a band source of width w on the entropy of the NIB and NIZ constrained fields.

2.6.2 Estimating the entropy

The upper bounds obtained by a band source converge to the entropy. However, the computational complexity of computing the transfer matrix makes investigating the bounds impossible but for a few (small) widths of the band source. It is interesting to see how far from the true value of the entropy these upper bounds are.

If we subtract the per row entropy of a band source of width n from

w	SRLL(1,2)	w	SRLL(2,3)
7	0.5249	9	0.2678
8	0.5179	10	0.2650
9	0.5123	11	0.2627
10	0.5079	12	0.2610
11	0.5043	13	0.2602
12	0.5013	14	0.2583
13	0.4987	15	0.2570

Table 2.3: Upper bounds H_U on the entropy of the SRLL(1,2) and SRLL(2,3) constrained fields. The bounds are obtained with a band source of width w.

RLL$(2,\infty)$		RLL$(3,\infty)$	
$H_B(10) - H_B(9)$	0.44548	$H_B(7) - H_B(6)$	0.36851
$H_B(11) - H_B(10)$	0.44549	$H_B(8) - H_B(7)$	0.36772
$H_B(12) - H_B(11)$	0.44549	$H_B(9) - H_B(8)$	0.36743
$H_B(13) - H_B(12)$	0.44549	$H_B(10) - H_B(9)$	0.36744
$H_B(14) - H_B(13)$	0.44549	$H_B(11) - H_B(10)$	0.36752
$H_B(15) - H_B(14)$	0.44549	$H_B(12) - H_B(11)$	0.36752
$H_B(16) - H_B(15)$	0.44548	$H_B(13) - H_B(12)$	0.36752
$H_B(17) - H_B(16)$	0.44549	$H_B(14) - H_B(13)$	0.36751
$H_B(18) - H_B(17)$	0.44549	$H_B(15) - H_B(14)$	0.36751

Table 2.4: Entropy estimates based on difference of band entropies for RLL$(2,\infty)$ and RLL$(3,\infty)$.

the per row entropy of a band source of with $n+1$ we in a way estimate the per symbol entropy.

Example 2.16. Consider the RLL$(4,\infty)$ constraint. Calculating the entropy H_B of a band source of width 15 and 14, respectively, for the constraint we get $H_B(15) = 4.9485$ and $H_B(14) = 4.6321$. This yields the entropy estimate

$$\tilde{H} = H_B(15) - H_B(14) = 0.3164.$$

Compare this estimate with the upper bounds of Table 2.1, where the best upper bound is 0.3299.

We can make this more precise. Let H be the entropy of some constrained field F. Let $H_B(n)$ denote the entropy of a band source of width n for F. We know that

$$\lim_{n\to\infty} \frac{H_B(n)}{n} = H.$$

The question is whether $H_B(n+1) - H_B(n)$ converges to H for $n \to \infty$.

Due to the following standard result on sequences [3] we can show that if the difference converges then it converges to the true entropy H.

Lemma 2.17. Let $(x_n)_n$ be a sequence of real numbers. Then the following holds:

$$\liminf \left|\frac{x_{n+1}}{x_n}\right| \leq \liminf |x_n|^{1/n} \leq \limsup |x_n|^{1/n} \leq \limsup \left|\frac{x_{n+1}}{x_n}\right|$$

Proposition 2.18. *Let H be the entropy of the constrained field F. Let $H_B(n)$ denote the entropy of a band source of width n for F. If the sequence $(H_B(n+1) - H_B(n))$ converges, then it converges to H.*

Proof. Assume that

$$(H_B(n+1) - H_B(n)) \to x \text{ for } n \to \infty$$

for some $x \in \mathbb{R}$. Recall that $H_B(n) = \log_2 \lambda_n$, where λ_n is the largest eigenvalue for the band source. Hence we have

$$H_B(n+1) - H_B(n) = \log_2 \lambda_{n+1} - \log_2 \lambda_n = \log_2 \frac{\lambda_{n+1}}{\lambda_n} \to x.$$

Therefore $\frac{\lambda_{n+1}}{\lambda_n} \to 2^x$. According to Lemma 2.17 with $x_n = \lambda_n$ we then have

$$\lambda_n^{1/n} \to 2^x$$

or equivalently

$$\frac{H(n)}{n} = \log_2 \lambda_n^{1/n} \to x$$

and we conclude that $x = H$. $\square$

Regard Table 2.4 where we have given estimates of the entropy of RLL$(2, \infty)$ and RLL$(3, \infty)$ based on this method. It can be seen that the estimate seems to converge. Indeed, experiments for all checkerboard constraints considered here, as well as the NIZ and NIB constraints seems to indicate that the estimates converges.

It is not clear, though, how to actually decide whether the estimates converge. For "well-behaved" constraints like the checkerboard constraints in Table 2.4 it seems reasonable to assume that the values indeed converge, but we have no criteria that assures convergence.

	Den(2,4)	Den(4,5)
$H_B(6) - H_B(5)$	0.66496	0.46171
$H_B(7) - H_B(6)$	0.69092	0.47924
$H_B(8) - H_B(7)$	0.68929	0.47422
$H_B(9) - H_B(8)$	0.67971	0.47656
$H_B(10) - H_B(9)$	0.68929	0.48164

Table 2.5: Entropy estimates based on difference of band entropies for some density constraints.

Indeed, for other constraints one may encounter difficulties. Regard Table 2.5 where we have given estimates for the density(2,4) and density(4,5) constraint. Note that these estimates fluctuate much more than the ones for the RLL constraints. This might be due to the bands having smaller widths (due to complexity issues, it is difficult to use larger bands), but it also might have to do with the fact that the density constraints are not checkerboard constraints or lack some other property that ensures that the band differences actually converge.

SRLL(1,2)		SRLL(2,3)	
$H_B(8) - H_B(7)$	0.4689	$H_B(10) - H_B(9)$	0.2393
$H_B(9) - H_B(8)$	0.4678	$H_B(11) - H_B(10)$	0.2400
$H_B(10) - H_B(9)$	0.4681	$H_B(12) - H_B(11)$	0.2428
$H_B(11) - H_B(10)$	0.4681	$H_B(13) - H_B(12)$	0.2505
$H_B(12) - H_B(11)$	0.4684	$H_B(14) - H_B(13)$	0.2328
$H_B(13) - H_B(12)$	0.4678	$H_B(15) - H_B(14)$	0.2395

Table 2.6: Estimate $\tilde{H} = H_B(n+1) - H_B(n)$ of the entropy of the SRLL(1,2) and SRLL(2,3) constrained fields.

The symmetric RLL constraints demonstrates a similar behavior. Regard Table 2.6 where we have collected the entropy estimates for the SRLL(1,2) and SRLL(2,3) constraint. While estimates for SRLL(1,2) seem to fluctuate around the value 0.468, it is much more difficult to guess at a “true” value for the SRLL(2,3) constraint as the estimates fluctuate much more. It is not clear from the small number of elements in the sequence whether the fluctuations become smaller for SRLL(2,3), so that the values converge or whether it is not the case.

For completeness we offer what appears to be converged estimates in Table 2.7 for the NIZ constraint. In contrast to this we have shown some estimates of the entropy for the NIB constraint based on corresponding widths of the band sourcesas but in this case the estimates have not yet converged.

Based on the examples given one should exercise caution when utilizing Proposition 2.18. Nevertheless, as can also be seen it offers an easy way to obtain what appears to be good estimates for many constrained fields.

2.6.3 Lower bounds from bands

Can we use the entropy of the band (2.5) to obtain a lower bound on the entropy of the field?

In some cases, we can. Assume that we have two arbitrary bands X and Y of width m over some field F with entropy $H(F)$. If it is always possible to find a merging array V of width c such that the band XVY

	NIB	NIZ
$H_B(6) - H_B(5)$	0.9237815934	0.9619299
$H_B(7) - H_B(6)$	0.9238459123	0.9613020
$H_B(8) - H_B(7)$	0.9231771872	0.9617083
$H_B(9) - H_B(8)$	0.9244957524	0.9616608
$H_B(10) - H_B(9)$	0.9238294390	0.9616609

Table 2.7: Estimate $\tilde{H} = H_B(n+1) - H_B(n)$ of the entropy of the constrained fields NIB and NIZ.

is admissible then the following lower bound on the entropy holds:

$$\frac{H_B(m)}{m+c} \leq H(F). \tag{2.6}$$

We note that for the RLL(d, ∞) constraints this is always the case. We can simply use a merging array consisting of d columns of zeros. Utilizing this we can complement the upper bounds of Table 2.1 with the lower bounds found in Table 2.8. As can be seen, there is quite a large gap between the upper and lower bounds obtained in this fashion.

RLL$(2, \infty)$		RLL$(3, \infty)$		RLL$(4, \infty)$	
w	H_L	w	H_L	w	H_L
12	0.3920	10	0.2961	9	0.2345
13	0.3956	11	0.3012	10	0.2404
14	0.3987	12	0.3056	11	0.2455
15	0.4015	13	0.3095	12	0.2499
16	0.4039	14	0.3129	13	0.2538
17	0.4061	15	0.3160	14	0.2573
18	0.4081	16	0.3186	15	0.2604

Table 2.8: Lower bounds H_L for the entropy of RLL(d, ∞) constraints.

The idea of the merging array for the RLL(d, ∞) constraint can be extended to any checkerboard constraint. Let F be a checkerboard constraint of extent $N \times M$. Then it is clear that the band of width M

consisting of all zeros can be used as a merging array between any valid bands over F.

We can use this observation to show that the entropy of a checkerboard constraint is computable.

Proposition 2.19. *The entropy $H(F)$ of any checkerboard constraint F is computable.*

Proof. Let $N \times M$ be the extent of F and let B be a band source of width n for F. As we have previously seen the entropy of the band source offers an upper bound on the entropy of F. On the other hand we have just argued that since the extent of F in the horizontal direction is M we can use a merging array of all zeros for any two bands from B. Thus we can use Equation (2.6) with M in place of c. All in all we have the following bounds on the entropy:

$$\frac{H_B(n)}{n} \geq H(F) \geq \frac{H_B(n)}{n+M}.$$

Considering the difference between the upper and lower bound we have

$$\begin{aligned}\frac{H_B(n)}{n} - \frac{H_B(n)}{n+M} &= \frac{H_B(n)(n+M) - H_B(n)n}{n(n+M)} \\ &= \frac{H_B(n)M}{n(n+M)} \\ &\leq \frac{nM}{n(n+M)} = \frac{M}{n+M}\end{aligned}$$

where we used that the band entropy $H_B(n) \leq n$ for any n. Thus for a given precision $\epsilon > 0$ we have that

$$\left| \frac{H_B(n)}{n} - H(F) \right| \leq \epsilon$$

provided that $M/(n+M) \leq \epsilon$ or equivalently that $n \geq \frac{1-\epsilon}{\epsilon} M$. Since we can bound the number of computations needed to calculate $H(n)$ to K^n, where K is some suitably chosen constant, we have thus shown that $H(F)$ is computable in accordance with Definition 2.15. □

As can be seen from the proof of the proposition, computable does not equal tractable. Indeed, in order to assure an accuracy of say 10^{-4}

we would have to compute the entropy of a band of width 10^4. Since bands of widths 30 are beyond the computational resources of almost any organization, this is clearly not an option.

Example 2.20. Let us consider the NIZ and NIB constraints for a moment. Can we always find a merging array for these two constraints? NIZ is easy. Since the only forbidden word is an isolated 0, we can only violate the constraint by using a 1. Hence a single column of 0s can be used as a merging array between any two valid NIZ bands and we get the following lower bound on the entropy of the NIZ constraint

$$H \geq \frac{H_B(w)}{w+1},$$

where $H_B(w)$ is the entropy (per row) of a band source of width w for NIZ.

Let us turn to the NIB constraint. First we note that we cannot always make do with a single column as we could for NIZ. Consider the following configuration. No matter how we choose c_2 we cannot make the configuration valid.

$$\begin{array}{ccccc} & 1 & c_1 & 0 & \\ 1 & 0 & c_2 & 1 & 0 \\ & 1 & c_3 & 0 & \end{array}$$

If we try to avoid isolating the 0 in the left pattern, c_2 must be 0, but this makes the right hand pattern forbidden and vice versa. However, this offers a hint that we can always merge any two valid NIB arrays X and Y by using two columns for a merging array $C = C_1C_2$ in the following fashion.

The left column of C is simply the last column in X and the right column in C is the first column in Y. C itself is valid, since the extent of the forbidden words are 3×3. XC_1 is valid, since we can only violate the constraint by having complementary symbols next to each other. For the same reason C_2Y is valid. Let us then consider XC_1C_2Y. Since both XC_1 and C_2Y are valid, any forbidden word in XC_1C_2Y must have two of its columns in C. But the patterns 101 and 010 do not occur in $C_1C_1C_2C_2$ by construction, so we conclude that XC_1C_2Y is valid.

Hence we have the following lower bound on the entropy of the NIB constraint:

$$H \geq \frac{H_B(w)}{w+2}.$$

In Table 2.9 we have collected some lower bounds on the entropy obtained in this fashion. Again there is a rather large gap between these lower bounds and the upper bounds in Table 2.2. Furthermore, it can be seen that the lower bounds are further from the entropy estimates of Table 2.7 than the upper bounds are. As the widths of the bands grow towards infinity, the lower bound and the upper bounds will converge to the entropy, but for the small band widths considered here, the width of the merging array is considerable in comparison to the total width of the band.

It appears to be somewhat more involved to construct merging arrays for the SRLL and density constraints. However, in Section 2.7 we will use a variation of the merging array idea to obtain lower bounds on the entropy for these two constraints.

w	NIZ	NIB
6	0.8342	0.8122
7	0.8501	0.8261
8	0.8625	0.8370
9	0.8724	0.8457
10	0.8805	

Table 2.9: Lower bounds on the entropy of the NIZ and NIB constraints using a band source of width w and a merging array of width 1 and 2, respectively.

Finally it may be remarked that there are constraints for which it is clear that one cannot find a finite (in the horizontal direction) merging configuration for any two valid configurations. The standard example is domino tiling, where the whole plane is tiled by one by two vertical and horizontal domino pieces. Consider the case where the array X has a vertical zig-zag boundary of all horizontal pieces where the piece in every other row is displaced one position relative to its two neighbors. In this case there is only one solution extending off the boundary of X, namely that which locks up with the boundary, except for the first and last piece of the zig-zag. (By induction it is seen that for any finite width merging array, a long enough zig-zag boundary can lead to a conflict with the opposing boundary.)

2.6.4 Improved upper bounds by using cylinder sources

Calkin and Wilf were the first to produce tight upper and lower bounds for the entropy of the Hard Square constraint [6]. Their methods depend crucially on certain symmetry requirements of the transfer matrices involved in describing the constraint and are only applicable to first order constraints. In [16] some combinatorial techniques inspired by the symmetry requirement, but applicable for higher order constraints are investigated by Forchhammer and Justesen.

We will only refer to part of their work, namely that the upper bound on the combinatorial entropy by (2.5) can be further tightened by considering cylinder sources rather than bands.

Let T_m be the transfer matrix of some band source of width m for some particular constraint of extent $N \times M$. Let $p > 1$ be an integer and set $n = p + N - 1$. Define the matrix $A_m = T_m^p$. The element a_{ij} gives the number of valid configuration of size $n \times m$ where the first $N - 1$ rows correspond to state i of T_n and the last $N - 1$ rows correspond to state j.

The following definition is from [16]:

Definition 2.21. A configuration on a *two-seam cylinder* is given by a pair of configurations which are the output of two A_m transitions having identical starting states i and identical ending states j. The A_m configurations belong to $E(n, m)$.

We say that the width of the two-seam cylinder is $2p$, where $p = n - (N - 1)$.

Similarly to the band sources, we can make a cylinder source for a two-seam cylinder of fixed width $2p$ and $n \to \infty$. Let $H_C(2p)$ denote the entropy of the two-seam cylinder source. It is then shown in [16] that

$$H \leq \frac{H_C(2p)}{2p}. \tag{2.7}$$

Furthermore it is shown that Equation (2.7) for small values of p provide very tight upper bounds on the entropy of the fields considered.

But how do we actually compute the entropy H_C of the cylinder source? It turns out that we can model a two-seam cylinder by an ordinary band where the rows have the following form:

$$r = \overbrace{r_0 \dots r_{d-1}}^{i} \underbrace{r_d \dots \overbrace{r_{n-d} \dots r_{n-1}}^{j}}_{p} \underbrace{\overbrace{r_n \dots r_{n+d-1}}^{i} r_{n+d} \dots}_{p} \overbrace{r_{2n-d} \dots r_{2n-1}}^{j} .$$

Here $d = N - 1$ correspond to the height of the states of T_m and the parts of the row marked i and j correspond to states of A_m, i.e. they have to be identical.

Let $\tilde{B}$ denote the band source modeling the two-seam cylinder. Then $H_C = \log_2 \lambda_{\tilde{B}}$, where $\lambda_{\tilde{B}}$ is the greatest eigenvalue of $\tilde{B}$ according to Equation (2.4).

Hence we can rewrite the cylinder upper bound as

$$H \leq \frac{\log_2 \lambda_{\tilde{B}}}{2p}. \tag{2.8}$$

Consider Table 2.10 where we have collected some values of the entropy of the two-seam cylinders for the RLL(d, ∞) constraints for $d = 2, 3$ and 4. Note that the least value of p for which we can construct a valid two-seam cylinder varies with the constraint as $p \geq d$. Results are shown with the assured accuracy as detailed in Appendix A.2.1. It was not possible to construct two-seam cylinders for the RLL$(3, \infty)$ and RLL$(4, \infty)$ for widths greater than $2p = 16$ using the available computational resources.

If we compare the upper bounds with the ones of Table 2.1 obtained by utilizing band sources, we note that the new upper bounds are tighter. Indeed, they quickly approach what we estimated the entropy to be in Table 2.4.

We have also employed the cylinder upper bound on the Diamond constraint. Results are shown in Table 2.11.

2.7 Cascading arrays

In [11] Etzion studies 2D SRLL(d, k) constraints. He discusses methods for constructing a merging array, W, given any two admissible arrays, X and Y for this particular constraint. Furthermore, he gives expressions for the minimum width of the merging array, W, for which merging is always possible in terms of d.

p	RLL$(2,\infty)$	RLL$(3,\infty)$	RLL$(4,\infty)$
2	0.4807988		
3	0.4541391	0.3963193	
4	0.4480430	0.3782551	0.3413823
5	0.4465341	0.3720841	0.3281802
6	0.4462489	0.3698155	0.3225834
7	0.4460829	0.3688255	0.3200389
8	0.446002	0.368555	0.318804
9	0.445942		

Table 2.10: Upper bounds on the entropy of RLL(d,∞) constraints obtained by using a two-seam cylinder source of width $2p$.

p	$\diamond(3)$
5	0.359422
6	0.357945
7	0.356841
8	0.356019
9	0.355386
10	0.354878
11	0.354462
12	0.354116

Table 2.11: Upper bounds on the entropy of the Diamond constraint obtained by using a two-seam cylinder source of width $2p$.

This of course can be utilized to obtain lower bounds for the entropy of SRLL(d, k) constraint. As we saw in Section 2.6.3, however, these bounds were not tight. We will now explore a different approach whereby we hope to improve the lower bounds on the entropy.

Instead of designing the merging array, W after X and Y, we shall construct W first and then examine how X and Y independently may be constrained such that the cascading $WXWYW$ is admissible. The merging arrays are simpler and easier to identify (e.g. for synchronization) and we hope to obtain better lower bounds on the entropy. Alternatively, we independently constrain the arrays X, Y and Z, such that the cascading XYZ is admissible.

2.7.1 Periodic merging arrays

Let F be some arbitrary but fixed constrained field of extent $M \times M$. When we in the sequel says that some configuration or array is admissible it means it is a valid configuration of F.

Consider a 2D array W of width w and infinite height. This array is repeated at intervals of $m + w$ columns horizontally. This leaves arrays of width m undefined in between the repetition of two merging arrays.

Let X be another array of width m and infinite height. Let WX be the concatenation of the two, horizontally placing them with their boundaries next to each other.

Definition 2.22. Let the array W of width w be periodic vertically with the period p. Let WXW and WYW be arrays admissible according to the constraint. The periodic array W is a *periodical merging array* if any pair of arrays, WXW and WYW, may be cascaded to form the admissible array $WXWYW$.

The admissible array WXW in Definition 2.22 is called a *$W-$boundary constrained array.*

For $w \geq M-1$ any pair of admissible arrays, WXW and WYW, may be cascaded to form $WXWYW$. Thus these arrays are $W-$boundary constrained arrays. The reason is that the merging array separates the arrays X and Y in the sense that no $N \times M$ rectangle contains symbols from both X and Y. Further they are bounded by identical merging arrays. For some constraints, periodic merging arrays W of width

$w < M-1$ may be specified such that the property of the $W-$boundary constrained array still holds. Repeating the construction adding one array at a time, the cascading of arrays may extend to define an admissible configuration in the entire plane.

The admissible arrays WXW may be described by a finite state source with states of height $n \geq N-1$. For a merging array period $p \leq n+1$ the phase of the period is contained in the transition in the part of the states given by symbols of W. For $p \geq n+1$ the phase information could still be uniquely defined by the states. If this is not the case the (missing) information about the phase has to be added to the states. The per symbol entropy for the merged array obtained by the use of periodic merging arrays is

$$H = \frac{H_W(m)}{m+w}, \tag{2.9}$$

where $H_W(m)$ is the entropy of the W-boundary constrained array WXW in which X and W have widths m and w, respectively.

The array WXW may be perceived as a cylinder of circumference $m+w$ where the w columns of W are predefined. Thus for a given constraint a periodic merging array may be determined by finding a periodic sequence on $w \geq M-1$ columns of the cylinder.

Besides facilitating the use of the finite state source description, the periodic arrays may have the desired property that they provide easy synchronization.

2.7.2 Periodic mergings arrays for SRLL

For the SRLL(d,k) constrained fields we propose the following SRLL periodic merging array of width $w = 2d$ and vertical period $2d$.

Let $x = 0^d 1^d$ be a row of d 0s followed by d 1s. Let x_d be d identical rows of x on top of each other. Let $\bar{x}$ be the bitwise negation of x. The periodic configuration of W is given by alternating between x_d and $\bar{x}_d$.

Example 2.23. For the SRLL$(2,3)$ constrained field the periodic merging array will look like:

$$\begin{array}{l} p \left\{ \begin{array}{l} 0011 \\ 0011 \\ 1100 \\ 1100 \end{array} \right. \\ \quad\;\; \begin{array}{l} 0011 \\ 0011 \\ 1100 \\ 1100 \\ \vdots \end{array} \end{array}$$

Using SRLL periodic merging arrays, W, admissible arrays WXW are W-boundary constrained arrays whether or not $w = 2d \geq k = M - 1$ is satisfied. For $p = 2d \leq n + 1$ the period is contained within the transition of the finite state source.

The SRLL periodic merging array, W, above is both R and L d balanced by the definition of Etzion [11]:

Definition 2.24. An admissible array is R (L) balanced if the last (first) columns of W are equal and the column before (after) these d columns is the complement of each of these d columns.

This property ensures that for any admissible array, X, of width $m \leq k - d$, WXW is also a W-boundary constrained array, i.e. WXW is admissible. Therefore a simple lower bound on the combinatorial entropy of an SRLL(d, k) constraint is

$$H \geq \frac{H(k-d)}{k+d}. \tag{2.10}$$

where $H(k - d)$ is given by (2.5).

Example 2.25. For SRLL$(2, 3)$, the periodic merging array has the period $p = 2d = 4$. The finite state source for WXW therefore also has a period of four and thereby four phases. The $W-$boundary constrained array X has the property that the column in each side which is next to the boundary column is the negation of the nearest boundary column of W.

Let us return to the work of Etzion [11]. He presents methods to merge any two SRLL constrained arrays (of finite width). The width w of the merging arrays is given as a function of (d, k).

Etzion makes a distinction between three different cases:

Case 1: If $k \geq 4d - 2$ then $w = 2d$.

Case 2: If $4d - 3 \geq k \geq 2d$ then $w = 4d$.

Case 3: If $2d - 1 \geq k \geq d + 1$ then

$$w = 3d(\lceil \frac{d-1}{k-d} \rceil + 1). \tag{2.11}$$

In the first case the width $w = 2d$ is shown to be optimal. In this case Etzion's method is more efficient than using a periodic merging arrays of width $w = 2d$. For Cases 2 and 3, an entropy comparison will be a trade-off of the periodic merging arrays having a smaller width (w) versus the higher entropy of an array X of a given width (m), when it is not boundary constrained.

For SRLL(2,3) the periodic merging array is $w = 4$ compared to a width (2.11) of $w = 12$ for Etzion's merging array (Case 3.) (Etzion considered the more general case with possibly different values of d and k horizontally and vertically, but it is only the length of runs across the merging array which determines the width of the merging array.)

2.7.3 Offset periodic merging

Periodic merging arrays have a period of p. The finite state source for WXW therefore also has a period of p and thereby p phases. The construction may be generalized by offsetting the phase of say the right merging array. Define the W_ϕ boundary constrained array by WXW_ϕ, where W_ϕ is offset ϕ rows ($0 \leq \phi \leq 2d-1$) relative to W. A similar construction as before may be used. The entropy is still found by applying (2.9) to WXW_ϕ.

This approach may be generalized to use different periodic merging arrays, but this requires the use of multiple finite state sources.

2.7.4 Density constraints

We now use the method of cascading periodic arrays on the density constraints introduced in Section 2.1.1. Let rX denote the configuration given by row r followed by X having $N-1$ rows. For the density constraint, for any admissible configuration rX, the configuration Xr is also admissible. Of course, this no longer is true if X has more than $N-1$ rows. Therefore any admissible configuration within an $n=N$ by $m \geq M-1$ rectangle may be used as the period of a merging array.

In Section 2.7.7 we give results for the density constraint with parameters $(d_{min}, d_{max}) = (4,5)$ and $N = M = 3$.

As for the SRLL constraint merging arrays specified after X and Y could be used. It is not clear what the width of these merging arrays for the density constraint should be.

Example 2.26. For the $(d_{min}, d_{max}) = (4,5)$ and $N = M = 3$ density constraint, we can show that there are no solutions to the problem of finding a merging array Y of width w between any two arrays X and Z where $w \leq 7$. Consider the configuration:

$$\begin{array}{cccccccc}
\multicolumn{2}{c}{X} & & & Y & & \multicolumn{2}{c}{Z} \\
0 & 0 & y_{11} & y_{12} & \cdots & y_{17} & 1 & 1 \\
0 & 0 & y_{21} & y_{22} & \cdots & y_{27} & 1 & 1 \\
1 & 0 & y_{31} & y_{32} & \cdots & y_{37} & 1 & 0
\end{array}$$

Due to the constraint, the first column, (y_{11}, y_{21}, y_{31}), of Y must be all 1. Hence $y_{12}, y_{22}, y_{32}, y_{13}, y_{23}, y_{33}$ will contain at least four 0s and at most two 1s. Thus the fourth column has to contain at least two 1 elements, (y_{i4}). On the other hand, the seventh column of Y must be all 0. Hence columns five and six, (y_{15}, y_{25}, y_{35}) and (y_{16}, y_{26}, y_{36}), will contain at least four 1s and at most two 0s. Thus the fourth column has to contain at least two 0 elements, leading to a contradiction. This example gives an instance for which no merging array of width seven exists.

In a similar fashion one can give examples of particular arrays X and Z that do not have any merging arrays of width 6 or less. This is done in Appendix C.

2.7.5 Cyclic matrices

As mentioned the use of periodic merging arrays introduces a period p and phases in the finite state source. The states may be grouped in classes according to their phase. With a proper ordering this means that each transition is from a state in class i to one in class $i+1$ modulo p.

With a proper ordering of indices the transfer matrix $\mathbf{T}_m$ is block cyclic. This matrix has a unique maximum positive eigenvalue λ_m. The matrix after p transitions, $\mathbf{T}_m^p$, has a block diagonal form with p blocks namely a block for each phase. By Perron-Frobenius [43], λ_m^p is $p-$tuple eigenvalue for $\mathbf{T}_m^p$ and $\mathbf{T}_m$ has p simple eigenvalues $\lambda_m e^{ik2\pi/p}$, $k = 0, ..., p-1$.

2.7.6 Boundary constrained arrays

Consider the domino tiling constraint. Instead of defining a periodic merging array, the array X is just boundary constrained such that no (horizontal) piece crosses the boundary, i.e. the pieces form a tiling of the rectangle of width m. This way the arrays may be cascaded without an actual merging array. The array may be called a boundary constrained array. (This solution to domino tiling could also be viewed as a special case with $w = 0$.)

For the SRLL(d, k) constraint for $k \geq 2d$ (Etzion's Cases 2 and 3), it is possible to use boundary constrained arrays without (periodic) merging arrays. For $k \geq 2d$ define $r = k - 2d$ and $q \leq r$. An array X may be specified with the boundary constraint that the rightmost run is between d and $d + r - q$ and the leftmost run is between d and $d + q$. Any admissible pair of these arrays, X and Y, may be merged to the admissible array XY.

2.7.7 Numerical results

Table 2.12 presents lower bounds on the entropy of the SRLL(2,3) constraint using two methods: Etzion's merging arrays of width 12 and W-boundary constrained arrays where the periodic merging array has width $w = 4$ as given in Example 2.23.

As argued in Example 2.25, the W-boundary constrained arrays for SRLL(2,3) have 2 deterministic columns besides the columns of W. Therefore we have compared bounds obtained by arrays with the same number of non-deterministic columns, i.e. width m for the merging array and $m + 2$ for the periodic merging array, respectively.

For further comparison we have shown the number of transitions, sz$(\cdot)$ of the transfer matrices T and T' where the last is for the W-boundary constrained band.

Several things can be noted. $\frac{H_W(m)}{m+4}$ is not monotonic. The size of the transfer matrix T' is much smaller than that of T and only for $m = 13$ does the bound compare favorably for the W-constrained arrays.

The periodic merging array imposes constraints on the inner array which for some values of m are particular severe. Hence the drop in the number of configurations from $m = 10$ to $m = 11$.

The offset periodic merging of Section 2.7.3 was also applied using W_ϕ. The last column in Table 2.12 gives the highest entropy (denoted H_ϕ) among the offsets for each m. It may be noticed that the values for which H_W is not good one of the offsets produce better results. We conjecture that using the offset periodic merging arrays there for all m be a particular offset (depending on m) which would perform well.

For comparison, we give an upper bound of 0.257 for the entropy for SRLL$(2, 3)$. This was obtained using a band source of width 11 and the upper bound (2.5). It is clear that there is a large gap between the upper and lower bounds on the entropy obtained in this fashion.

m	$\mathrm{sz}(T_m)$	$\mathrm{sz}(T'_{m+2})$	$\frac{H(m)}{m+12}$	$\frac{H_B(m+2)}{m+6}$	$\frac{H_\phi(m+2)}{m+6}$
10	95464	18192	0.120	0.100	0.117
11	233952	10264	0.126	0.0923	0.131
12	562536	134996	0.131	0.116	0.123
13	1358632	1075216	0.135	0.136	0.136
14	3292760	2239364	0.139	0.137	0.137
15	7939920	1508272	0.143	0.124	0.138

Table 2.12: Lower bounds on the entropy of SRLL(2,3)

Table 2.13 gives the entropy using a specific periodic merging array for the density constraint for $(d_{min}, d_{max}) = (4, 5)$ and $N = M = 3$. The chosen merging array has period $p = 2$, and the period is given by 10 and 01 in every other row. The results provide a lower bound on the entropy of the constraint. An upper bound (2.5) on the entropy of 0.554 was obtained using an a band source of width $m = 11$.

The minimum width for which some merging array exists for any arbitrary pair of arrays X and Y from the density constraint is not known to us. Since, by Example 2.26, it is at least 8 we have also shown the lower bounds on the entropy using this value in Table 2.13.

It is seen that the periodic merging array approach provides significantly higher entropy than what can be hoped for using a merging array in this example.

m	$\frac{H(m)}{m+8}$	$\frac{H_W(m)}{m+2}$
7	0.269	0.353
8	0.280	0.360
9	0.289	0.372
10	0.297	0.382

Table 2.13: Lower bounds on the entropy of density(4,5)

2.7.8 Extension to 3D

In this section we give an outline of how the ideas of the previous sections can be extended to 3D in a straightforward manner.

In general a 3D constrained volume is characterized by a set of forbidden volumes or equivalently an admissible subset of the configurations within a N by M by L volume [37], [16].

The technique of periodic merging arrays may also be extended to 3-D. The 3-D information carrying array X may be described as a sequence of n by m arrays extending in the third direction. Periodic merging arrays of width w are repeated at a distance of $m + w$. Perpendicular to these, another set of periodic merging arrays of width v are repeated at a distance of $n + v$. (These merging arrays are infinite in the two other directions.) The two sets of merging arrays must have identical symbols on coinciding positions where they intersect.

We consider the simple case, where the period is w and v for the two sets of arrays, respectively. Further m is a multiple of w and n is a multiple of v. In this case the two sets of merging arrays form a periodic structure with a basic period of $m + w$ by $n + v$ in the $(m, n)-$plane.

The domino tiling may be generalized by packing the 3-D space with pieces of 1 by 1 by 2 each of which are parallel to one of the axis. A 3-D boundary constrained 3-D array is given by packing the sequence of n by m arrays with the 3-D domino pieces, thus no piece crosses the boundary and $w = v = 0$.

The (d, k) SRLL constraint is generalized by requiring that the 2D (d, k) SRLL is satisfied for any plane perpendicular to one of the three axis. The periodic merging arrays are specified by setting $w = v = 2d$. A period of $2d$ by $2d$ by $2d$ is introduced where any array perpendicular to an axis is identical to the period of the 2D $2d$ by $2d$ periodic merging array (or its bitwise negation). For m and n being multiples of $2d$, this period may be centered at the intersections of the merging arrays and thereafter repeated in each of the directions. Now a sequence of boundary constrained arrays of m by n may be specified.

2.8 Discussion

Another method of obtaining lower and upper bounds for general constrained fields by Markley and Paul appear in [35]. Their method is refined by Friedland in [18]. It would be interesting to implement them to compare them with the other bounds cited in this work.

We introduced the notion of splitting constraints as an example of some particularly simple constrained fields where one could hope that some of the fallacies of the general case were less severe. As an example consider the zero capacity problem. This appears to have been solved for a large class of splitting constraints by Kato and Zeger, namely the asymmetric RLL [28], [29]. One could then hope that this result could be extended to all splitting constraints.

Chapter 3

Stochastic models of constrained fields

We are interested in modeling the behavior of various coding schemes for constrained fields. In 1D the theory of Markov Sources and their entropy has been a powerful tool in ordinary constrained coding.

Hence, it seems natural to consider probability distributions when dealing with constrained fields. The obvious generalization of a Markov Source to 2D is a Markov Random Field (MRF). While MRFs have found widespread use in image processing [31] their use in coding applications have been rather limited. This stems from two drawbacks of the MRF. First of all, they are generally not causal and secondly, in order to calculate the entropy of a MRF one has to determine the partition function, which in all but the simplest cases is an intractable problem [5].

In this chapter we will investigate a particular simple MRF where we are able to calculate the entropy, namely the classical construction of the Pickard Random Field (PRF) [40], which is based on properties of the probability distribution on 2×2 elements.

The goal is to model the behavior of coding schemes. In our particular case we present a model of a bit-stuffer for the NIB constraint. We do this by considering a PRF where the conditional probabilities are induced by a bit-stuffer.

3.1 Measures on fields

Let F be the elements of an $n \times m$ rectangle. Let μ_F be a measure on $\mathcal{A}^F$ that agrees with the constraint. That is, of all the $|\mathcal{A}|^{n\times m}$ possible configurations those configurations that contains forbidden words have probability zero according to μ_F.

The (measure theoretic) entropy of μ_F is defined as

$$H(\mu_F) = -\frac{1}{nm} \sum_{x \in \mathcal{A}^{n\times m}} \mu_F(x) \log_2 \mu_F(x). \tag{3.1}$$

We are interested in measures that can be extended to a stationary measure on an arbitrary rectangle $n^* \times m^*$.

In general it appears to be difficult to find stationary probability distributions for 2D constrained fields.

3.1.1 Symmetric Pickard Fields

We will study a special kind of probability distribution on a field that is completely determined by the distribution on a 2×2 rectangle.

$$\begin{matrix} A & B \\ C & D \end{matrix}$$

where A, B, C, D are random variables over $\mathcal{A}$.

Definition 3.1. Let X, Y, Z be random variables over some alphabet $\mathcal{A}$. We say that X and Y are *conditionally independent given* Z, denoted $X \perp Y \mid Z$, if $\forall x, y, z \in \mathcal{A}$:

$$\Pr(X = x, Y = y | Z = z) = \Pr(X = x | Z = z) \Pr(Y = y | Z = z).$$

Definition 3.2. Let F be a stochastic field. If F satisfies the following conditions for any 2×2 lattice, we say that F is a *Pickard Field*.

1. $\Pr(A = x, B = y) = \Pr(A = x, C = y) = \Pr(A = y, B = x)$.
2. $B \perp C \mid A$.
3. The distribution on rows are identical, as is the distribution on columns.

The joint distribution of $(ABCD)$ is completely specified by the probability distribution (A) as well as the three conditional probability distributions $(B|A)$, $(C|A)$ and $(D|ABC)$. This is due to the following decomposition:

$$\Pr(ABCD) = \Pr(D|ABC)\Pr(ABC) \tag{3.2}$$

and due to the condition $B \perp C \mid A$

$$\Pr(ABC) = \Pr(BC|A)\Pr(A) = \Pr(B|A)\Pr(C|A)\Pr(A). \tag{3.3}$$

Example 3.3. This example appears in [15]. We will construct a Pickard field that satisfies the Hard Square constraint of no neighboring ones. Let (A) be determined by $\Pr(1) = 1/5$. By symmetry $\Pr(10) = \Pr(01)$ and due to the constraint $\Pr(0|1) = 1$. Hence $\Pr(10) = \Pr(1)\Pr(0|1) = 1/5$ and therefore $\Pr(00) = 3/5$. Since $\Pr(1|0) = \frac{1/5}{4/5} = 1/4$ we have determined $(B|A)$ and due to symmetry $(C|A)$. By the decomposition (3.3) we can determine (ABC). We only need to determine the conditional distribution $(D|ABC)$. Due to symmetry we have

$$\Pr(01,00) = \Pr(10,00) = \Pr(00,10) = \Pr(00,01)$$

and $\Pr(01,10) = \Pr(10,01)$. Calculations then give $\Pr(01,00) = 3/20$ and $\Pr(10,01) = 1/20$ and hence $\Pr(00,00) = 1 - 12/20 - 2/20 = 3/10$.

The entropy $H = H(D|ABC)$ is then $H = 0.5755$. By comparison Calkin and Wilf showed that the entropy of the Hard Square constrained field itself was approximately 0.5879 [6].

3.1.2 Extending the measure

Given the distribution on the 2×2 lattice, $(ABCD)$, we can extend this to a measure $\mu_{n\times m}$ on an $n \times m$ lattice $x = (x_{ij})$ in the following manner.

First the symbol x_{11} is drawn according to (A). Then the first row $x_{12} \dots x_{1m}$ is drawn according to the conditional distribution $(B|A)$ one symbol at a time. Then the first column $x_{21} \dots x_{n1}$ is drawn according to $(C|A)$ one symbol at a time. x_{22} can then be drawn using $(D|ABC)$.

Proceeding in this manner one has (using short hand notation for probabilities given by the argument):

$$\begin{aligned}\mu_{n\times m}(x) = &\Pr(x_{11})\\ &\cdot\Pi_{j=2}^{m}\Pr(x_{1j}|x_{1(j-1)})\\ &\cdot\Pi_{i=2}^{n}\Pr(x_{i1}|x_{(i-1)1})\\ &\cdot\Pi_{i=2}^{n}\Pi_{j=2}^{m}\Pr(x_{ij}|x_{(i-1)(j-1)},x_{(i-1)j}x_{i(j-1)}).\end{aligned} \tag{3.4}$$

The construction of the extended measure shows that a Pickard Field is causal and can be simulated in a straightforward manner.

The extended measure is *stationary* if the joint distribution of $(ABCD)$ does not depend on which 2×2 rectangle within the $n\times m$ rectangle we regard.

While the symmetry constraints of a Pickard field greatly facilitates computational ease, they also limits the modeling power. Both points were illustrated in Example 3.3 where the entropy of a Pickard field satisfying the Hard Square constraint was lower than the entropy of the Hard Square constrained field itself.

Therefore we loosen some of the requirements of symmetry. As before we have four stochastic variables A, B, C and D over some alphabet $\mathcal{A}$ and we consider the joint distribution of (A, B, C, D) on the 2×2 lattice

$$\begin{matrix} A & B \\ C & D \end{matrix}$$

Definition 3.4. If $B \perp C \mid A$, the joint distribution of $(ABCD)$ on the 2×2 lattice is a *Pickard model.*

Recall, that a constrained field is called higher order if the forbidden blocks cannot fit in a 2×2 lattice. Consistent with this terminology we say that a Pickard model is first order, since it is completely determined by the probabilities on a 2×2 lattice.

The Pickard model suffers from being a first order model. This means that we can not model higher order constraints such as the RLL$(2,\infty)$ constraint on the symbol level. However, in the next sections we will try to increase the modeling power of the Pickard model by using an extended alphabet.

3.2 Modeling higher order constraints

One way to model higher order constraints in the framework of the first order Pickard model is to increase the alphabet size. We use an extended alphabet of blocks of the same size built of symbols of the underlying alphabet.

In order for the Pickard model to capture the constraint, the blocks of the extended alphabet must be large enough to contain the size of the constraint in a 2×2 lattice. On the other hand we want to have the alphabet to be as small as possible, in order to keep the number of parameters of the model tractable. As a general rule of thumb, if $N \times M$ is the size of the constraint, then the blocks of the extended alphabet should be of size $n \times m$, where $n = \lceil N/2 \rceil$ and $m = \lceil M/2 \rceil$.

Example 3.5. Consider the RLL$(2, \infty)$ constraint. It has extent 3×3 so we will use blocks of size 2×2. The forbidden words horizontally and vertically are $\mathcal{F} = \{11, 101\}$. Therefore there is no need to include blocks where 11 occurs in the alphabet. This leads to the following alphabet

$$\mathcal{A} = \begin{Bmatrix} 00 & 01 & 10 & 00 & 00 & 01 & 10 \\ 00, & 00, & 00, & 10, & 01, & 10, & 01 \end{Bmatrix} = \{s, t, u, v, x, y, z\}.$$

Consider the four stochastic variables $A, B, C, D \in \mathcal{A}$ in the lattice

$$\begin{matrix} A & B \\ C & D \end{matrix}$$

The question is whether we can find probability distributions $(B|A)$, $C(|A)$ and $(D|ABC)$ that agrees with the constraint and such that $B \perp C \mid A$. First, we note that one cannot have that (BC) and A are independent, since

$$\Pr(A = y, B = u, C = u) = 0 \neq \Pr(A = y) \Pr(B = u, C = u).$$

We have shown how to choose an extended alphabet for the RLL$(2, \infty)$ constraint. In Section 3.2.1 we will show that we can find a Pickard model over this alphabet.

The Pickard model essentially says that there is no "diagonal interaction". This is also true for splitting constraints such as RLL(d, ∞). So in a sense, we get diagonal independence "for free" when considering splitting constraints.

Proposition 3.6. *Let F be a splitting constrained field. Assume that there exists conditional distributions $(B|A)$ (horizontally) and $(C|A)$ (vertically) that agrees with F. Then there exists a Pickard model $(ABCD)$ for F.*

Proof. Essentially, we have to show that we can define a probability distribution (ABC) such that $B \perp C \mid A$. We simply define

$$\mu_{(ABC)} = \Pr(A)\Pr(B|A)\Pr(C|A). \tag{3.5}$$

This is by construction diagonally independent, but the question is whether it is well-defined, i.e. whether the definition of $\mu_{(ABC)}$ agrees with the constraint. Assume for a moment, that a particular configuration abc is invalid (and we want this configuration to have probability zero by (3.5)). Since the constraint is splitting, the only forbidden words occur horizontally and vertically. That means that either ab or $\genfrac{}{}{0pt}{}{a}{c}$ is invalid (or both). If ab is invalid then $\Pr(B = b|A = a) = 0$ since we assume that $(B|A)$ agreed with the constraint. And then $\mu_{(ABC)}(abc) = 0$. Similarly if $\genfrac{}{}{0pt}{}{a}{c}$ is invalid we also have $\mu_{(ABC)}(abc) = 0$. So the constructed probability agrees with the constraint.

Note that we have not made a clear distinction between the symbol and block level in this proof due to the notational burden. However, it is hopefully clear that the construction is indeed valid because of the splitting of the constraint. □

3.2.1 A Pickard model for RLL$(2, \infty)$

We now return to Example 3.5 where the alphabet

$$\mathcal{A} = \{s, t, u, v, x, y, z\}$$

was introduced. We will show that several Pickard models for RLL$(2, \infty)$ exists over this alphabet.

We need to construct distributions $(B|A)$ and $(C|A)$ that agrees with the constraint. We will try to be as unbiased as possible. Hence we begin by determining admissible $|\mathcal{A}| \times |\mathcal{A}|$ matrices T_B and T_C, that are defined in the following manner:

$$(T_B)_{ab} = \begin{cases} 1 & \text{if } ab \text{ doesn't violate the RLL}(2,\infty) \text{ constraint.} \\ 0 & \text{otherwise.} \end{cases}$$

T_C is defined in a similar fashion except that the constraint is now checked vertically.

$$(T_C)_{ac} = \begin{cases} 1 & \text{if } \begin{smallmatrix} a \\ c \end{smallmatrix} \text{ doesn't violate the RLL}(2,\infty) \text{ constraint.} \\ 0 & \text{otherwise.} \end{cases}$$

By inspection one can see that

$$T_B = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } T_C = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Note that $T_B \neq T_C$.

We now distribute the probabilities evenly among the valid transitions. Define the row sum of a matrix A as $s_i^A = \sum_j A_{ij}$. Then we set

$$(Q_B)_{ij} = \frac{1}{s_i^B}(T_B)_{ij} \text{ and } (Q_C)_{ij} = \frac{1}{s_i^C}(T_C)_{ij}. \tag{3.6}$$

By construction both Q_B and Q_C are stochastic matrices. Furthermore, they can be thought of as an unbiased (with regards to the constraint) choice of the probabilities on the border.

We will construct the distribution of $(D|ABC)$ in a similar fashion. First we determine the admissible matrix T_D.

$$(T_D)_{(abc)d} = \begin{cases} 1 & \text{if } \begin{smallmatrix} a & b \\ c & d \end{smallmatrix} \text{ is valid.} \\ 0 & \text{otherwise.} \end{cases}$$

This is a $|\mathcal{A}^3| \times |\mathcal{A}|$ matrix. Since, in this case $|\mathcal{A}| = 7$ this matrix with 343 rows is a bit unwieldy. However, due to the nature of the constraint, things simplify. Since the constraint is splitting we can ensure that A and D are independent and that the distribution of $(D|ABC)$ only depends on (BC). We have

$$\begin{aligned} \Pr(ABCD) &= \Pr(D|ABC)\Pr(ABC) \\ &= \Pr(D|BC)\Pr(ABC) \end{aligned}$$

So we only need to specify the 49×7 matrix $\tilde{T}_D$. We will do this in an algorithmic fashion. Note that $(\tilde{T}_D)_{(bc)d} = 1$ by definition if both $\begin{smallmatrix} b \\ d \end{smallmatrix}$ and cd are valid configurations. But by definition of T_B and T_C this is the same as $(T_B)_{cd} = 1 = (T_C)_{bd}$. So we simply set

$$(\tilde{T}_D)_{(bc)d} = (T_B)_{cd} \cdot (T_C)_{bd}. \tag{3.7}$$

We then take as an unbiased distribution:

$$(Q_D)_{ij} = \frac{1}{s_i^D}(\tilde{T}_D)_{ij}.$$

Hence we have constructed a Pickard model for the higher order constraint by utilizing an alphabet extension.

3.2.2 The problem of the exploding number of parameters

Let $n = |\mathcal{A}|$ be the size of the extended alphabet. In order to specify a Pickard model over this alphabet, we need to specify the four distributions (A), $(B|A)$, $(C|A)$ and $(D|ABC)$. Consider first $(B|A)$. It has n^2 parameters of the form $q_{st} = \Pr(B = t|A = s)$. Since n of these are given by the condition that a probability vector should sum to 1, we are left with $n(n-1)$ parameters.

In a similar fashion one can see that in the general case the distribution of $(D|ABC)$ will have $n^3(n-1)$ parameters. Thus our Pickard model has $(n-1) + 2(n(n-1)) + n^3(n-1) = (n-1)(n^3+2n+1)$ parameters. This might be slightly overparametrized in the sense that some of these parameters will be zero due to the particular constraints of the field in question. The point still stands however, that the number of parameters grow fast with the size of the extended alphabet. In the example of the RLL$(2,\infty)$ constraint, there are 2148 parameters.

The unbiased construction in Section 3.2.1 could be considered canonical and thus one way with dealing with the large numbers of parameters. However, while our construction is canonical, we would like the resulting model to have additional properties. First of all, we would like to have a stationary model and secondly we would like to have a model with high entropy.

The large number of parameters in our Pickard model brings the hope that we indeed should be able to capture these two properties. On the other hand it raises the question of how to choose the parameters. In Section 3.3 we discuss when it is possible to find stationary Pickard models and in section 3.4 we will propose a method that in some cases can choose the parameters to ensure stationarity.

3.2.3 Limitations of the model

As long as the constrained field is splitting, we get diagonal independence "for free" as demonstrated by Proposition 3.6. However, this is not always the case. Consider the diamond constraint introduced in Section 2.3. We have

$$\mathcal{F}_\diamond = \left\{ 11, 101, \begin{matrix}01\\10\end{matrix}, \begin{matrix}10\\01\end{matrix}, \begin{matrix}1\\1\end{matrix}, \begin{matrix}1\\0\\1\end{matrix} \right\}.$$

Hence in the hope of using a Pickard model we introduce the following extended alphabet:

$$\mathcal{A} = \left\{ \begin{matrix}00\\00\end{matrix}, \begin{matrix}01\\00\end{matrix}, \begin{matrix}10\\00\end{matrix}, \begin{matrix}00\\10\end{matrix}, \begin{matrix}00\\01\end{matrix} \right\} = \{s, t, u, v, x\}. \tag{3.8}$$

Now consider the configurations sv and $\begin{smallmatrix}s\\t\end{smallmatrix}$. Both of these are valid diamond configurations. However the configuration

$$\begin{matrix} s & v \\ t & \end{matrix} = \begin{matrix} 00\,00 \\ 00\,10 \\ 01 \\ 00 \end{matrix}$$

violates the constraint. This means that B and C are not conditionally independent given A. In this case we cannot apply the Pickard model. Note that this problem can not be solved by extending the alphabet size further. No matter how large the extended alphabet is, B and C will be close to each other and possibly influence each other.

In general, this will be a potential problem when modeling any non-splitting constrained field. However, we will propose an extension to the model in Section 3.5 that at least from a theoretical point of view solves the problem.

3.3 A stationary model

In Section 3.1.2 we saw how we could extend the distribution on a 2×2 to an arbitrary lattice. We now turn to the question of how to ensure that this extension is stationary.

We say that the distribution $(ABCD)$ is *stable* if the distributions on (AB) and (CD) are identical and the distributions on columns (AC) and (BD) are identical.

The following Theorem due to Pickard [40] gives a sufficient condition on $(ABCD)$ for the extended measures (3.2-3.4) to be stationary.

Theorem 3.7. *Let $\mu_{2\times 2}$ be a stable measure induced by $(ABCD)$ satisfying $B \perp C \mid A$. If $B \perp C \mid D$ then the extended measure $\mu_{n\times m}$ based on (3.2-3.4) is Markovian and stationary for any $n, m \geq 2$.*

Theorem 3.7 provides sufficient conditions for the measure $\mu_{2\times 2}$ to be extended to a stationary measure. Since

$$\begin{aligned} &B \perp C \mid D \Leftrightarrow \\ \Pr(BCD) &= \Pr(D)\Pr(B|D)\Pr(C|D) \end{aligned} \tag{3.9}$$

Equation (3.9) expresses the independence conditions in a form which may be checked.

Further it follows that

Theorem 3.8. *The entropy per symbol of a stationary measure $\mu_{n\times m}$ defined by Theorem 3.7 and given by (3.1) is bounded by*

$$H(\mu_F) \geq H(D|ABC). \tag{3.10}$$

Proof. The $n\times m$ rectangle $x_{11}, \dots x_{nm}$, of the measure $\mu_{n\times m}$, is divided into the initial boundary $x_{i1}, 1 \leq i \leq n$ and $x_{1j}, 1 \leq j \leq m$ and the remaining interior part. The entropy of each element of the latter is given by $H(D|ABC)$ due to the stationarity and the chain rule. The entropy of the initial boundary is not less than this as the distribution on the initial boundary is given by (A) and the Markov chains $(B|A)$ and $(C|A)$, which are identical to the marginal distribution on the rows (and columns) of the remaining interior, due to the stationarity. □

Consider a row by row traversal of the rectangle. As argued in the proof, the entropy of the elements not belonging to the initial boundary given the causal elements is given by $H(D|ABC)$.

3.4 An iterative search for a stationary solution

As noted in Section 3.2.2 a Pickard model for a higher order constrained field has a large number of parameters. We would like to have a stationary model. While Theorem 3.7 gives sufficient conditions for a Pickard model to be stationary, it is not clear how to choose the parameters of the model to actually satisfy the conditions of the theorem. In this section, two iterative procedures are described as a method to determine the parameters for a stationary model.

The first procedure is part of the second and this combination provides a search for stationary solutions. The first procedure changes the boundary distributions $(B|A)$ and $(C|A)$ until $(ABCD)$ is stable. The second procedure extends this by changing $(D|ABC)$ until $B \perp C \mid D$.

We now introduce some notation in order to present the procedures. Let $|\mathcal{A}|$ denote the size of the extended alphabet. Let $Q_B = (q_{ij})$ denote the $|\mathcal{A}| \times |\mathcal{A}|$ transfer probability matrix for $(B|A)$, that is $q_{ij} = \Pr(B = j|A = i)$. Let π_A denote the stationary distribution for Q_B. We will use π_A as the distribution of (A). For symmetry reasons we set the boundary distributions $(B|A)$ and $(C|A)$ to be identical. This way $(B|A)$ and $(C|A)$ automatically have the same stationary distribution, π_A, which is used for the distribution (A). (This symmetry constraint need not be imposed as long as $(B|A)$ and $(C|A)$ have the same stationary distribution.)

Let $Q_D = (q_{ij})$ denote the $|\mathcal{A}|^3 \times |\mathcal{A}|$ transfer probability matrix for $(D|ABC)$. Let Q_{BD} and Q_{CD} denote the $|\mathcal{A}|^2 \times |\mathcal{A}|^2$ transfer probability matrices for the distributions $(BD|AC)$ and $(CD|AB)$, respectively.

Algorithm 3.9 iterates until the measure is stable, i.e. until (AB) and (CD) are identical and (AC) and (BD) are identical. Algorithm 3.10 iterates until (3.9) is also satisfied calling Algorithm 3.9 in each iteration.

In general, let P_X denote the probability vector of the marginal distribution X of the joint distribution $(ABCD)$. Let a superscript index the iterations, e.g. Q_B is initialized by $Q_B^{(0)}$ and after n iterations $Q_B^{(n)}$ is output. We set $Q_C^{(n)} = Q_B^{(n)}$ due to the symmetry assumption.

Algorithm 3.9. Assume that $Q_B^{(0)}$ and Q_D are given. Let ϵ denote

some prescribed tolerance of error.

1. Calculate $\pi_A^{(n)}$ from $Q_B^{(n-1)}$. Calculate $P_{ABCD}^{(n)}$ (3.2-3.3).
2. Calculate $P_{AB}, P_{CD}, P_{AC}, P_{BD}$ from $P_{ABCD}^{(n)}$.
3. If $\|P_{AB} - P_{CD}\| + \|P_{AC} - P_{BD}\| < \epsilon$ then goto 9.
4. Calculate $Q_{CD}^{(n)}$ and $Q_{BD}^{(n)}$ from $P_{ABCD}^{(n)}$.
5. Find the stationary distribution $\pi_{AB}^{(n)}$ for $Q_{CD}^{(n)}$ and $\pi_{AC}^{(n)}$ for $Q_{BD}^{(n)}$.
6. Let $\pi^{(n)}$ be the average of $\pi_{AB}^{(n)}$ and $\pi_{AC}^{(n)}$.
7. Calculate $Q_B^{(n)}$ from $P_{AB}^{(n)} = \pi^{(n)}$.
8. Goto 1.
9. Output $Q_B^{(n)}$ and $\pi_A^{(n+1)}$.

Step 3 performs the test for $(ABCD)$ being stable. In Step 7 we calculate the conditional distribution $(B|A)$ from the joint distribution (AB). By the imposed symmetry we force the conditional distribution $(C|A)$ to be the same as $(B|A)$.

We cannot state sufficient conditions for Algorithm 3.9 to converge. However, the following conditions appears to be beneficial in that regard.

- Q_B and Q_C are identical.
- Q_B and Q_C are irreducible.
- Q_D obeys the following symmetry constraint

$$\forall a, b, c \in \mathcal{A} : \Pr(D|abc) = \Pr(D|acb). \tag{3.11}$$

Now we extend the procedure to search for model parameters that satisfy the conditional independence $B \perp C|\ D$. The main condition is (3.9). We will write this in a slightly different form:

$$\sum_{x \in \mathcal{A}} \Pr(A = x, BCD) = \Pr(D) \Pr(B|D) \Pr(C|D). \tag{3.12}$$

Inspired by (3.12) we introduce a scale parameter for each context (bcd). Define

$$\lambda_{bcd} = \frac{\Pr(D = d)\Pr(B = b|D = d)\Pr(C = c|D = d)}{P_{BCD}(bcd)} \tag{3.13}$$

For each bcd configuration the two sides of (3.9) are calculated. Temporary parameters for the conditional probabilities of Q_D involved are modified by the scale parameter (3.13) to achieve equality for the given bcd configuration. This leads to the following algorithm.

Algorithm 3.10. Given Q_B and $Q_D^{(0)}$.

1. Run Algorithm 3.9 until convergence.
2. Calculate P_{BCD} and $P_t = P_D Q_{B|D} Q_{C|D}$
3. If $\|P_{BCD} - P_t\| < \epsilon$ Goto 8.
4. $\forall bcd \in \mathcal{A}^3 : \lambda_{bcd} = P_t / P_{BCD}(bcd)$.
5. $\forall abcd \in \mathcal{A}^4 : Q_D^{(n)}(d|abc) = \lambda_{bcd} Q_D^{(n-1)}(d|abc)$.
6. Normalize $Q_D^{(n)}$.
7. Goto 1.
8. Output $Q_D^{(n)}$,

In Step 2, the terms are calculated from P_{ABCD} (3.2-3.3). In Step 5, the update of Q_D is given by considering the terms contributing to the difference, $\|P_{BCD} - P_t\|$ in Step 3 (3.13). After one pass over the configurations, these parameters are normalized (Step 6) to define a new set of conditional probabilities $Q_D^{(n)}$ appropriately summing to 1. (A stop criteria may be included for the case that the algorithm does not converge within a given number of steps.)

The search given by Algorithm 3.10 calls Algorithm 3.9 in its iterations, proceeding until both have converged. Given a solution, according to Theorem 3.7, this output satisfying the constraints of Algorithms 3.9 and 3.10, and thus being stable as well as satisfying (3.9), can be extended to a stationary measure on any $n \times m$ rectangle.

Example 3.11. If we return to the RLL$(2,\infty)$ constraint, we immediately note that the boundary distributions $(B|A)$ and $(C|A)$ can never be identical, since $T_B \neq T_C$. As this appears to be a requirement for the convergence of Algorithm 3.10, it seems we cannot use the method for this particular constraint.

As we have seen a Pickard model is still possible for this constraint, but it is unresolved how to actually choose the parameters in order to obtain a stationary solution.

Despite Example 3.11 we remain confident that the presented algorithms will be useful for some fields with more inherent symmetry. In Section 3.6 we discuss one such case.

3.5 Fields on lattices

We now return to the problem illustrated by the diamond constraint in Section 3.2.3. We can solve this by considering configurations on rectangles of lattices rather than just horizontal-vertical rectangles. When dealing with an extended alphabet, there are several ways to tile the plane using the blocks of the extended alphabet. We are only interested in configurations where the blocks have the same orientation. Let $n \times m$ be the size of the blocks of the extended alphabet. Then consider the lattice defined by the basis vectors

$$e_1 = \begin{pmatrix} 1 \\ -(\max(m-1,1)) \end{pmatrix} \quad \text{and} \quad e_2 = \begin{pmatrix} 1 \\ \max(m-1,1) \end{pmatrix}. \tag{3.14}$$

As an example consider the configuration, on a 3×3 rectangle, over $\mathcal{A}$ in Fig. 3.1. Here the basis vectors are $e_1 = (1,-1)^T$ and $e_2 = (1,1)^T$.

Hence our Pickard model $(ABCD)$ would be arranged in the following 2×2 lattice

$$\begin{matrix} & B & \\ A & & D \\ & C & \end{matrix} \tag{3.15}$$

In the sequel when we refer to an $n \times m$ rectangle it will be oriented along the diagonal basis vectors of a rotated and scaled coordinate system as in Fig. 3.1, i.e. the entries of the configuration is labeled as (row, column), but here the “rows” and “columns” run along the diagonals.

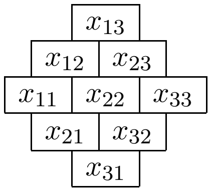


Figure 3.1: A 3×3 rectangle on the lattice defined by the basis vectors $e_1 = (1, -1)^T$ and $e_2 = (1, 1)^T$.

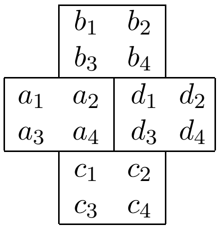
The definition of entropy holds in this setting as well. The measure theoretic entropy (3.1) has not changed whereas for the combinatorial entropy (2.1) one has to consider rectangles defined in the manner above. It is reasonable to question whether the combinatorial entropy defined on a lattice agrees with the normal definition.

3.5.1 Returning to the diamond constraint

Consider once more the Diamond constraint and the following alphabet

$$\mathcal{A} = \left\{ \begin{matrix} 00 & 01 & 10 & 00 & 00 \\ 00' & 00' & 00' & 10' & 01 \end{matrix} \right\} = \{s, t, u, v, x\}.$$

We now arrange our random variables $A = (a_1 a_2, a_3 a_4)$, $B = (b_1 b_2, b_3 b_4)$, $(c_1 c_2, c_3 c_4)$ and $D = (d_1 d_2, d_3 d_4)$ in the following lattice:



(3.16)

This corresponds to basis vectors $e_1 = (1, -2)^T$ and $e_2 = (1, 2)^T$. When considering whether $B = b$ can occur after $A = a$ we will inspect the configuration of ab but arranged according to (3.16). In general we write ab for "horizontal" concatenation, i.e. along the direction of the basis vector e_2 and $\substack{a\\c}$ for "vertical" concatenation, i.e. along the direction of the basis vector e_1.

Since the extent of the constraint is 3×3 we have made sure that B and C do not "interact" by this construction. Hence we have that $B \perp C \mid A$. Thus we are able to employ a Pickard model in this case.

Let us consider the admissible matrices T_B and T_C. We define

$$(T_B)_{ab} = \begin{cases} 1 & \text{if } ab \text{ doesn't violate the } \diamond(3) \text{ constraint.} \\ 0 & \text{otherwise.} \end{cases}$$

Similarly we have

$$(T_C)_{ac} = \begin{cases} 1 & \text{if } \substack{a\\c} \text{ doesn't violate the } \diamond(3) \text{ constraint.} \\ 0 & \text{otherwise.} \end{cases}$$

By inspection it is seen that

$$T_B = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix} \text{ and } T_C = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

We can then define the admissible matrix T_D for the distribution $(D|ABC)$ by

$$(T_D)_{(abc)d} = \begin{cases} 1 & \text{if } \substack{ab\\cd} \text{ is valid.} \\ 0 & \text{otherwise.} \end{cases}$$

In a similar fashion as we did for the RLL$(2, \infty)$ constraint in Section 3.2.1 we can then construct unbiased distributions $(B|A)$, $(C|A)$ and $(D|ABC)$.

Again we note that $T_B \neq T_C$ so it is unclear how to choose a stationary Pickard model as we cannot use the search of Section 3.4. We will not pursue a Pickard model for the Diamond constraint further. Instead, in the next section we utilize the lattice Pickard model as well as the iterative search method to completely specify a stationary model for the NIB constraint.

3.6 A stationary model for NIB

Until now the symbol blocks we have considered have been quadratic in size. Now, we give an example of some rectangular blocks which lead to a simple model for the NIB constraint. The model approximates the behavior of a bit-stuffer for NIB.

Consider the higher order alphabet, $\mathcal{A}$, over the binary alphabet given by

$$\mathcal{A} = \{00, 01, 10, 11\} .$$

The configurations considered will be on the lattice defined by the two basis vectors $e_1 = (1, -1)^T$ and $e_2 = (1, 1)^T$. In Fig. 3.2, an example of a 3×3 rectangle over $\mathcal{A}$ is shown. If we construct admissible matrices

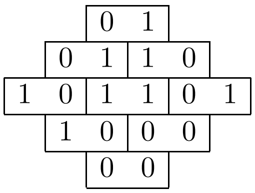


Figure 3.2: An example of a NIB configuration on a 3×3 field.

T_B and T_C we note that $T_B = \mathbf{1} = T_C$, i.e. all transitions are valid according to the constraint. Furthermore the two matrices are identical, so we can utilize Algorithm 3.10.

First we initialize $(B|A)$ (and by symmetry $(C|A)$ to be an unbiased distribution, $Q_B = (q_{ij})$ with $q_{ij} = 1/16$ for all $i, j \in \mathcal{A}$. We then use the bit-stuffer to induce the conditional probabilities Q_D. This is described in Section 3.6.1. The resulting stationary model from applying Algorithm 3.10 is then further modified based on a search for higher entropy, with the method described in Section 3.6.3. Finally we shall present the optimized model in Section 3.6.4.

3.6.1 Using a bit-stuffer to induce $(D|ABC)$

One can use a bit-stuffer to define the conditional distribution $(D|ABC)$. The bit-stuffer works on the binary symbol level in the following manner.

$$\begin{array}{cccc} & b_1 & b_2 & \\ a_1 & a_2 & \multicolumn{2}{|c|}{d_1 \quad d_2} \\ & c_1 & c_2 & \end{array}$$

Whether bits are stuffed or written as is to positions d_1 and d_2 is decided solely on the basis of the context a_1, a_2, b_1, b_2 and c_1, c_2 in order not to violate the NIB constraint. Based on the bit-stuffer one can induce the probabilities $(D|ABC)$ at the block level.

Example 3.12. Consider the following context $(a = 01, b = 00, c = 00)$:

$$\begin{array}{cccc} & 0 & 0 & \\ 0 & 1 & \multicolumn{2}{|c|}{1 \quad f} \\ & 0 & 0 & \end{array}$$

This results in a stuffed '1' in position d_1, whereas position d_2 is 'free', denoted f.

Hence

$$\Pr(D = 00|A = 01, B = 00, C = 00) = 0$$

and

$$\Pr(D = 01|A = 01, B = 00, C = 00) = 0$$

since the bit-stuffer will stuff a '1' in position d_1, whereas

$$\Pr(D = 10|A = 01, B = 00, C = 00) = 1/2$$

and

$$\Pr(D = 11|A = 01, B = 00, C = 00) = 1/2$$

since position d_2 is free and an unbiased input stream was assumed. We refer to this as the unbiased bit-stuffer.

In a similar fashion one can determine the probabilities $(D|abc)$ for every context (abc).

As an alternative a biased bit-stuffer may be applied. We consider introducing a bias for d_1 given abc, such that the distribution on the admissible values of $d = d_1, d_2$ is uniform. (This is already the case for the biased bit-stuffer when 2 or 4 values of d are admissible for a given context abc. The only difference is for contexts abc where three values are admissible. In this case the conditional probability for d_1 is biased with probabilities 1/3 and 2/3. The following decision for d_2 is either forced by the constraint or unbiased.)

It may be noted that the bit-stuffing induced probabilities $(D|ABC)$ are symmetric in the following fashion. For any $a, b, c \in \mathcal{A}$ we have $\Pr(D|abc) = \Pr(D|acb)$.

3.6.2 Numerical results

Applying Algorithm 3.10 starting with the model with conditional probabilities induced by unbiased bit-stuffing, we obtained a stationary solution in accordance with Theorem 3.7. The entropy (3.10) is expressed by $H(D|ABC) = 0.9037$ per binary symbol. This is less than the lower bound of 0.9127 presented as a lower bound for bit-stuffing in [20]. The latter lower bound was based on using both an unbiased and a biased bit-stream. Thereafter the biased bit-stuffer above was used to initialized the conditional probabilities Q_D. (The biased stream has the same probabilities as in [20], but is only applied for one (d_1) of the two elements of the block.) This gave a stationary solution with an entropy of $H(D|ABC) = 0.9073$ per binary symbol.

3.6.3 Searching for a higher entropy

Now taking our point of departure in a model derived, the parameters of this model may be perturbed slightly in order to search for a model with higher entropy.

To keep things simple we insist that the boundary distributions should still be identical. Furthermore the conditional distributions $(D|ABC)$ should have the same symmetry as the bit-stuffing induced probabilities, that is $\Pr(D|abc) = \Pr(D|acb)$.

Let $\delta > 0$. For each context (abc) there are three probability parameters (of which 0, 1 or 2 may be 0)

$$\begin{aligned} p_{abc} &= \Pr(D = 00|abc) \\ q_{abc} &= \Pr(D = 01|abc) \\ r_{abc} &= \Pr(D = 10|abc) \end{aligned}$$

For each context (abc) and for each parameter $p \in \{p_{abc}, q_{abc}, r_{abc}\}$, we perturb the parameter $p = p + \delta$, ensure that conditional probabilities $\Pr(D|abc)$ still sum to 1 and then iterate the model towards stationarity according to the method described in Section 3.4.

Experimenting with different δ, we noted that not all parameterizations of the model iterated to a stationary solution satisfying the sufficient condition, $B \perp \ C|\ D$, for the Markov property. In Section 3.6.4, the parameters of the model with the highest entropy obtained in this fashion are presented. It should be noted that only a limited search has been carried out, from the two initial distributions, to demonstrate that increasing the entropy by a simple search is possible.

3.6.4 Parameters of the NIB model

In this section we present the parameters for a stationary Pickard model of the NIB constraint that has an entropy of $H(D|ABC) = 0.9082$.

We obtained this by applying Algorithm 3.10 to the distribution induced by the biased bit-stuffed as explained in Example 3.12. The resulting stationary model was then modified by the search method described in Section 3.6.3 in an effort to increase the entropy. The parameters of the stationary model with the highest entropy found in this manner is presented in Tables 3.1, 3.2 and 3.3, respectively.

Note that due to the symmetry constraints imposed on the probabilities, Table 3.3 only contains 40 of the 64 different contexts (abc). (The remaining are given by the symmetry.)

For comparison we carried out a search for higher entropy based on the stationary model obtained for the unbiased bit-stuffer. This yielded a stationary model with an entropy $H(D|ABC) = 0.9041$ per binary symbol.

A	00	01	10	11
$\Pr(A)$	0.2699	0.2309	0.2321	0.2671

Table 3.1: A model for NIB. Distribution of (A).

	B			
A	00	01	10	11
00	0.2351	0.2367	0.2224	0.3058
01	0.2574	0.1962	0.2735	0.2729
10	0.2796	0.2718	0.1919	0.2566
11	0.3073	0.2194	0.2409	0.2325

Table 3.2: A model for NIB. Boundary distribution $(B|A)$.

3.7 Discussion

In this chapter we applied the first order model of a PRF to various higher order constraints. The induced measures of the bit-stuffing scheme for the Hard Square constraint [41] can be seen as a variation of a PRF. While the entropy was optimized for this particular constraint, the constraint itself is first order.

In [27] some higher order models for higher order constraints are investigated. The models can be seen as a generalization of the Pickard field.

Our iterative method for finding a stationary solution works well for the NIB constraint. However, it is not clear under which general conditions it converges. While symmetry seems beneficial, it is not always easy to assure as was seen from the RLL$(2, \infty)$ example.

Our use of Algorithm 3.9 can be seen as a special case of the following general problem: Given a coding method (in this case bit-stuffing) how does the distribution on the border influence the long term behavior of the encoder? Our experiments show that it is possible to initialize a bit-stuffer for the NIB constraint such that the induced measure is stable. However, we want more, namely that the measure is stationary. While our presented model is that, we have modified the original distribution of the bit-stuffer in order to have that $B \perp C \mid D$. While this ensures

	D			
ABC	x	y	z	v
xxx	0.2928	0.3470	0	0.3602
xxy	0.2759	0.2475	0.2174	0.2591
xxz	0.2985	0.3416	0	0.3599
xxv	0.2705	0.2274	0.2332	0.2689
xyy	0.2616	0.2754	0.2483	0.2146
xyz	0.2705	0.2343	0.2295	0.2658
xyv	0.2440	0.3219	0.2364	0.1977
xzz	0.2718	0.3536	0	0.3746
xzv	0.2659	0.2157	0.2445	0.2740
xvv	0.2557	0.2757	0.2590	0.2095
yxx	0	0	0.5333	0.4667
yxy	0	0	0.4563	0.5437
yxz	0.2032	0.2324	0.3195	0.2449
yxv	0.2705	0.2274	0.2332	0.2689
yyy	0	0	0.5364	0.4636
yyz	0.2705	0.2343	0.2295	0.2658
yyv	0.3598	0	0.3486	0.2915
yzz	0.3008	0.2141	0.2506	0.2345
yzv	0.2659	0.2157	0.2445	0.2740
yvv	0.3531	0	0.3576	0.2893
zxx	0.2928	0.3470	0	0.3602
zxy	0.2759	0.2475	0.2175	0.2592
zxz	0.2985	0.3416	0	0.3599
zxv	0.2705	0.2274	0.2332	0.2689
zyy	0.2616	0.2754	0.2483	0.2146
zyz	0.2705	0.2343	0.2295	0.2658
zyv	0.2440	0.3219	0.2364	0.1977
zzz	0.4346	0.5654	0	0
zzv	0.5521	0.4479	0	0
zvv	0.4812	0.5188	0	0
vxx	0.2074	0.2458	0.2916	0.2552
vxy	0.2759	0.2475	0.2175	0.2591
vxz	0.2032	0.2324	0.3195	0.2449
vxv	0.2705	0.2274	0.2332	0.2689
vyy	0.3611	0	0.3427	0.2962
vyz	0.2705	0.2343	0.2295	0.2658
vyv	0.3598	0	0.3486	0.2915
vzz	0.1978	0.2573	0.2723	0.2726
vzv	0.2659	0.2157	0.2445	0.2740
vvv	0.3531	0	0.3576	0.2893

Table 3.3: A model for NIB. The conditional probabilities $P(D|ABC)$. x denotes 00; y, 01; z, 10 and v, 11, respectively.

stationarity, we no longer deal with the original bit-stuffer, but only what one could see as an approximate model of its behavior.

One could consider the opposite problem. Given the border distribution how do we choose the conditional probabilities $(D|ABC)$ such that the field is stationary and (preferably) have a high entropy? It turns out that the method of iterative scaling [7] can solve this problem, indeed in a fashion that yields the maximum entropy under the given constraints.

Chapter 4

Bit-stuffing for checkerboard constrained fields

In this chapter we present a variation of the bit-stuffing scheme introduced in Section 2.4.2. The method is applicable to all checkerboard constrained fields. We investigate probability measures induced by the bit-stuffing scheme. We show how to calculate the entropy of the measures, thus obtaining a lower bound on the entropy of the constraints considered.

4.1 Introduction

We have already discussed how to use a bit-stuffing scheme to encode a data stream into an $n \times m$ array W such that some arbitrary, but fixed checkerboard constraint is obeyed. Recall, that we were not able to calculate the average coding rate of the scheme. Indeed, this appears to be a difficult problem. In [41], [20] a detailed analysis of the bit-stuffing scheme for RLL(d, ∞) constraints is carried out and the authors offer several lower bounds on the average code rate of the scheme.

We will now modify the bit-stuffing scheme slightly. Instead of writing one row at a time, left to right, we will access the positions in a different order. This enables us to calculate the average code rate of the new scheme. More precisely, we model the behavior of the bit-stuffing encoder as a Markov band source and show how to extend this to a measure on the quarter plane. We then find the entropy of the extended measure.

We will first describe the modified bit-stuffer for a single RLL(d, ∞) array. We then introduce the Markov band source model of the scheme in Section 4.2. Then in Section 4.3 we show how to extend the bit-stuffing to the quarter plane. We then introduce measures modeling the bit-stuffer and calculates their entropy. Finally, in Section 4.4 we present the construction in a general setting applicable to all checkerboard constraints.

4.1.1 A modified bit-stuffing scheme

Let W denote an $n \times m$ array. We think of W as having two borders X and Z each of width b and an interior Y of width $m - 2b$ such that $W = XYZ$. The width of the borders has to be at least $M - 1$. That is, in the case of RLL(d, ∞) we have $b = d$.

The modification to the bit-stuffer of Section 2.4.2 is simply this: We still write to W one row at a time, but first we write to the border X from left to right, then to Z from left to right, and finally to Y, left to right. Zeros as stuffed as necessary in the same manner as before.

Example 4.1. Consider a modified bit-stuffer for the RLL$(2, \infty)$ constraint. Assume that the array W has width $m = 7$. In this case the borders have width $b = 2$. A '0' is simply written as a '0', but we stuff zeros around a '1' and then skip to the next available position. This is depicted below, where x is some previously written symbol (in this case a zero) and y is the next available position. Note that y is in Z, because we write in the order X, Z and then Y. Hence the last position in Y will only be written after we have filled the two positions of Z.

	X			Y		Z	
$1 \mapsto$	x	1	0	0		y	
		0					
		0					

We show a more involved example. Again, the y denotes the next available position.

$$\begin{array}{r|c|c|c|c|c|c|c|}
\multicolumn{1}{r}{} & \multicolumn{2}{c}{X} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{Y} & \multicolumn{1}{c}{} & \multicolumn{2}{c}{Z} \\
\cline{2-8}
 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
\cline{2-8}
0110011 \mapsto & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\cline{2-8}
 & 1 & 0 & 0 & & & 0 & y \\
\cline{2-8}
 & 0 & & 0 & & & & \\
\cline{2-8}
\end{array}$$

We will model the behavior of the modified bit-stuffing scheme as the output from a finite state band source. We first introduce the necessary notation, then describe how to model the bit-stuffer.

4.1.2 Markov band sources

We now return to the finite state band sources introduced in Section 2.6. Let B be a finite state source with states of height $N-1$ and width m. Let $T = (t_{ij})$ denote the transfer matrix of the source. We introduce transition probabilities $\mathbf{P}_m = (p_{ij})$ for the finite state source. We will only consider transition probabilities satisfying that $p_{ij} > 0$ if and only if $t_{ij} > 0$. We will refer to a band source with transition probabilities as a Markov Band Source (MBS).

Definition 4.2. Let B be a MBS with transition probabilities $\mathbf{P}_m$. Let π be the stationary distribution of $\mathbf{P}_m$. We define the entropy of B as

$$H(B) = \sum_i \sum_j \pi_i p_{ij} \log_2(1/p_{ij}). \tag{4.1}$$

One may wonder whether the definition is useful since it relies on the existence of a stationary distribution for the transition probabilities. However, as the following lemma shows, as long as we only deal with checkerboard constraints, as is the focus of the present chapter, it is well-defined.

Lemma 4.3. For a MBS B defined over a checkerboard constraint the entropy is well-defined and given by (4.1).

Proof. If a stochastic matrix is irreducible, then the stationary distribution exists and is unique [5]. Since the stochastic matrix for B is defined such that $p_{ij} > 0$ if and only if $t_{ij} > 0$, it is sufficient to show that the transfer matrix T is irreducible.

Let s and t be two states corresponding to any two valid configurations. For a checkerboard constraint the all zero state $\mathbf{0}$ is always a valid configuration. Consider the configuration obtained by stacking s, $\mathbf{0}$, and t on top of each other. Since this is a valid configuration, it is possible to go from state s to state t in $2N-2$ transitions. Hence the transfer matrix is irreducible. In fact, let $k = 2N-2$. Then the matrix T^k is strictly positive, i.e. the transfer matrix is *primitive.* $\square$

Finally, we remark that the combinatorial entropy of the band is given by the max-entropic solution based on the transfer matrix. Using the notation of Section 2.6 we thus have $H(B) \leq H_B$, as originally shown by Shannon [44].

4.2 A MBS model of the modified bit-stuffer

Let W be some valid RLL(d,∞) $n \times m$ array generated by the modified bit-stuffing scheme and assume that $m \geq 2d+1$. We will show how to consider the array as being generated by a MBS, $\mathbf{W}$, one row at a time.

Recall, that the extent of RLL(d,∞) is $(d+1)\times(d+1)$ so the the states of the MBS are $d \times m$. The process $\mathbf{W}$ has the marginal processes at the borders, $\mathbf{X}$ and $\mathbf{Z}$, having a width $b = M-1 = d$. We denote the m new elements generated by the transition from one state, i, to the next state, j, by

$$r = (r_0 r_1 \dots r_{m-1}) = (x_0, ..., x_{d-1}, y_0, ..., y_{m-2d-1}, z_0, ..., z_{d-1}).$$

In Figure 4.1 we have depicted the output of a MBS for the RLL$(2,\infty)$ constraint in this manner.

It remains to show how to determine the transition probabilities (p_{ij}) of the MBS such that they mimic the bit-stuffer.

Let R_l denote a binary stochastic variable defined on the element in column l of the new row, r. In order not to violate the constraint we have to consider the conditional distribution of R given the context of the current symbol. We therefore introduce a series of stochastic variables $C(l)$, $l = 1, \dots, m$. We let (the size of) the context depend on the column since we have to take into account that the bit-stuffer writes

X	**Y**	**Z**
$\mathbf{x}_0$	$\mathbf{y}_0$	$\mathbf{z}_0$
$\vdots$	$\vdots$	$\vdots$
$\mathbf{x}_i$	$\mathbf{y}_i$	$\mathbf{z}_i$
x_0x_1	$y_0, y_1, \ldots, y_{m-5}$	z_0z_1

Figure 4.1: Output from a MBS for the RLL$(2,\infty)$ constraint defined by the modified bit-stuffing scheme. The source starts in state $\mathbf{w}_0 = \mathbf{x}_0\mathbf{y}_0\mathbf{z}_0$. The states have height 2. At the transition from state $\mathbf{w}_i$ to state $\mathbf{w}_{i+1}$ the row $x_0x_1y_0y_1\ldots y_{m-5}z_0z_1$ is output.

the bits in a different order. Below we have written the symbols of the new row r in the causal order

$$\tilde{r} = (x_0, ..., x_{d-1}, z_0, ..., z_{d-1}, y_0, ..., y_{m-2d-1}). \tag{4.2}$$

Furthermore from the contexts we can deduce whether zeros are stuffed or not.

In general the transition probabilities p_{ij} of course depends on the elements of the state i and the elements of the state j. The idea of the context variable is to determine the minimal set of elements necessary to calculate the probability.

Whether it is possible to write a 1 in a given position at the time of writing is only dependent on the d previous elements in the same column and the previous elements of the current row after reordering, i.e. preceding the current element in (4.2). Let $s = (s_1 \ldots s_d)^T$ be a state consisting of d rows corresponding to the RLL(d,∞) constraint. Then the contexts $C(l)$ of r_l for $l = 0, \ldots, m-1$ are depicted below.

1. $l = 0, \ldots, d-1$. This case covers the columns in X.

$$\begin{array}{ll} & s_1(l) \\ & \vdots \\ & s_d(l) \\ r_0 \ldots r_{l-1} & \end{array}$$

2. $l = d, \ldots, m-2d$. This case covers the columns in the left part of Y, that is at least d elements away from the right border. Z.

$$\begin{array}{ll} & s_1(l) \\ & \vdots \\ & s_d(l) \\ r_{l-d} \ldots r_{l-1} & \end{array}$$

3. $l = m-2d+1, \ldots, m-d-1$. This case covers the columns in the right part of Y, where we have to take Z into account. If l is say k positions from the start of Z we have to incorporate $d-k+1$ of the elements of Z into the context.

$$\begin{array}{lll} & s_1(l) & \\ & \vdots & \\ & s_d(l) & \\ r_{l-d} \ldots r_{l-1} & & r_{m-d} \ldots r_{m-k+1} \end{array}$$

4. $l = m-d, \ldots, m-1$. This case covers the columns of Z which has to have the same type of contexts as X.

$$\begin{array}{ll} & s_1(l) \\ & \vdots \\ & s_d(l) \\ r_{m-d} \ldots r_{l-1} & \end{array}$$

The transition probabilities of $\mathbf{P}_m$ is then determined by the product of the conditional probabilities, $p(R_l = r_l | C(l) = c(l))$,

$$p_{ij} = \prod_{l=0}^{m-1} p(R_l = r_l | C(l) = c(l)), \tag{4.3}$$

where the conditional probabilities are given by

$$\Pr(R_l = 1|C(l) = c(l)) = \begin{cases} p_1(l) & \text{if a 1 is admissible in position } l. \\ 0 & \text{if a 1 is not admissible in position } l. \end{cases} \tag{4.4}$$

Here $p_1(l)$ denotes the probability of a '1' in the input stream to column l. We consider the general case where we have several biased streams, possibly one for each column. We refer to the vector $p_1 = (p_1(0), \dots, p_1(m-1))$ as the bit-stuffing probabilities. The simple unbiased bit-stuffer would correspond to the case where $p_1(l) = 1/2$ for all l. Note that to avoid pathological cases we assume that $0 < p_1(l) < 1$ for all l.

We have defined the transition probabilities of the MBS, but it might not be entirely clear that they, in fact, are transition probabilities. We show this now.

Lemma 4.4. If the bit-stuffing probabilities p_1 satisfy $0 < p_1(l) < 1$ for all $l = 0, \dots m-1$, then the probabilities defined by (4.3-4.4) constitutes a stochastic matrix, $\mathbf{P}_m$, with $p_{ij} > 0$ if and only if $t_{ij} > 0$.

Proof. The first step is to show that $p_{ij} = 0 \Leftrightarrow t_{ij} = 0$. Assume $p_{ij} = 0$ for some transition with $t_{ij} > 0$. Since $p_1(l) < 1$ we have $\Pr(R_l = 0|C(l) = c(l)) > 0$ for all l. Therefore, $\Pr(R_l = 1|C(l_0) = c(l_0)) = 0$ for some specific l_0. Hence a '1' in position l_0 is not admissible given the context $c(l_0)$. Thus the configuration, corresponding to the transition from state i to j, is not admissible. Hence $t_{ij} = 0$. Conversely, if $t_{ij} = 0$ there is some position l_0 where a '1'is not admissible but actually occur and hence $p_{ij} = 0$.

The second step is to show that (4.3-4.4) forms a stochastic matrix, i.e. $\sum_j p_{ij} = 1$ for all i. Consider any given i. The new elements generated by the transition to j are considered one element at a time. All admissible configurations may be represented by a complete binary tree as follows. Each time a decision may be made, i.e. $\Pr(R_l = 1|C(l) = c(l)) > 0$, two branches with two new nodes are created and for each new node the bit-stuffing scheme is continued. Each node may be assigned a probability namely the probability given by the product of probabilities in the path of the node. As the sum of the branching probabilities always sums to one, so will the sum over the leaves of the tree given the sum over all admissible configurations as determined by the bit-stuffing scheme. □

Example 4.5. Consider again $F = \mathrm{RLL}(2, \infty)$ and let $m = 7$. Consider the following three states of a MBS for a modified bit-stuffer for F.

s	t_1	t_2
0000010	0001000	0001000
0001000	0000001	1000001

Note that there is a transition from state s to both state t_1 and state t_2. We will show how to calculate the probability of outputting the rows 0000001 and 1000001 respectively, given that we are in state s. Or equivalently, calculate the transition probabilities p_{st_1} and p_{st_2}.

For simplicity, let us assume that the bit-stuffer has two biased streams. One feeding the borders X and Z and one feeding the interior. Let p_X denote the possibility of a '1' in the stream for the borders and let p_Y denote the possibility of a '1' in the second biased stream. That is, the bit-stuffing probabilities are $p_1 = (p_X, p_X, p_Y, p_Y, p_Y, p_X, p_X)$.

The transition probabilities are defined by (4.3) and (4.4). We have

$$\begin{aligned} p_{st_1} &= \prod_{l=0}^{6} \Pr(R_l = r_l | C(l) = c(l)) \\ &= (1 - p_X)(1 - p_X)(1 - p_Y) \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot p_X. \end{aligned}$$

The first zero of the row isn't stuffed, since there are no '1' in its context. Hence $\Pr(R_0 = 0|c(0)) = 1 - p_X$, that is the probability of a '0 in the input stream feeding the X part. The same applies to the second and third zero, but for the latter we use the probability of a '0 in the stream feeding the Y. Notice, however, that $\Pr(R_3 = 0|c(3)) = 1$ since the zero is stuffed due to the '1' above it. We also have $\Pr(R_4 = 0|c(4)) = 1$, but this time the zero is stuffed because of the '1' in the last part of the row. This illustrates the order that the bit-stuffer operates in, i.e. Z is written to before Y and is thus captured by the way the transition probabilities is defined.

In a similar fashion we can compute p_{st_2}. Here we note that all of the zeros are stuffed. Hence

$$p_{st_2} = p_X \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot p_X.$$

4.3 Cascading bit-stuffing

We have described our MBS model of the modified bit-stuffing scheme for a single array W of $\mathrm{RLL}(d, \infty)$. Now we show how to extend the bit-stuffing scheme in the horizontal direction. We will cascade bands in a fashion similar to the method of Section 2.7. Consider arrays $W_0, \ldots W_{K-1}$ of the same size $n \times m$. As before $W_i = X_i Y_i Z_i$, but we insist that the arrays overlap consecutively, such that $Z_i = X_{i+1}$. We denote the concatenated or cascaded array as

$$W_0^{K-1} = X_0 \{Y_j Z_j\}_{j=0}^{K-1} = X_0 Y_0 Z_0 Y_1 Z_1 \cdots Y_{K-1} Z_{K-1}, \tag{4.5}$$

The bit-stuffer writes the first row of W_0. Then the first row of W_1, but here $X_1 = Z_0$ has already been written, so the bit-stuffer writes Z_1 and then Y_1 and then proceeds to write Z_2 and Y_2 and so on concluding with the first row of W_{K-1}. Then the second row of W_0 is written and so on.

It is the average coding rate of this cascading bit-stuffer we wish to compute. We now show how to model the bit-stuffer with probability measures. The idea is that the conditional entropy of the configurations on (Y_i, Z_i) conditioned on the configuration Z_{i-1} determines the entropy of the extended array.

4.3.1 The basic measures

Defining probability measures for constrained coding in general is a challenge. We now impose some restrictions that will facilitate analysis of our model of the bit-stuffer.

Let W be some $n \times m$ array over $\mathrm{RLL}(d, \infty)$. We assume that $W = XYZ$ with X, Y and Z defined as previously. Let μ_W be a probability measure on W. We consider the restriction that the (marginal) measures on the borders X and Z are identical, i.e.

$$\mu_X \equiv \mu_Z. \tag{4.6}$$

Furthermore we insist that the borders are independent. Let $\mathbf{X}$ and $\mathbf{Z}$ denote stochastic variables on X and Z, respectively. Then the requirement is that for all configurations $\mathbf{x}, \mathbf{z} \in E(n, b)$,

$$\Pr(\mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z}) = \mu_X(\mathbf{x}) \mu_Z(\mathbf{z}). \tag{4.7}$$

In order for the marginal measures to be identical we have to use the same bit-stuffing probabilities for Z and X. That is $p_1(l) = p_1(m - d + l)$ for $l = 0, \dots d - 1$. Furthermore, in order for the borders to be independent we have to let $m \geq 3d$. This will ensure that the width of Y is at least d and since the extent of the constraint is $d + 1 \times d + 1$ we can write to Z without regarding X. Indeed, considering the cascading (4.5) where each new Z_i may be seen as X_{i+1} in relation to Y_{i+1}, we have that the bit-stuffing of all Z_i are independent of all Y_i at the time the elements are assigned a value.

We sum up our observations so far with regards to extending the bit-stuffing from W to a larger rectangle. Let the height of the rectangle be $n + N - 1$, such that n refers to the number of transitions in a MBS generating the rectangle. For a given d and m, let $B_{n,K}$ denote the extension of W to the array W_0^{K-1} as defined by (4.5) consisting of $(n + N - 1)$ rows by $(K(m - d) + d)$ columns.

Definition 4.6. Given m, $p_1(l), 0 \leq l < m - d$ and the first d rows of the rectangle $B_{n,K}$, the elements of $B_{n,K}$ are addressed row by row. The *modified bit-stuffing scheme* on $B_{n,K}$ for the 2D RLL(d, ∞) constraint is defined by for each row addressing the elements of W_i in order of increasing i and within each W_i in the order given by $\tilde{r}$ (4.2). The same bit-stuffing probabilities (4.4) are applied for the elements within each W_i, i.e.

$$p_1(l) = p_1(l \ \text{ modulo}(m - d)), \quad 0 \leq l < K(m - d) + d. \tag{4.8}$$

Note that the behavior of the modified bit-stuffing scheme on a single band W_i can be modeled by a MBS as detailed in Section 4.2. Combining Lemma 4.4 and Lemma 4.3 gives that the entropy of this MBS is well-defined and given by (4.1).

We will now proceed to construct a measure on $B(n, K)$ where we can exploit the independence on the borders of the individual W_i.

4.3.2 Bit-stuffing induced measures on the quarter plane

Given a MBS $\mathbf{W}$ based on bit-stuffing we can construct a series of measures in a natural fashion that extends towards being defined on the quarter plane. In this section we will describe the construction in detail.

Let $\mathbf{w}_0^{n,K-1}$ denote a configuration on $B_{n,K}$, composed as follows. First consider the border process $\mathbf{X}$ of $\mathbf{W}$. The transition matrix for $\mathbf{X}$ has a stationary distribution π_X by Lemmas 4.3 and 4.4. Draw the initial $(N-1)\times d$ element state $\mathbf{x}_0$ from π_X and generate n rows using $\mathbf{X}$. This is the $(n+N-1)\times d$ left hand vertical boundary $B_n^{(v)}$ of $B_{n,K}$. Let $\mu^v(\mathbf{x}_1,\ldots\mathbf{x}_n|\mathbf{x}_0)$ denote the conditional measure on $B_n^{(v)}$ conditioned on the initial state $\mathbf{x}_0$.

Given the stationary distribution, π_W, of the process $\mathbf{W}$, consider a configuration $\mathbf{w}=\mathbf{x},\mathbf{y},\mathbf{z}$ and let $\pi_{\mathbf{y},\mathbf{z}|\mathbf{x}}$ be the conditional distribution $\pi_{\mathbf{y},\mathbf{z}|\mathbf{x}}=\frac{\pi_W(\mathbf{x},\mathbf{y},\mathbf{z})}{\pi_X(\mathbf{x})}$. Draw $(\mathbf{y}_0,\mathbf{z}_0)$ using this measure conditioned on the initial state $\mathbf{x}_0$ and continue to draw $(\mathbf{y}_i,\mathbf{z}_i)$ conditioned on $\mathbf{z}_{i-1}$, $i=1,\ldots,K-1$. This constitutes the $(N-1)\times K(m-d)+d$ upper horizontal boundary, $B_K^{(h)}$, of $B_{n,K}$. Let $\mu^h(\mathbf{y}_0,\mathbf{z}_0,\ldots,\mathbf{y}_{K-1},\mathbf{z}_{K-1}|\mathbf{x}_0)$ denote the conditional measure on $B_K^{(h)}$ conditioned on the initial state $\mathbf{x}_0$. We denote the interior set of elements by $B^*(n,K)$, i.e. $B^*(n,K)=B_{n,K}\backslash(B_n^{(v)}\cup B_K^{(h)})$. This is depicted in Figure 4.2.

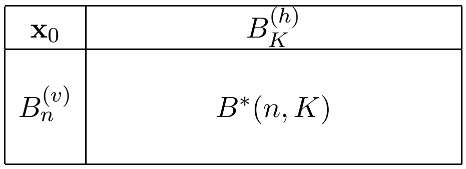


Figure 4.2: A high level view of the rectangle $B(n,K)$ consisting of borders and interior $B^*(n,K)$. Compare with Figure 4.3.

Considering the last row of a set of elements $(\mathbf{y},\mathbf{z})$, i.e. the elements of a state on Y and Z, as a single symbol of an extended alphabet, the interior $B^*_{n,K}$ may be viewed as an $n\times K$ rectangle over this alphabet. Out of the elements $(\mathbf{y},\mathbf{z})$ of a state $\mathbf{w}_j$ the transition outputs the elements $(y_0,...,y_{m-2d-1},z_0,...,z_{d-1})$ and the other elements of $(\mathbf{y},\mathbf{z})$ overlaps with the elements of the predecessor state.

For a given transition from state $\mathbf{w}_i$ to $\mathbf{w}_j$ with the combined configuration s_1^{d+1}, let $pre(\mathbf{w}_j)$ denote the predecessor state, $\mathbf{w}_i=s_1^d$, of the state $\mathbf{w}_j=s_2^{d+1}$.

Let t, $1\le t\le nK$, denote the index of $(\mathbf{y},\mathbf{z})_t$. Let $(\cdot,\mathbf{z})_t$ denote the

$\mathbf{z}$ part of the symbol $(\mathbf{y},\mathbf{z})_t$ with the following exceptions: We define $(\cdot,\mathbf{z})_{iK} = \mathbf{x}_{i+1}$ for $i = 0,\ldots,n-1$ corresponding to the first column which is the left border of the interior.

Let $P((\mathbf{y},\mathbf{z})_t|(\cdot,\mathbf{z})_{t-1}, pre(\mathbf{w}_t))$ denote the probability of $(\mathbf{y},\mathbf{z})_t$ conditioned on the causal part of the transition of the process $\mathbf{W}_i$ it is part of. This may be seen as conditioning the new elements of $(\mathbf{y},\mathbf{z})_t$ conditioned on the causal elements of the bit-stuffing. By drawing the symbol $(\mathbf{y},\mathbf{z})_{t+1}$ conditioned on $((\cdot,\mathbf{z})_t, pre(\mathbf{w}_{t+1}))$ using the conditional probability $P((\mathbf{y},\mathbf{z})_t|(\cdot,\mathbf{z})_{t-1}, pre(\mathbf{w}_t))$ one can then fill out the interior $B^*(n,K)$ one row at a time given the boundaries $B_n^{(v)}$ and $B_K^{(h)}$. This is illustrated in Figure 4.3.

Hence a measure $\mu_{B_{n,K}}$ on $B_{n,K}$ is defined by

$$\begin{aligned}\mu_{B_{n,K}}(\mathbf{w}_0^{n,K-1}) = {} & \pi_X(\mathbf{x}_0)\\ & \cdot \mu^v(\mathbf{x}_1,\ldots,\mathbf{x}_n|\mathbf{x}_0)\\ & \cdot \mu^h(\mathbf{y}_0,\mathbf{z}_0,\ldots,\mathbf{y}_{K-1},\mathbf{z}_{K-1}|\mathbf{x}_0)\\ & \cdot \prod_{t=1}^{nK} P((\mathbf{y},\mathbf{z})_t|(\cdot,\mathbf{z})_{t-1}, pre(\mathbf{w}_t)),\end{aligned} \tag{4.9}$$

where t is the index traversing the interior, row by row.

$\mathbf{x}_0$	$\mathbf{y}_0$	$\mathbf{z}_0$	$\mathbf{y}_1$	$\mathbf{z}_1$	$\ldots$	$\mathbf{y}_{K-2}$	$\mathbf{z}_{K-2}$	$\mathbf{y}_{K-1}$	$\mathbf{z}_{K-1}$
$\mathbf{x}_1$	$(\mathbf{y},\mathbf{z})_1$				$\ldots$			$(\mathbf{y},\mathbf{z})_K$	
$\mathbf{x}_2$	$(\mathbf{y},\mathbf{z})_{K+1}$				$\ldots$			$(\mathbf{y},\mathbf{z})_{2K}$	
$\vdots$	$\vdots$				$\vdots$			$\vdots$	
$\mathbf{x}_n$	$(\mathbf{y},\mathbf{z})_{(n-1)K+1}$				$\ldots$			$(\mathbf{y},\mathbf{z})_{nK}$	

Figure 4.3: The rectangle $B_{n,K}$ on which the measure $\mu_{B_{n,K}}$ is defined.

We have shown how to construct a measure $\mu_{B_{n,K}}$ that models the modified bit-stuffing scheme. We now state how to actually compute the entropy based on the MBS description.

Theorem 4.7. *Consider a modified bit-stuffing scheme on the rectangle $B(n, K)$ as given by Definition 4.6. Let $\mu_{B_{n,K}}$ be the induced measure as defined by (4.9). Then based on $\mu_{B_{n,K}}$ the per symbol entropy of the interior $B^*_{n,K}$ given the boundary of $B_{n,K}$ is given by*

$$H_{mb}(d, \infty) = \frac{H(\mathbf{W}) - H(\mathbf{X})}{m - d} \tag{4.10}$$

where $\mathbf{W}$ is the MBS induced by the bit-stuffer and $\mathbf{X}$ its border process.

Proof. Consider the MBS $\mathbf{W}$ defined by the modified bit-stuffing of W. We have already seen how the reordering of the bit-stuffer and setting $m \geq 3d$ ensures that the border processes $\mathbf{X}$ and $\mathbf{Z}$ of the MBS are independent.

By Lemma 4.3 and Lemma 4.4, the processes $\mathbf{X}$ and $\mathbf{Z}$ are well-defined and the stationary distributions π_X and π_Z of their transition probabilities exist. Since we bit-stuff in the same order (left-to-right) and use the same bit-stuffing probabilities for both $\mathbf{X}$ and $\mathbf{Z}$ when defining the transition probabilities p_{ij} (4.3) we can conclude that the stationary distributions are the same, i.e. $\pi_X = \pi_Z$. Initializing the boundary according to these identical stationary distributions, ensures $\mu_X = \mu_Z$ (4.6).

By construction of the MBS and the cascading bit-stuffer each element of X_i and Z_i is written prior to any element in Y_i which coincides with the 0-neighborhood of the elements of X_i and Z_i. Therefore any given $\mathbf{W}_i$ is independent of all $\mathbf{X}_j, j \notin \{i, i+1\}$ and the bit-stuffing scheme defines a MBS $\mathbf{W}_i$.

The same bit-stuffing probabilities are applied for all $\mathbf{W}_i$. Thus the transfer matrix $\mathbf{P}_m$ is identical for all $\mathbf{W}_i$. By construction of the measure $\mu_{B_{n,K}}$ we can choose that the initialization of the boundary is done by the "right" stationary distributions, ensuring that the border process $\mathbf{X}$ is in the stationary regime and that all $\mathbf{X}_i$ have identical measures, $\mu_{X_i} = \mu_X,\ \ 0 \leq i \leq K-1$. Initializing the boundary based on the stationary solution π_W for all $\mathbf{W}_i$, gives $\mu_{W_i} = \mu_W$ for all $i \leq K-1$.

Consider the elements $(\mathbf{y}, \mathbf{z})_t$, on Y_i as one symbol as in (4.9). Due to the independence of the $\mathbf{X}_i$s the conditional probability of an element $(\mathbf{y}, \mathbf{z})_t$ is given by the process $\mathbf{W}_i$ it belongs to. The conditional probability is given by one transition of $\mathbf{W}_i$ where all prior elements coinciding with a 0-neighbor element of an element in $(\mathbf{y}, \mathbf{z})_t$ are part of the predecessor state or one of the new elements in $\mathbf{X}_i$ or $\mathbf{Z}_i$ of the transition.

For notational consistency, let us write $\mathbf{w}_t$ as $(\cdot,\mathbf{z})_{t-1},(\mathbf{y},\mathbf{z})_t$ when representing it in a decomposed form. The conditional probability derived from one transition within some $\mathbf{W}_i$ is

$$\begin{aligned} P(\mathbf{w}_t|pre(\mathbf{w}_t)) &= P((\cdot,\mathbf{z})_{t-1}|pre(\mathbf{w}_t))\cdot P((\mathbf{y},\mathbf{z})_t|((\cdot,\mathbf{z})_{t-1},pre(\mathbf{w}_t)) \\ &= P((\cdot,\mathbf{z})_{t-1}|pre((\cdot,\mathbf{z})_{t-1}))\cdot P((\mathbf{y},\mathbf{z})_t|(\cdot,\mathbf{z})_{t-1},pre(\mathbf{w}_t)) \end{aligned}$$

as $(\cdot,\mathbf{z})_{t-1}$ only depends on $pre((\cdot,\mathbf{z})_{t-1})$ of $pre(\mathbf{w}_t)$.

We can write this as

$$P((\mathbf{y},\mathbf{z})_t|(\cdot,\mathbf{z})_{t-1},pre(\mathbf{w}_t)) = \frac{P(\mathbf{w}_t|pre(\mathbf{w}_t))}{P((\cdot,\mathbf{z})_{t-1}|pre((\cdot,\mathbf{z})_{t-1}))} \tag{4.11}$$

The important thing to note is that the terms are given by stationarity by $\mu_{W_i}=\mu_W$, independent of the value of the index t within the interior (for each W_i).

Therefore calculating the conditional entropy H of the new elements, $(\mathbf{y},\mathbf{z})_t$, of one transition gives,

$$H((\mathbf{y},\mathbf{z})_t|(\cdot,\mathbf{z})_{t-1},pre(\mathbf{w}_t)) = H(\mathbf{W})-H(\mathbf{X}).$$

Dividing by the number of elements, $(m-d)$, of $(\mathbf{y},\mathbf{z})_t$, gives (4.10).

□

Considering the combined symbols $(\mathbf{y},\mathbf{z})$ the measure (4.9) on the interior provides a stationary description. Viewing the individual elements in $(\mathbf{y},\mathbf{z}_t)$ the measure may be seen as quasi-stationary.

4.3.3 Some practical remarks

A drawback of bit-stuffing is the fact that it is a variable length code and has unlimited error propagation. However, the use of independent borders, in modified bit-stuffing, limits error propagation in the horizontal direction.

Writing data row by row introduces a latency of $m-d-1$ elements if $z_0,...z_{d-1}$ is written before y_i. This latency may be reduced to d if the writing of zs and ys are interleaved.

For RLL(d, ∞), the plane may be written column by column, using the biased sequence designated to the column. Thus the choice of biased sequence is only changed once for each new column. The column with z_i must be written before the column with y_{m-3d+i} in this case.

4.4 Bit-stuffing for checkerboard constraints

In the previous section, we investigated bit-stuffing for RLL(d, ∞) constraints. Since bit-stuffing including the modified bit-stuffing scheme is applicable for all checkerboard constraints, we will now offer a more general presentation of the modified bit-stuffing. The main difference is that for a transition, the elements of Y_i may be offset a few rows back compared to the elements of X_i and Z_i.

Example 4.8. Consider modifying bit-stuffing for the $\diamond(M)$ constraint. The width of the borders is chosen as $b = M - 1$. To maintain independence of the borders of the process $\mathbf{W}$, the elements of a new row of X and Z must be written prior to a new row of Y which is at least $M - 1$ rows back. This is obtained by writing the new elements of X and Z before the new elements of Y in one transition and letting these new elements of Y be $M - 2$ rows behind the row with new elements of X and Z. This is to ensure that the 1-norm distance between the last element, x_{b-1}, of the new row of X and the first element of the old row of Y is M such that no element of Y in the old row will influence the new elements in X. For an example we refer to Figure 4.4, where the construction is shown for the $\diamond(3)$ constraint. Note that the lag of Y relative to X and Z is $M - 2 = 1$ in this case.

Besides this modification, the bit-stuffing scheme and the calculation of $\mathbf{P}_m$ may proceed as for the 2D RLL(d, ∞) constraint. The transition probabilities, (p_{ij}), are defined by a product of conditional probabilities (4.3) derived from the bit-stuffing probabilities, $p_1(l)$, using (4.4). But of course, the context $C(l)$ has to be defined in an appropriate manner. A large portion of the present section is devoted to this and how to define the states of the MBS in a proper manner.

In the general case let the extent of the checkerboard constraint be $N \times M$. We shall introduce the modified bit-stuffing and the corresponding MBS jointly. Note that this presentation follows [13] very closely. Let

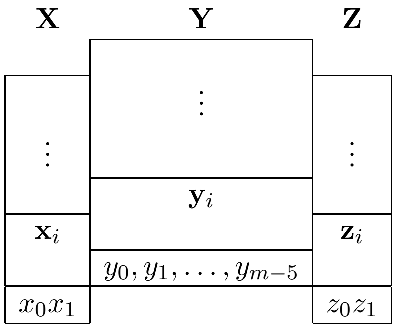


Figure 4.4: The modified bit-stuffing scheme for the $\diamond(3)$ constraint. The state $\mathbf{w}_i$ consists of the three blocks $\mathbf{x}_i, \mathbf{y}_i$ and $\mathbf{z}_i$, each having a height of two rows. On the transition to state $\mathbf{w}_j$ the new elements $x_0, x_1, z_0, z_1, y_0, \ldots, y_{m-5}$ are output.

b be the width of X and Z. Let $x_0, \ldots, x_{b-1}, z_0, \ldots, z_{b-1}, y_0, \ldots, y_{m-2b-1}$ be the order of the new elements of state $\mathbf{w}_j$ in the transition from the predecessor state $\mathbf{w}_i$. The new elements of X and Z belong to the same row, whereas the new elements of Y may be positioned a few rows behind. Let S_b specify the number of rows the new elements of Y are behind, e.g. for the $\diamond(M)$ constraint we have $S_b = M - 2$. Let S_t denote the number of rows the elements of a state of $\mathbf{W}$ are defined on. We will now give conditions on the set of elements defining the states of the Markov process $\mathbf{W}$.

4.4.1 The minimal set of elements for a state

We state the following conditions for the set of elements defining the states of $\mathbf{W}$ given by the modified bit-stuffing for checkerboard constraints.

C.1 $m \geq 2b + M - 1 \wedge b \geq M - 1$.

C.2 Positioning the 1 of the 0-neighborhood $\mathcal{N}$ at each position of $(x_0, ..., x_{b-1})$ and $(z_0, ..., z_{b-1})$, no 0 of $\mathcal{N}$ may coincide with a causal element of Y, but all causal elements of X and Z, respectively, are part of the state.

C.3 Positioning the 1 of the 0-neighborhood $\mathcal{N}$ at each position of $(y_0, ..., y_{m-2b-1})$, the state transition includes all elements coinciding with a 0 of $\mathcal{N}$, which are either an element of X or Z or a causal element of Y.

C.4 When considering a transition, in each column the new element and the elements of the old state of the column must form a contiguous set.

Thus the conditions above specifies a minimum set of elements, which must be included in the state of $\mathbf{W}$ for a given checkerboard constraint and requirements for the elements of one transition. Theorem 4.7 is formulated for the minimum set of elements for 2D RLL(d, ∞).

For any checkerboard constraint of extent (N, M), the state specified below will satisfy the conditions. Let the width of the borders X and Z be $b = M - 1$ and the width, m, of W be a value which satisfies C.1. In order to avoid any influence of elements of Y (C.2) on the writing of elements on X (and Z), $(x_0, \ldots, x_{b-1})$ of $\mathbf{X}$ (and $(z_0, \ldots, z_{b-1})$ of $\mathbf{Z}$) are written $N - 1$ rows ahead of writing $(y_0, \ldots, y_{m-2b-1})$ on Y. The state of $\mathbf{W}$ is given by the union of the following elements:

U_i The $N - 1$ elements above each of the elements $(y_0, \ldots, y_{m-2b-1})$ due to the last part of C.3.

U_b The $2N-1$ elements above each of the elements $(x_0, \ldots, x_{b-1})$ and $(z_0, \ldots, z_{b-1})$. These are included to ensure C.2 and the first part of C.3.

Based on $\mathbf{W}$ satisfying C.1-4, the bit-stuffing is extended to the set of elements, $D_{n,K}$ defined by the extension from W to W_0^{K-1} (4.5). Assume the initial state of each $\mathbf{W}_i$ is given. The configuration on the segment $D_{n,K}$ is determined by the set of initial states and n transitions of each $\mathbf{W}_i$ on W_0^{K-1}. The segment $D_{n,K}$ has width $K(m - b) + b$. The difference compared to the array $B_{n,K}$ is that the height may vary according to whether the column belongs to X_i or Y_i. (Columns of Z_i have the same height as columns of X_i.) The height in column l is given by n plus the height of the state in column l. Given the initial states the bit-stuffing schemes addresses the elements of $D_{n,K}$ in the order given by one transition of each $\mathbf{W}_i$ in order of increasing i before proceeding to the next sequence of transitions. Within each transition the elements of Z_i is addressed prior to the elements of Y_i.

This modified bit-stuffing scheme defines the transition probabilities p_{ij} (4.3) based on the conditional probabilities, $p(R_l = r_l|C(l) = c(l))$. When a 1 is admissible given context $c(l)$ the conditional probabilities must satisfy $0 < p(R_l = r_l|C(l) = c(l)) < 1$.

The four conditions, C.1-4, above defines restrictions on the states of $\mathbf{W}$. These are supplemented by two conditions on the contexts of the conditional probabilities.

C.5. The contexts $c(l)$ must include the causal 0-neighborhood elements of the given $\mathbf{W}_i$ at position l.

When the context elements of $c(l)$ are exactly given by the causal 0-neighborhood elements and the conditional probabilities by $p_1(l)$ (4.4), the bit-stuffing is called a *a modified bit-stuffing scheme for a checkerboard constraint.* The modified bit-stuffing for 2D RLL(d, ∞) is an example of this.

The contexts may also be extended and include more elements than just those defining whether $r_l = 1$ is admissible. In this case we call it a context-based bit-stuffing scheme.

C.6. For $\mathbf{X}$ (and $\mathbf{Z}$) the contexts are mappings of causal elements of $\mathbf{X}$ (and $\mathbf{Z}$) within the states of the corresponding transition. For the elements of Y the contexts are mappings of the causal elements of the transition, i.e. elements of the old state and the new elements of $\mathbf{X}$ and $\mathbf{Z}$ and prior new elements of Y. The context mappings for column l in $\mathbf{X}$ and column $m-b+l$ in $\mathbf{Z}$, $0 \leq l < b$ are identical and the same set of conditional probabilities are applied. This is to ensure the independence (4.6).

4.4.2 A context based bit-stuffing scheme

For a given checkerboard constraint, consider a Markov process $\mathbf{W}$, of width m, having states satisfying C.1-4 and transition probabilities derived from a set of conditional probabilities satisfying C.5-6. Assume the values of the elements of the initial states of all $\mathbf{W}_i$ on W_0^{K-1} are given. The *context based bit-stuffing scheme* on $D_{n,K}$ based on $\mathbf{W}$ is defined by a set of conditional probabilities $p(R_l = 1|C(l') = c(l))$, $l' = l$ modulo$(m-b)$ satisfying C.5-6. The order in which the elements of $D_{n,K}$ are addressed is given by a transition of each $\mathbf{W}_i$ in order of increasing i. Within each transition the order is given by the reordering (4.2).

The independence of the borders (4.7) is ensured by the offset of the new row of $\mathbf{Y}$ and $m \geq 2b + M - 1$. The conditions C.1-2 ensures that $\mathbf{Y}_{i-1}$ and $\mathbf{Y}_i$ are independent given the output of $\mathbf{X}_i$.

The conditional probabilities of the context-based bit-stuffing scheme for checkerboard constraints defines the probability matrix $\mathbf{P}_m$ of $\mathbf{W}$. The stationary distribution is given by $\pi_W \mathbf{P}_m = \pi_W$. The construction of $\mu_{B_{n,K}}$ (4.9) for the 2D RLL(d, ∞) constraint is generalized to a measure $\mu_{D_{n,K}}$ on $D_{n,K}$, which in the same manner is based on $\mathbf{W}$ and the transition matrix $\mathbf{P}_m$. The stationary distribution π_W is used to initialize the initial states of all $\mathbf{W}_i$ on W_0^{K-1} defining a measure on the upper horizontal boundary. The measure on the left vertical boundary is defined by $\mathbf{X}_0$ and the stationary distribution π_X. The probabilities on the interior, $D^*_{n,K}$, is given by the conditional probabilities of $(\mathbf{y}, \mathbf{z}_t)$ as for $B_{n,K}$.

The context based bit-stuffing thus defines a sequence of (2D) probability measures $\mu_{D_{n,K}}$ indexed by n and K. By construction, the measure $\mu_{D_{n,K}}$ is obtained as the marginal measure on $D_{n,K}$ of any $\mu_{D_{n'',K''}}$, where $n \leq n''$ and $K \leq K''$. Thus the measure is nested. Let $H(\mu_{D_{n,K}})$ denote the entropy of the measure $\mu_{D_{n,K}}$. The nesting property allows us to take the limit [41].

Let n' and m' denote the size of the sides of the bounding box of the elements of $D_{n,K}$. Set the elements of the bounding box, which are not in $D_{n,K}$, equal to 0.

Definition 4.9. The entropy, $H_C(m, b)$ of the context based bit-stuffing for a checkerboard constraint is defined as

$$H_C(m, b) = \lim_{n,K \to \infty} \frac{H(\mu_{D_{n,K}})}{n'm'}, \tag{4.12}$$

where the sides of the bounding box $n', m' \to \infty$ as $n, K \to \infty$.

Theorem 4.10. *Consider a given checkerboard constraint of extent $N \times M$). Assume that the MBS $\mathbf{W}$ has states and conditional probabilities satisfying the conditions, C.1-6 with $b \geq M - 1$ and $m \geq 2b + M - 1$. A context based bit-stuffing scheme for the checkerboard constraint based on $\mathbf{W}$ has the entropy*

$$H_C(m,b) = \frac{H(\mathbf{W}) - H(\mathbf{X})}{m-b}, \tag{4.13}$$

where $H(\mathbf{W})$ and $H(\mathbf{X})$ are the entropies per row (4.1) of the processes $\mathbf{W}$ and $\mathbf{X}$.

Proof. The first step of the proof follows the proof of Theorem 4.7 for a given n and K. $\mathbf{W}$, $\mathbf{X}$ and $\mathbf{Z}$ are finite state Markov processes. Again by Lemmas 4.3 and 4.4, $\mathbf{W}$ and the borders $\mathbf{X}$ and $\mathbf{Z}$ are all well-defined and have stationary solutions. The conditions C.1-3 maintains the independence of border processes, $\mathbf{X}$ and $\mathbf{Z}$. Using the same context mappings and conditional probabilities in the bit-stuffing scheme for all $\mathbf{W}_i$ gives $\mu_{W_i} = \mu_W$.

Consider the output of transition j of each process $\mathbf{W}_i$, $0 \leq i < K$, which overlap such that the configuration on X_i is given by the output of $\mathbf{Z}_{i-1}$. The elements output by this set of transitions, corresponds to one row of elements on $B_{n,K}$ in Theorem 4.7, but now possibly with the new elements of X_i and Z_i in one row and the new elements of Y_i in another row. The interior $D^*_{n,K}$ of $D_{n,K}$ is given by the output of these transitions for $1 \leq j \leq n$, where the elements of the first b columns, i.e. the left vertical boundary, are given by $\mathbf{X}_0$. The distribution on the boundary, i.e. the initial states of $\mathbf{W}_i$, $0 \leq i < K$ and the b first columns, is initialized based on the stationary distribution, π_W. Thus the boundary is initialized based on the stationary distributions in the same manner as for $\mu_{B_{n,K}}$. As in Theorem 4.7, the stochastic variables $\mathbf{Y}_{i-1}$ and $\mathbf{Y}_i$ are independent given the output of $\mathbf{X}_i$, as $b \geq M-1$ and the contexts of $\mathbf{X}_i$ do not have any elements of Y_{i-1} and Y_i. Likewise $\mathbf{W}_i$ is independent of all $\mathbf{X}_j, j \notin \{i, i+1\}$. By definition of the bit-stuffing scheme, $\mathbf{P}_m$ is identical for all $\mathbf{W}_i$. This leads to identical stationary distributions on the initial states, which in turn gives $\mu_{W_i} = \mu_W, 0 \leq i < K$.

The arguments of Theorem 4.7 still hold under these generalizations for each transition of p_{ij}. The probability of the elements $(\mathbf{y},\mathbf{z})_t$ conditioned on the causal elements is given by the identical and stationary processes $\mathbf{W}_i$. Therefore the contribution to the entropy for each $(\mathbf{y},\mathbf{z})_t$ conditioned on the causal elements is as in (4.11) given by

$$H(\mathbf{W}) - H(\mathbf{X}), \quad m \geq 2b + M - 1. \tag{4.14}$$

In the second part of the proof we will show that asymptotically the expression (4.14) determines the entropy. Let n^* and $m^* = K(m-b)$ denote the sides of the largest rectangle defined on the interior, $D^*_{n,K}$. The entropy $H(\mu_{D_{n,K}})$ relative to the size of the bounding box with sides n' and m' is bounded by

$$\begin{aligned} \frac{H(\mathbf{W}) - H(\mathbf{X})}{m-b}\frac{n^*m^*}{n'm'} &\leq \frac{H(\mu_{D_{n,K}})}{m-b}\frac{n^*m^*}{n'm'} \\ &\leq \frac{H(\mathbf{W}) - H(\mathbf{X})}{m-b}\frac{n^*m^*}{n'm'} \\ &+ \frac{n'b + m'(S_t+S_b)}{n'm'}|A| \end{aligned} \tag{4.15}$$

where S_t is the height of the states, S_b is the number of rows which X extends below Y, and $|A|$ is the size of the alphabet.

As $m^* = m' - b$ and $n^* = n' - S_t - S_b$ and since m, b, S_t, S_b and $|A|$ are all fixed values, asymptotically both the lower and upper bound in (4.15) converge to

$$\frac{H(\mathbf{W}) - H(\mathbf{X})}{m-b} \text{ for } n, K \to \infty.$$

□

Theorem 4.10 also applies to the modified bit-stuffing scheme as this is a special case of the context based bit-stuffing for checkerboards constraints. Actually the assumption of initialization based on the stationary distribution, π_W is not necessary as each $\mathbf{W}_i$ will converge to the stationary solution for $n \to \infty$.

4.4.3 Optimizing the entropy

The use of biased sequences in the modified bit-stuffing scheme is completely predefined in the sense that which biased stream to use is decided a priori based on the column number l.

For context based bit-stuffing, an increased number of biased sequences may be used. The decision of which biased sequence and thereby the conditional probability to be used may depend on a larger context.

Example 4.11. For the $\diamond(3)$ constraint, we applied context based bit-stuffing choosing the bit-stuffing probabilities p_1 conditioned on the values of the elements in the previous state. The states of $\mathbf{W}$ were defined as depicted on Fig. 4.4. The next row of the processes $\mathbf{X}$ (and $\mathbf{Z}$) was specified according to probabilities conditioned on the two previous rows. These processes were chosen such that they were symmetric in the two columns. The elements of the new row of Y was then specified according to probabilities conditioned on all 3 rows of the transition of $\mathbf{X}$ and $\mathbf{Z}$ combined with the 2 rows of the predecessor state on Y. These conditional probabilities for the new elements $(y_0, \dots, y_{m-2b-1})$ on Y were obtained from the transition probabilities of the maxentropic solution [15] for $\mathbf{W}$ derived from the transfer matrix, $\mathbf{T}_m$. Thus p_{ij} was specified directly based on a product of the conditional probabilities of the three sets $(x_0, \dots, x_{b-1})$, $(z_0, \dots, z_{b-1})$, and $(y_0, \dots, y_{m-2b-1})$ generated in one transition.

4.5 Numerical results

In this section we will present the entropies of the induced measures for some applications of the modified bit-stuffing scheme. The following examples are considered: Three instances of the RLL(d, ∞) constraint, $d = 2, 3$, and 4, as well as the $\diamond(3)$ constraint.

	m	$H_{p=1/2}$	H_p	H_{p_X,p_Y}	$H_{\mathbf{p}opt}$	H_U
RLL$(2,\infty)$	19	0.3917	0.4398	0.4410	0.4415	0.4459
RLL$(3,\infty)$	16	0.3050	0.3606	0.3628	0.3640	0.3686
RLL$(4,\infty)$	15	0.2487	0.2982	0.3110	0.3121	0.3188
$\diamond(3)$	14	0.2763	0.3440	0.3466	0.3478	0.3541

Table 4.1: Numerical results for the entropy of the bit-stuffing induced measures using different biased streams. The final column H_U provides an upper bound on the entropy of the constraints.

Table 4.1 presents entropy results (4.10) for applying the modified bit-stuffing scheme (4.4-4.8) to the considered checkerboard constraints.

The width of the band used is also given. For a given width m, the modified bit-stuffing scheme is specified by the parameters, $p_1(l)$. Starting with one unbiased sequence and increasing the number of parameters used we calculated the following entropies (and thereby lower bounds). $H_{p=1/2}$ gives the entropy using an unbiased bit-stuffer. H_p is the optimized entropy over a single biased sequence, whereas H_{p_X,p_Y} is optimized choosing two different biased sequences: One for the borders, X and Z, and one for the interior, Y. Table 4.2 offers the parameters optimizing the entropy for both one biased and two biased sequences, respectively.

	p_1	$p_{1X,Y}$
RLL$(2,\infty)$	0.289	$(0.220, 0.297)$
RLL$(3,\infty)$	0.250	$(0.193, 0.271)$
RLL$(4,\infty)$	0.220	$(0.175, 0.285)$
$\diamond(3)$	0.225	$(0.160, 0.245)$

Table 4.2: Parameters for the biased streams optimizing the entropy of the modified bit-stuffer for the constraints RLL$(2,\infty)$, RLL$(3,\infty)$, RLL$(4,\infty)$ and $\diamond(3)$ using a band of width $19, 16, 15$ and 14, respectively.

By using a different biased sequence for each column l, $0 \leq l < m - d$, and optimizing the entropy we obtained a slight improvement of the lower bound. The optimization was performed using a steepest descent approach, viewing the entropy as a function of the conditional probabilities (4.4) indexed by the column, $(p_1(0), \ldots, p_1(m-d-1))$, and searching in the direction of the gradient. The obtained entropy is given as $H_{\mathbf{p}opt}$ in Table 4.1. The optimal parameters determined by the search is given in Table 4.3.

Following the description in Example 4.11 in Section 4.4.3, the context based bit-stuffing method was used to improve the lower bound to 0.3497 for the $\diamond(3)$ constraint.

Finally H_U gives the best upper bound we have for the various constraints. The upper bounds were obtained using the method of two-seam cylinder sources [16] as described in Section 2.6.4.

	RLL$(2,\infty)$	RLL$(3,\infty)$	RLL$(4,\infty)$
i		$p_1(i)$	
1	0.1968	0.1603	0.1261
2	0.2283	0.1841	0.1422
3	0.2761	0.2116	0.1669
4	0.2774	0.2550	0.1805
5	0.2779	0.2554	0.2417
6	0.2779	0.2558	0.2478
7	0.2778	0.2558	0.2636
8	0.2778	0.2558	0.2731
9	0.2778	0.2550	0.2733
10	0.2778	0.2531	0.2848
11	0.2778	0.2635	0.4205
12	0.2778	0.2925	0.1261
13	0.2780	0.3836	0.1422
14	0.2781	0.1603	0.1669
15	0.2762	0.1841	0.1805
16	0.3225	0.2116	
17	0.4153		
18	0.1968		
19	0.2283		

Table 4.3: The bit-stuffing probabilities p_1 used to achieve $H_{\mathbf{p}opt}$ for the three constraints RLL$(2,\infty)$, RLL$(3,\infty)$ and RLL$(4,\infty)$ with a width of 19, 16 and 15, respectively.

It should be noted that the upper and lower bounds are quite close to each other, that is their relative difference is 1-2%.

For comparison the lower bounds on the entropy of bit-stuffing presented by Halevy et al. [20] are cited: For RLL$(2, \infty)$ they obtained the lower bound 0.4267, for RLL$(3, \infty)$ the lower bound 0.3402 and finally for RLL$(4, \infty)$ the lower bound 0.2858.

4.6 Discussion

In [41], a detailed study of the relation of actual bit-stuffing coding schemes and the measure induced is presented. By comparison we have restricted the analysis to determining the entropy of the scheme based on a measure.

Simulation results [42], based on performing bit-stuffing, supports a conjecture that the entropy of conventional bit-stuffing is slightly greater than that of the modified bit-stuffing. If this is the case, the $H_{\mathbf{p}opt}$ of Table 4.1 provides a new lower bound on the entropy of conventional bit-stuffing.

As a special case of the modified bit-stuffing scheme one could consider deterministic borders $\mathbf{X}$ and $\mathbf{Z}$. This could be motivated by a desire of having a synchronization component or by types of constraints for which bit-stuffing is not readily applicable. An example of the latter is 2D (d, k) SRLL constraints, which we investigated in Section 2.7. As Etzion has shown [11] it was always possible to find find a merging array between arbitrary arrays for this constraint. The existence of a solution in between two given arrays is a necessary but not sufficient prerequisite for applying a modified bit-stuffing scheme. It is not clear how to actually modify the scheme to work for this constraint.

However, using a deterministic periodic border defining $\mathbf{X}$ and $\mathbf{Z}$ one can find the max-entropic solution for $\mathbf{Y}$ conditioned on $\mathbf{X}, \mathbf{Z}$ and in this way obtain a lower bound on the entropy of the constraint using (2.6). This is another way of looking at the method of cascading periodic arrays described in Section 2.7.

Chapter 5

Conclusion

We have described the theory of constrained fields as a framework for addressing some of the challenges of code construction for advanced data storage devices that treat the recording media as a surface, rather than a series of tracks.

It was shown that the two dimensional setting harbors some intrinsic difficulties: It is not in general possible to determine the entropy of a constrained field. However, by employing one dimensional techniques we can in many cases obtain good bounds on the entropy. Explicit bounds for many examples was shown.

We showed how a Pickard Model could be used to describe some higher order constraints. Also we presented an iterative method that in some cases can help with choosing the large numbers of parameters in order to obtain a stationary model. A complete stationary model for the No Isolated Bits constraint was presented.

We presented a variation of the encoding scheme of bit-stuffing that is applicable to the class of checkerboard constrained fields. It is possible to calculate the entropy of the coding scheme thus obtaining lower bounds on the entropy of the fields considered. The lower bounds on the RLL(d, ∞) constraints are very tight offering new improvements on the best known bounds for these constraints.

Since a large part of this work has been concerned with obtaining good bounds on the entropies of various constrained fields it only seems fitting to conclude by presenting the best known bounds in Table 5.1. We note that except for the lower bound for the NIB constraint [20] and the lower bound for the $\diamond(3)$ constraint [15] the bounds of Table 5.1 have been developed as part of this work. The estimate $\tilde{H}$ of the entropy of the $\diamond(3)$ constraint is taken from [16]. Of course, several authors have investigated constraints not dealt with here, so the list should not be considered exhaustive in any way. As can be seen, the

F	H_L	H_U	$\tilde{H}(F)$
RLL$(2,\infty)$	0.4415	0.445942	0.445489
RLL$(3,\infty)$	0.3640	0.368555	0.367515
RLL$(4,\infty)$	0.3121	0.318804	0.31669
$\diamond(3)$	0.350306	0.354116	0.35030719
Den(2,4)		0.725761	0.68929
Den(4,5)	0.382	0.561824	0.48164
SRLL(1,2)	0.42968	0.4987	0.4678
SRLL(2,3)	0.143	0.2570	0.2395
NIZ	0.8724	0.9686	0.9617083
NIB	0.9127	0.9387	0.9231771872

Table 5.1: Best lower and upper bounds on the entropy for various constrained fields. An estimate $\tilde{H}$ of the entropy is given as well.

methods considered here have worked best for what could be considered the simplest constraints, namely the splitting constraints RLL(d,∞) and the checkerboard constraint $\diamond(3)$.

Appendices

Appendix A

Some notes on transfer matrices

In this appendix we will present some relevant theory of non-negative matrices that turn out to be crucial for calculating the entropy of the band sources we consider. Furthermore we discuss some of the issues dealing with the software used to calculate both the transfer matrices and the eigenvalues. A standard reference on non-negative matrices is [43].

A.1 Perron-Frobenius

Definition A.1. A real matrix $A = (a_{ij})$ is called *non-negative* (respectively, *positive* if $a_{ij} \geq 0$ (respectively positive) for all i, j. It is called *stochastic* if $\sum_j a_{ij} = 1$ for all i.

We denote that a matrix A is non-negative by $A \geq 0$. Similarly $A > 0$ means that A is positive. Given a $n \times n$ matrix $A \geq 0$, we can construct its associated *communication graph* with vertices $V = \{1, \ldots, n\}$ and edges $i \rightarrow j$ if and only if $a_{ij} > 0$.

We say that some matrix $A \geq 0$ is *irreducible* if its communication graph is connected. The number of connected components of the communication graph is called the *period* of A. If there is only one such component A is *aperiodic.*

We say that $A \geq 0$ is *primitive* if there exists integer k such that $A^k > 0$. Clearly, a non-negative matrix is primitive if and only if it is irreducible and aperiodic.

A proof of the following powerful theorem can be found in [43].

Theorem A.2 (Perron-Frobenius Theorem). *Let $A \geq 0$ be a primitive matrix. There exists a real eigenvalue λ_A of A such that $\lambda_A > 0$ and $\lambda_A > |\lambda|$ for any eigenvalue $\lambda \neq \lambda_A$ of A. Furthermore, the left eigenvector u and right eigenvector v associated with λ_A can be chosen such that $u^T v = 1$*

If in addition A is stochastic then $\lambda_A = 1$.

If A is stochastic and irreducible with period $d > 1$ then there are exactly d distinct eigenvalues of modulus 1, namely the dth roots of unity and all other eigenvalues of A have modulus strictly less than 1.

We note that transfer matrices of finite state sources are non-negative matrices. We often need to determine the largest positive eigenvalue of some specific non-negative integer matrix when calculating the entropy. From the Perron-Frobenius Theorem we know that such an eigenvalue exists and that it is unique. Due to this, it is called the Perron value of the matrix. But many of the matrices we consider are very large (over a million rows, say), so how do we actually compute the largest eigenvalue?

A.2 The power method

We use a variation of the power method [8]. Let A be some non-negative matrix and let λ be its Perron value. The main idea of the power method is to iterate the expression

$$x^{(n+1)} = Ax^{(n)},$$

where $x^{(0)}$ is some arbitrary positive vector. If ϕ is a linear functional then

$$\phi(x^{(n+1)})/\phi(x^{(n)}) \to \lambda \text{ for } n \to \infty. \tag{A.1}$$

The sum of the vector seems to be a popular functional. Furthermore, renormating the vector after each iteration is done in order to combat round-off errors.

In order to speed up convergence we use the method of Aitken acceleration which we will not cover here. We refer to [8].

Finally, we remark that in the case described in Section 2.7.5 where the transfer matrix has a block cyclic structure we have to take this into account when using the power method.

Thus if p is the period of the transfer matrix, we consider the ratio $\phi(x^{(n+p)})/\phi(x^{(n)})$ when determining convergence to the Perron value.

A.2.1 A useful bound on the computed eigenvalue

Since we are interested in good bounds on the entropies of the various fields we consider, we must ensure that the precision of the determined Perron values is at least as high as what we want for the entropies.

Fortunately the following proposition [36] offers good bounds on the computed Perron value.

Proposition A.3. *Let A be a non-negative $n \times n$ matrix with Perron value λ. Then for any non-negative vector $x \neq 0$ the following exact bounds are valid*

$$\min_{1\le i\le n} \frac{1}{x_i}\sum_{j=1}^{n} A_{ij}x_j \le \lambda \le \max_{1\le i\le n} \frac{1}{x_i}\sum_{j=1}^{n} A_{ij}x_j. \qquad \text{(A.2)}$$

We simply use the proposed numerically calculated eigenvector in place of x in (A.2). If the upper and lower bounds are within some desired accuracy we have calculated λ to that desired accuracy.

A.2.2 Computing the stationary distribution

Given an irreducible stochastic $n \times n$ matrix Q we know that its stationary distribution π exists, i.e. that $\pi = \pi Q$. Note that, this equation shows that 1 is an eigenvalue of Q with right left eigenvector π. Hence, according to Perron-Frobenius, it is actually the Perron value. Thus we can use a variant of the power method to compute the stationary distribution. We iterate over

$$\pi^{(n+1)} = \pi^{(n)}Q.$$

In each iteration we make certain that $\pi^{(n)}$ sum to 1 and renormate it, if it isn't the case. When $\|\pi^{(n+1)} - \pi^{(n)}\| < \epsilon$ we set $\pi = \pi^{(n)}$. This is how we determined the stationary distributions of the MBS used in determining the lower bounds on the entropies of checkerboard constraints reported in Chapter 4.

A.3 Implementation details

The actual computation of the transfer matrices and the bit-stuffing transition probabilities as well as the subsequent execution of the power method was done with custom made programs written in C. An interesting problem is how to actually compute the transfer matrix for some band source B. Let the extent of the constraint be $N \times M$ and the width of B be m. The states of B are then all the valid $(N-1) \times m$ configurations. It is clear that for most states s, t there will not be a transition between s and t. That is, the transfer matrix is sparse and we only need to store the non-zero elements. One way to compute the valid transitions is to have two nested loops over the states. That is for each pair of states $s = s_1 \dots s_{N-1}$ and $t = t_1 \dots t_{N-1}$ we check whether $s_2 \dots s_{N-1} = t_1 \dots t_{N-2}$ and whether the configuration $s_1 t_1 \dots t_{N-1}$ is valid.

We note that the complexity of this algorithm runs like the square of the number of states, which is a big problem considering the large number of states.

Since most of the states wont overlap consecutively, it seems like a waste of time running through all the states in the inner loop. Instead one could simply run through the valid rows. Unfortunately, the problem with this approach is that we then need to calculate the index of the second state t given the first state s and some valid row r. The easy way to calculate the index (linear search through the states) doesn't solve anything.

However, by building an index table of the valid patterns as we generate the states, we are able to calculate the index by simple table lookup. We do this by first finding the valid rows R. Then finding the valid patterns of height 2 and so on.

Let P_d denote the valid patterns of height d (and width m where m is the width of the band). At each height d we loop through the valid patterns of height $d-1$ and for each pattern $p \in P_{d-1}$ we loop through the valid rows, $r \in R$ to see whether the configuration $\substack{p \\ r}$ is valid or not.

At each level d we have a $|P_{d-1}| \times |R|$ matrix T_d defined in the following manner.

$$(T_d)_{pr} = \begin{cases} -1 & \text{If pattern } p \text{ and row } r \text{ doesn't make a valid configuration.} \\ i_d & \text{If the configuration } \begin{matrix} p \\ r \end{matrix} \text{ is valid.} \end{cases}$$

Here i_d is the index of the last valid configuration in P_d. Each time we find a new valid configuration, we increment i_d and store the configuration in P_d.

While this approach greatly facilitates the calculation of the transfer matrix, especially when $N > 3$ it is clear that these index matrices will be rather large indeed.

This is an example of the classical space time tradeoff, where we gain an increase in speed by using more memory. In general, the sparser the transfer matrix is, the faster the memory intensive approach works.

Appendix B

Band entropies

In this appendix we have collected entropies of band sources of various widths for most of the constrained fields considered in this text. All the entropies are given with the assured accuracy obtained by the bound (A.2) as detailed in Appendix A.2.1.

w	RLL$(2,\infty)$	RLL$(3,\infty)$	RLL$(4,\infty)$
9	4.151764523	3.481576435	3.04860618
10	4.584962501	3.849026673	3.36553452
11	5.042740306	4.216539810	3.68219859
12	5.488229108	4.584062103	3.99880042
13	5.933717831	4.951578811	4.315429024
14	6.321928095	5.319091814	4.63210247
15	6.824695221	5.686603617	4.94879
16	7.270183907	6.05411876	
17	7.715672614		
18	8.161161317		
19	8.60665		

Table B.1: Band entropies $H_B(w)$ for the RLL(d,∞) constrained fields for $d = 2, 3, 4$.

w	NIZ	NIB
5	4.877667574	4.7615512324
6	5.839597483	5.6853328258
7	6.800899900	6.6091787381
8	7.76260823	7.5323559253
9	8.7242690	8.4568516777
10	9.6859299	9.3806811167

Table B.2: Band entropies $H_B(w)$ for the NIZ and NIB constraints.

w	Den(2,4)	Den(4,5)
5	3.84544	3.24487
6	4.51040	3.70658
7	5.20132	4.18582
8	5.89061	4.66004
9	6.57032	5.13660
10	7.25761	5.61824

Table B.3: Band entropies $H_B(w)$ for the density constraint.

w	SRLL(1,2)	w	SRLL(2,3)
6	3.20804	9	2.41029
7	3.67419	10	2.64961
8	4.14306	11	2.88964
9	4.61088	12	3.13240
10	5.07902	13	3.38285
11	5.54707	14	3.61572
12	6.01554	15	3.85523

Table B.4: Band entropies $H_B(w)$ for the SRLL(1,2) and SRLL(2,3) constraints.

Appendix C

Merging arrays for the Den(4,5) constraint

In this appendix we expand upon Example 2.26. We give examples showing that in the general case a merging array for any two valid Den(4,5) arrays has to have width at least 8.

Our exposition is based on the following observation. If there do not exists a merging array of width $w-1$ and we can show that for concrete X and Y that no array of width w exists then the width of the general merging arrays has to be at least $w+1$ even though we may find a specific merging array for X and Y of width less than w. This is easier to explain after having considered some concrete examples. We conclude this argument when covering the specific case of $w = 3$ below.

We consider the following valid 3×2 Den(4,5) configurations.

$$\begin{array}{ccc} A & B & C \\ \boxed{\begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{matrix}} & \boxed{\begin{matrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{matrix}} & \boxed{\begin{matrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \end{matrix}} \end{array}$$

Note that these configurations are chosen for the fact that they force any column next to them to be either all ones (in the case of A and B) or all zeros (in the case of C).

Let $Z = Z_1 \cdots Z_w$ denote a merging array of width w. In the case of $w = 0$, Z is the empty array. We now use various combinations of the valid configurations A, B, C to show that Z cannot exist for $w = 0, 1, \ldots, 7$.

$w = 0$ Consider $X = A = Y$. The array XY is not valid.

$w = 1$ Consider $X = A$ and $Y = C$. To make the array XZ valid, Z has to be all ones. On the other hand to make ZY valid Z has to be all zeros.

$w = 2$ Let again $X = A$ and $Y = C$. We have

$$\begin{array}{cc|cc|cc}
\multicolumn{2}{c}{X} & \multicolumn{2}{c}{Z} & \multicolumn{2}{c}{Y} \\
\hline
\multicolumn{1}{|c}{0} & 0 & 1 & 0 & 1 & \multicolumn{1}{c|}{1} \\
\multicolumn{1}{|c}{0} & 0 & 1 & 0 & 1 & \multicolumn{1}{c|}{1} \\
\multicolumn{1}{|c}{0} & 1 & 1 & 0 & 1 & \multicolumn{1}{c|}{0} \\
\hline
\end{array}$$

Note that ZY_1 is not valid since it contains 6 ones.

$w = 3$ Let $X = A = Y$. We note that $Z_1 = Z_3$ has to be all ones. But then Z contains 6 ones and are thus invalid.

$$\begin{array}{cc|ccc|cc}
\multicolumn{2}{c}{X} & \multicolumn{3}{c}{Z} & \multicolumn{2}{c}{Y} \\
\hline
\multicolumn{1}{|c}{0} & 0 & 1 & & 1 & 0 & \multicolumn{1}{c|}{0} \\
\multicolumn{1}{|c}{0} & 0 & 1 & & 1 & 0 & \multicolumn{1}{c|}{0} \\
\multicolumn{1}{|c}{0} & 1 & 1 & & 1 & 0 & \multicolumn{1}{c|}{1} \\
\hline
\end{array}$$

Note that while no merging array of width 3 exists, in this case we can actually find a merging array of width 1. We simply use $Z = Z_1$ consisting of all ones. This changes nothing, however. If we consider the valid configurations consisting of A atop A and A atop C respectively, we cannot find any merging arrays of width $w = 0, 1, 2, 3$ due to the already covered examples. Of course, this argument can be used for greater values of w.

$w = 4$ Let $X = A$ and $Y = C$. Due to the constraint Z_1 has to be all ones and Z_2 has to contain at least two ones. On the other hand Z_4 has to be all zeros and Z_3 has to contain at least one and at most two ones and hence at least one zero. We have depicted this below:

$$\begin{array}{cc|cccc|cc}
\multicolumn{2}{c}{X} & \multicolumn{4}{c}{Z} & \multicolumn{2}{c}{Y} \\
\hline
\multicolumn{1}{|c}{0} & 0 & 1 & 0 & 1 & 0 & 1 & \multicolumn{1}{c|}{1} \\
\multicolumn{1}{|c}{0} & 0 & 1 & 0 & & 0 & 1 & \multicolumn{1}{c|}{1} \\
\multicolumn{1}{|c}{0} & 1 & 1 & & 0 & 0 & 1 & \multicolumn{1}{c|}{0} \\
\hline
\end{array}$$

We see that $Z_2Z_3Z_4$ contains 6 zeros and thus is invalid.

$w = 5$ Let $X = A$ and $Y = B$. Then $Z_1 = Z_5$ has to be all ones. Furthermore Z_2 and Z_4 has to contain at least two zeros. Thus Z_2 and Z_4 contains at most one one each. Hence, in order for $Z_2Z_3Z_4$ to be valid, Z_3 has to contain at least two ones. This is depicted below.

$$\begin{array}{|cc|ccccc|cc|}
\multicolumn{2}{c}{X} & \multicolumn{5}{c}{Z} & \multicolumn{2}{c}{Y} \\
\hline
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & & & & 1 & 1 & 0 \\
\hline
\end{array}$$

Note that we cannot place a one in any of the remaining positions since then will either $Z_1Z_2Z_3$ or $Z_3Z_4Z_5$ contain at least 6 ones. On the other hand if we fill the remaining positions with zeros then $Z_2Z_3Z_4$ has 7 zeros and are thus invalid.

$w = 6$ Consider $X = A$ and $Y = B$. We have $Z_1 = Z_6$ has to be all ones. Z_2 and Z_5 has to have at least one and at most two ones and hence at least one zero. Hence Z_3 and Z_4 has to have at least two zeros each. This is depicted below.

$$\begin{array}{|cc|cccccc|cc|}
\multicolumn{2}{c}{X} & \multicolumn{6}{c}{Z} & \multicolumn{2}{c}{Y} \\
\hline
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & & 0 & 0 & & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & & & 1 & 1 & 1 & 0 \\
\hline
\end{array}$$

Consider the free position in Z_2. If it is a one then $X_2Z_1Z_2$ contains 6 ones in violation of the constraint. But if it is a zero then $Z_2Z_3Z_4$ contains 6 zeros, also violating the constraint.

$w = 7$ This case has already been covered in Example 2.26.

The above examples show that a merging array for the Den(4,5) constraint has to have width at least $w = 8$.

We have conducted an exhaustive search for merging arrays of width $w = 8$ between any two valid 3×2 arrays and were always able to find one. However, this only shows that it is possible for arrays of height 3. The general case of merging two arbitrary bands is still open.

Bibliography

[1] Roy Adler, Don Coppersmith, and Martin Hassner. Algorithms for sliding block codes—an application of symbolic dynamics to information theory. *IEEE Transactions on Information Theory*, 29(1):5–22, 1983.

[2] Jonathan Ashley and Brian Marcus. Two-dimensional low-pass filtering codes. *IEEE Trans. Communications*, 46(6):724–727, 1998.

[3] C. Berg. 2MA - Matematisk analyse. Matematisk Institut, Københavns Universitet, 1992. Lecture notes.

[4] R. Berger. The undecidability of the domino problem. *Mem. Amer. Math. Soc.*, 66, 1966.

[5] Pierre Brémaud. *Markov chains Gibbs Fields, Monte Carlo Simulation and queues.* Springer, 1999.

[6] N. J. Calkin and H. S. Wilf. The number of independent sets in a grid graph. *SIAM Journal of Discrete Mathematics*, 11(1):54–60, 1998.

[7] Imre Csiszár and Paul C. Shields. Information theory and statistics: A Tutorial. In *Foundations and Trends in Communications and Information Theory*, volume 1, pages 417–528. Now, 2004.

[8] D.Kincaid and W. Cheney. *Numerical Analysis.* Brooks/Cole, 1991.

[9] E. Eleftheriou, T. Antonakopoulos, G.K. Binnig, G. Cherubini, M. Despont, A. Dholakia, U. Duerig, M.A. Lantz, H. Pozidisand,

H.E. Rothuizen, and P. Vettiger. Millipede-a mems-based scanning-probe data-storage system. *IEEE Transactions on Magnetics*, 39(2), 2003.

[10] W. H. Erxleben and M. W. Marcellin. Error-correcting two-dimensional modulation codes. *IEEE Transactions on Information Theory*, 41:1116–1126, July 1995.

[11] T. Etzion. Cascading methods for runlength-limited arrays. *IEEE Transactions on Information Theory*, 43(1):319–324, January 1997.

[12] S. Forchhammer and T. V. Laursen. Cascading 2d constrained arrays using periodic merging arrays. In *International Symposition on Information Theory*, page 109. IEEE, 2003.

[13] S. Forchhammer and T.V. Laursen. Entropy of bitstuffing induced measures on checkerboard constraints. Submitted 2005.

[14] S. Forchhammer and T.V. Laursen. A model for the two-dimensional no isolated bits constraint. Accepted for ISIT 2006.

[15] Søren Forchhammer and Jørn Justesen. Entropy bounds for constrained two-dimensional random fields. *IEEE Transactions on Information Theory*, 45(1):118–127, 1999.

[16] Søren Forchhammer and Jørn Justesen. Bounds on the capacity of constrained two-dimensional codes. *IEEE Transactions on Information Theory*, 46(7):2659–2666, 2000.

[17] Søren Forchhammer and Torben V. Laursen. Entropy of quasi-stationary measures on images with applications to 2d constrained arrays. In *Proceedings fra den 14. Danske Konference i Mønstergenkendelse og Billedanalyse*, pages 51–59. DIKU, 2005.

[18] S. Friedland. Theory of computation of multidimensional entropy with an application to the monomer-dimer problem. *Advances of Applied Math.*, 34:486–522, 2005.

[19] Shmuel Friedland. On the entropy of Z^d subshifts of finite type. *Linear algebra and its applications*, pages 199–220, 1997.

[20] S. Halevy, J. Chen, R.M.Roth, P.H. Siegel, and J.K. Wolf. Improved bit-stuffing bounds on two-dimensional constraints. *IEEE Transactions on Information Theory*, 50:824–838, 2004.

[21] Shirley Halevy and Ron M. Roth. Parallel constrained coding with application to two-dimensional constraints. *IEEE Transactions on Information Theory*, 48(5):1009–1020, 2002.

[22] J. F. Heanue, M. C. Bashaw, and L. Hesselink. Volume holographic storage and retrieval of digital data. *Science*, pages 749–752, 1994.

[23] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata theory, languages and computation.* Addison-Wesley, 1979.

[24] L.P. Hurd, J.Kari, and K.Culik. The topological entropy of cellular automata is uncomputable. *Ergodig Theory Dynamic. Systems*, 12:255–265, 1992.

[25] Kees A. Schouhamer Immink, Paul H. Siegel, and Jack K. Wolf. Codes for digital recorders. *IEEE Transactions on Information Theory*, 44(6):2260–2299, 1998.

[26] W. Weeks IV and R.E. Blahut. The capacity and coding gain of certain checkerboard codes. *IEEE Transactions on Information Theory*, 44(3):1193–1204, 1998.

[27] J. Justesen. Finite state models of constrained 2D data. In *International Symposition on Information Theory.* IEEE, 2004.

[28] A. Kato and K. Zeger. On the capacity of two-dimensional run-length constrained channels. *IEEE Transactions on Information Theory*, 45(5):1527–1540, 1999.

[29] Akiko Kato and Kenneth Zeger. Partial characterization of the positive capacity region of two-dimensional asymmetric run length constrained channels. *IEEE Transactions on Information Theory*, 46(7):2666–2670, 2000.

[30] B. Kitchens and K. Schmidt. Markov subgroups of $(\mathbb{Z}/2\mathbb{Z})^{\mathbb{Z}^2}$. In P. Walters, editor, *Symbolic Dynamics and its applications*, number 192 in Contemporary Mathematics, pages 265–283. Providence, 1992.

[31] Stan Li. *Markov Random Field Modeling in Computer Vision.* Springer-Verlag, 1995.

[32] Douglas Lind and Brian Marcus. *An Introduction to Symbolic Dynamics and Coding.* Cambridge University Press, 1995.

[33] Brian Marcus, R.M. Roth, and Paul H. Siegel. Constrained systems and coding for recording channels. In Vera Pless and W.C.Huffman, editors, *Handbook of coding theory*, volume II, pages 1635–1764. Elsevier, North Holland, 1998.

[34] Brian Marcus, Paul Siegel, and J. Wolf. Finite-state modulation codes for data storage. *IEEE Journal on selected areas in communications*, (10):5–37, 1992.

[35] Nelson G. Markley and Michael E. Paul. Matrix subshifts for Z^v symbolic dynamics. *Proc. London Math. Soc.*, 43:251–272, 1981.

[36] Z. Nagy and K. Zeger. Entropy bounds for the hard-triangle model. *IEEE Transactions on Information Theory*, to appear.

[37] Zsigmond Nagy and Kenneth Zeger. Capacity bounds for the three-dimensional (0,1) run length limited channel. *IEEE Transactions on Information Theory*, 46(3):1030–1033, 2000.

[38] Zsigmond Nagy and Kenneth Zeger. Asymptotic capacity of two-dimensional channels with checkerboard constraints. *IEEE Transactions on Information Theory*, to appear.

[39] E. Ordentlich and R. M. Roth. Two-dimensional weight-constrained codes through enumeration bounds. *IEEE Transactions on Information Theory*, 46(4):1292–1301, July 2000.

[40] D. Pickard. Unilateral markov fields. *Adv. Appl. Probability*, 12:655–671, 1980.

[41] Ron Roth, Paul Siegel, and Jack Wolf. Efficient coding schemes for the hard-square model. *IEEE Transactions on Information Theory*, 47:1166–76, March 2001.

[42] T. B. Schultz. Coding for nanotech storage. Master's thesis, Technical University of Denmark, 2005.

[43] Eugene Seneta. *Non-negative Matrices and Markov Chains.* Springer-Verlag, second edition, 1981.

[44] Claude E. Shannon and Warren Weaver. *The mathematical theory of communication.* University of Illinois Press, 1949.

[45] R. Talyansky, T. Etzion, and R-M. Roth. Efficient code construction for certain two-dimensional constraints. *IEEE Transactions on Information Theory*, 45(2):3407–3416, 1999.

[46] InPhase Technologies. Half terabit per square inch data density. Press release, March 2006.

[47] A. Vardy, M. Blaum, P.H. Siegel, and G.T. Sincerbox. Conservative arrays: Multidimensional modulation codes for holographics recording. *IEEE Transactions on Information Theory*, 42(1):227–230, 1996.

[48] P. Vettiger, G. Cross, M. Despont, U. Drechsler, U. Durig, B. Gotsmann, W. Haberle, W. M.A. Lantz, H.E. Rothuizen, R. Stutz, and G.K. Binnig. The "millipede" - nanotechnology entering data storage. *IEEE Trans. Nanotechnol*, 1(1):39–55, 2002.