UNIVERSITY OF CALIFORNIA SAN DIEGO

Topics in Lossless Source and Channel Coding

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Electrical Engineering
(Communication Theory & Systems)

by

Spencer William Congero

Committee in charge:

Professor Kenneth Zeger, Chair
Professor Alon Orlitsky
Professor Daniel Rogalski
Professor Paul Siegel
Professor Lance Small

2023

The Dissertation of Spencer William Congero is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisor Ken Zeger for his generosity, guidance, and support. He is always introducing me to new ideas, fueling my curiosity, and urging me to pursue all of my interests. I could not have asked for a better advisor.

I would also like to thank Alon Orlitsky, Dan Rogalski, Paul Siegel, Lance Small, and Alex Vardy for serving on my various committees.

Anything I have done has been possible only with the constant encouragement of my parents, whose rallying belief in me has always spurred me on.

My life is filled everyday with joy and adventure by Sarah Ekaireb, and I am so lucky to have had her support while writing this dissertation.

Finally, I would like to thank Joe Connelly for all of the helpful advice he has given me throughout my time in graduate school.

Chapters 2 through 6 of this dissertation consist of published and submitted journal articles. The dissertation author was the primary investigator and author of each of these papers.

- Chapter 2 is a reprint of the material as it appears in: S. Congero and K. Zeger, "Hexagonal run-length zero capacity region—Part I: Analytical proofs", *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 130-152, January 2022.

- Chapter 3 is a reprint of the material as it appears in: S. Congero and K. Zeger, "Hexagonal run-length zero capacity region—Part II: Automated proofs", *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 153-177, January 2022.

- Chapter 4 is a reprint of the material as it appears in: S. Congero and K. Zeger, "The 3/4 Conjecture for fix-free codes with at most three distinct codeword lengths", *IEEE Transactions on Information Theory*, vol. 69, no. 3, pp. 1452-1485, March 2023.

- Chapter 5 is a reprint of the material as it appears in: S. Congero and K. Zeger, "Characterizations of minimal expected length codes", submitted to *IEEE Transactions on Information Theory*, November 13, 2023.

- Chapter 6 is a reprint of the material as it appears in: S. Congero and K. Zeger, "Competitive advantage of Huffman and Shannon-Fano codes", submitted to *IEEE Transactions on Information Theory*, November 13, 2023.

VITA

2016	Bachelor of Science in Electrical Engineering, Minor in Music Recording, University of Southern California
2017–2023	Teaching Assistant, University of California San Diego
2022	Associate Instructor, University of California San Diego
2023	Master of Science in Electrical Engineering (Communication Theory & Systems), University of California San Diego
2023	Doctor of Philosophy in Electrical Engineering (Communication Theory & Systems), University of California San Diego

PUBLICATIONS

S. Congero and K. Zeger, "Hexagonal run-length zero capacity region—Part I: Analytical proofs", *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 130-152, January 2022.

S. Congero and K. Zeger, "Hexagonal run-length zero capacity region—Part II: Automated proofs", *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 153-177, January 2022.

S. Congero and K. Zeger, "The 3/4 Conjecture for fix-free codes with at most three distinct codeword lengths", *IEEE Transactions on Information Theory*, vol. 69, no. 3, pp. 1452-1485, March 2023.

S. Congero and K. Zeger, "Characterizations of minimal expected length codes", submitted to *IEEE Transactions on Information Theory*, November 13, 2023.

S. Congero and K. Zeger, "Competitive advantage of Huffman and Shannon-Fano codes", submitted to *IEEE Transactions on Information Theory*, November 13, 2023.

ABSTRACT OF THE DISSERTATION

Topics in Lossless Source and Channel Coding

by

Spencer William Congero

Doctor of Philosophy in Electrical Engineering
(Communication Theory & Systems)

University of California San Diego, 2023

Professor Kenneth Zeger, Chair

This dissertation studies several topics in lossless source and channel coding, including: hexagonal run-length-limited constraints; fix-free codes; characterizations of minimal expected length codes; and the competitive advantage of prefix codes.

Chapter 1

Introduction

In 1948, Claude Shannon presented a theoretical foundation for modern communication and information theory in his landmark paper "A Mathematical Theory of Communication" [5]. To set the stage for his results, Shannon included the now well-known "Figure 1" (reproduced here as Figure 1.1), which depicts a block diagram of a general communication system. Over the past 75 years, researchers have drawn on the various fields of mathematics, electrical engineering, computer science, and more, in an effort to better understand each aspect of this diagram.

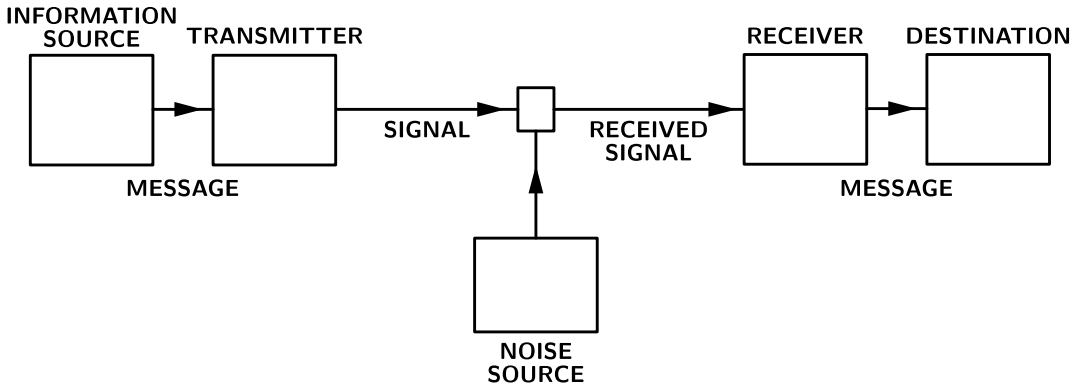


Figure 1.1. Claude Shannon's "Figure 1".

The main idea of Figure 1.1 is that a message (e.g., text or audio) enters on the left from an "Information Source", moves through the diagram, and exits on the right at the "Destination". On its journey, the information is encoded and transmitted across a channel by the "Transmitter", and then decoded by the "Receiver". The transmitted signal could be an electromagnetic wave, a voltage wave, a light wave, a sound wave, or even a hand wave seen across a crowded room. In these and other cases the source and destination points for the message's journey are separated by distance, but in other cases these points may be separated by time, such as when a user writes information to a storage device and then accesses that information sometime later. However, in either case,

a "Noise Source" may introduce errors in the signal, causing the received signal to differ from what was sent by the transmitter. Shannon's work was aimed at better understanding the extent to which reliable (i.e., error-free) communication of a message could be possible across such a noisy channel.

Two of Shannon's most fundamental results are known as the "source coding theorem" and the "channel coding theorem". In the source coding theorem, Shannon expresses the limit of compression during the encoding of an information source quantitatively as the *entropy*, which is an asymptotically attainable lower bound on the average length of any (unambiguous) code for the source. Then, in the channel coding theorem, Shannon expresses the limit of communication of such an encoded source over a noisy channel quantitatively as the channel *capacity*, which is an asymptotically attainable upper bound on the rate of reliable communication achievable over the channel. Prior to Shannon's work, it was not known, perhaps not even believed, that efficient reliable communication was possible over a channel in the presence of noise, since at first glance it would seem that any effort to correct for errors in communication would itself be subject to errors, and so on. After Shannon's work, however, a roadmap had been laid identifying the various aspects of communication whose understanding would help establish the successful global communication system we enjoy today.

When communicating a message through a system, in many cases it is crucial that the source coding operation is invertible, i.e., the original data can be recovered exactly from the encoded data, such as in banking, email, or other textual data applications. Such a situation is an example of *lossless* source coding, since there is no loss of data incurred by the source coding process. Examples of lossless source coding in data compression include the ZIP file format or lossless audio codecs, such as FLAC. However, in other cases, such as in image, video, or speech applications, some loss of data can be tolerated, and in exchange superior compression can be achieved. A common example is the JPEG file format for images, which is typically not invertible (i.e., it loses some original image data), but allows for greater compression than other lossless formats while maintaining a high image quality, at least according to a human observer. Such situations are referred to as *lossy*.

Similarly, communications channels can be described as either lossy or lossless. In the lossy case, there is a noise source in the channel, such as interference from buildings and other transmitters in wireless communication or corruption from physical components in wired communication, which induces errors in the transmitted signal. To combat these errors, the signal is transformed into a longer signal using an error-correcting code, which can help the receiver detect if errors have occurred and correct them, at the cost of lowering the communication rate. In contrast, there are also lossless channels that have no noise source, but instead impose constraints on what messages can be transmitted. An example is a theoretical model of flash memory storage, where restrictions are imposed on the arrays of quantized voltage levels that can be stored, since these restrictions can help prevent errors in real-world devices. The problems in coding for such "constrained channels" are in determining how to

transform an encoded data source into a signal that is accepted by the channel, and in quantifying the efficiency of this transformation.

This dissertation focus exclusively on the lossless cases of both source coding and channel coding. The following two sections briefly describe the specific areas studied in constrained (lossless) channel coding (Chapters 2 and 3), and in lossless source coding (Chapters 4, 5, and 6).

1.1 Constrained channel coding

Chapters 2 and 3 focus on a constrained channel that satisfies the *hexagonal (d,k) run-length-limited constraint*. In particular, the codewords of the channel are binary labelings of a two-dimensional hexagonal lattice, and in each of the three diagonal directions of the lattice the following constraint must be satisfied: any two cells labeled 1 must be separated by at least d zeros, and a consecutive run of more than k zeros can never appear. As an example, Figure 1.2 shows a labeling of a portion of a hexagonal lattice that satisfies the $(1, 4)$ constraint, where every cell labeled 1 is colored red.

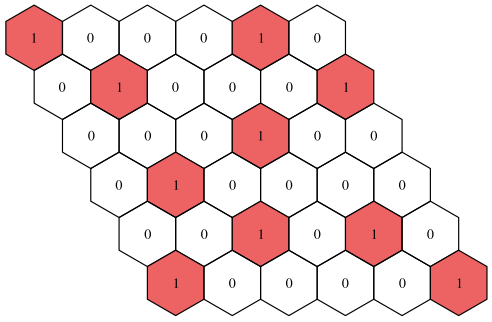


Figure 1.2. Labeling that satisfies the hexagonal $(1, 4)$ constraint.

A fundamental problem is to characterize for which (d, k) pairs the channel has a positive capacity, which in this setting means that the number of labelings of a portion of the hexagonal lattice with n cells that satisfy the constraint increases exponentially in n. Thus the problem is combinatorial in nature, and various special cases have been studied across a wide variety of math, physics, and engineering disciplines. For example, the $(1, \infty)$ constraint has connections to the study of Ising models in statistical mechanics, and even to the problem of placing non-attacking semiqueens (i.e., queens that cannot move along northwest-southeast diagonals) on a chessboard.

In Chapter 2, we show that the channels corresponding to a certain infinite class of (d, k) constraints have zero capacity, which essentially means that these constraints are too restrictive to allow for efficient coding of information in the long term. Chapter 2 is a reprint of the material as it appears in: S. Congero and K. Zeger, "Hexagonal run-length zero capacity region—Part I: Analytical proofs", *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 130-152, January 2022.

In Chapter 3, we show via computer-assisted proofs that channels satisfying certain other (d, k) constraints have either zero or positive capacity. The proofs have an analytical component that reduces their verification to a well-defined but massive computational task, which we then employ a supercomputer to perform. Chapter 3 is a reprint of the material as it appears in: S. Congero and K. Zeger, "Hexagonal run-length zero capacity region—Part II: Automated Proofs", *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 153-177, January 2022.

1.2 Lossless source coding

1.2.1 Fix-free codes

A binary code is *uniquely decodable* if any binary string can be factored in at most one way by the words of the code. Code C_1 in Figure 1.3 is uniquely decodable, since any binary string of length at most three can be written in at most one way by codewords in C_1, and the first four bits of any other binary string suffice to determine uniquely the codeword of C_1 that begins all factorizations of the string. Unique decodability is desirable because there is no ambiguity when decoding a given binary string; if the string is able to be decoded, then it was decoded correctly. However, a disadvantage of uniquely decodable codes is that a decoder may need to see an arbitrarily long portion of the binary string to be able to determine even the first codeword that was transmitted.

In contrast, a binary string encoded via a *prefix* code, which is a uniquely decodable code in which no codeword is a prefix of any other codeword, can be processed by simply reading left-to-right, and decoding each codeword immediately as soon as it has been read. For this reason, prefix codes, such as code C_2 in Figure 1.3, are sometimes called *instantaneous* codes. The English language is not a prefix code, since, for example, "in" and "inform" are proper prefixes of "information".

Similarly, a code is a *suffix* code if no codeword is a suffix of any other codeword. A code that is both a prefix code and a suffix code is called *fix-free*, and the code C_3 in Figure 1.3 is an example. Fix-free codes have the benefit that they can be decoded instantaneously by reading either left-to-right or right-to-left. This property has both efficiency and error-correcting advantages, which has led fix-free codes to be included in the H.263+ and MPEG-4 video coding standards.

Code C_1	Code C_2	Code C_3
0	0	0
01	10	11
110	110	101

Figure 1.3. Examples of uniquely decodable codes. Code C_1 is neither prefix nor suffix, code C_2 is prefix but not suffix, and code C_3 is both prefix and suffix, i.e., fix-free.

Chapter 4 studies fix-free codes, specifically in the context of the well-known 3/4 Conjecture, which was originally proposed in 1996 by Ahlswede, Balkenhol, and Khachatrian [1]. The 3/4 Conjecture states that for any sequence of positive integers satisfying a certain condition, there exists a fix-free code whose codeword lengths are given by that sequence. We prove a special case of this conjecture that contains an infinite class of length sequences, and the proof technique, reminiscent of Shannon's proof of the channel coding theorem, involves selecting fix-free codes at random and showing that on average certain beneficial properties hold. Consequently, we conclude that there must exist at least one specific code with these properties, and we then augment this code to show the existence of the fix-free code we desired. Chapter 4 is a reprint of the material as it appears in: S. Congero and K. Zeger, "The 3/4 Conjecture for fix-free codes with at most three distinct codeword lengths", *IEEE Transactions on Information Theory*, vol. 69, no. 3, pp. 1452-1485, March 2023.

1.2.2 Minimal expected length codes

In Chapter 5 we address the topic of prefix codes that achieve the minimum expected codeword length among all prefix codes for a given probabilistic source. In 1952, Huffman [4] published an algorithm in which a binary tree is constructed by an iterative process, where initially all source symbols are assigned to leaves, and at each step of the algorithm two nodes of smallest probability are combined into a supernode whose probability is their sum. At the conclusion of the algorithm, a binary tree is produced, and after labeling all left branches with '0' and all right branches with '1', a prefix code is defined by associating to each leaf symbol the binary word created by reading the branch labels along the path from the root to the leaf. The code C_1 with codewords 0, 10, 110, and 111, whose code tree is shown in Figure 1.4 in the next subsection, is an example of a code obtained by this procedure applied to a source with symbol probabilities 0.4, 0.3, 0.2, and 0.1.

A remarkable fact is that any prefix code obtained by the Huffman algorithm, referred to as a "Huffman code", achieves the minimum expected length. However, due to the iterative nature of the Huffman algorithm, and the non-linear minimizing operation performed at each step, it can be difficult to analyze the codes that result. In 1978, Gallager [3] provided an interesting necessary and sufficient condition for a prefix code to be a Huffman code, which he called the "sibling property". This property is very useful for proving results about Huffman codes, and we exploit it in multiple places in Chapter 5.

It turns out that the class of expected length minimal prefix codes is strictly larger than the class of Huffman codes for most probabilistic sources, but no analogous necessary and sufficient condition for a prefix code to achieve the minimum expected length has appeared in the literature. In Chapter 5, we provide such a condition by introducing a property called *strong monotonicity*, and proving that a prefix code is optimal if and only if it is "complete" and strongly monotone. We also study various operations on binary code trees that involve

"swapping" two nodes (and their subtrees) in the tree, and characterize relationships between minimal expected length prefix codes and Huffman codes in terms of these node swaps. Chapter 5 is a reprint of the material as it appears in: S. Congero and K. Zeger, "Characterizations of minimal expected length codes", submitted to *IEEE Transactions on Information Theory*, November 13, 2023.

1.2.3 Competitive advantage

In 1991, Cover [2] introduced the notion of "competitive optimality" for a prefix code, which contrasts the more standard concept of expected length minimality. If a prefix code is expected length minimal, then it will eventually produce the shortest encodings of longer and longer sequences of source symbols. In this way, expected length optimality is a measure of performance of a prefix code in the long term. In contrast, if a prefix code is competitively optimal, then, when compared to another prefix code, the first prefix code more often produces a shorter codeword for a randomly chosen source symbol. In this way, competitive optimality is a measure of performance of a prefix code in the short term.

Chapter 6 presents results related to the "competitive advantage" of one prefix code over another, which refines the notion of competitive optimality. Specifically, we define the *competitive advantage* of a prefix code C over another prefix code C' as the probability that C produces a shorter codeword than C', minus the probability that C' produces a shorter codeword than C. A prefix code is competitively optimal if it has a non-negative competitive advantage over every other prefix code for the source.

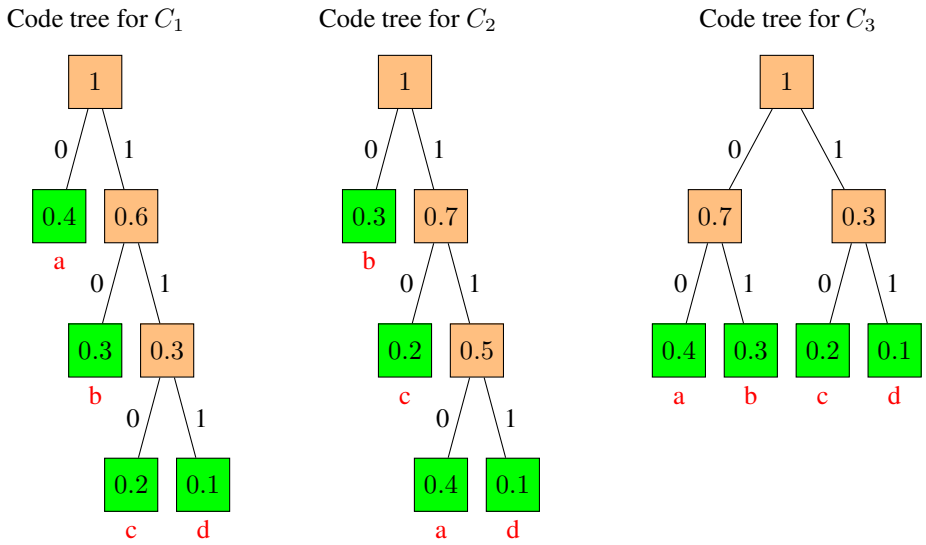


Figure 1.4. Three code trees for a source of size 4. Codes C_1, C_2, and C_3 form a cycle where each code has a positive competitive advantage over the previous code.

Figure 1.4 depicts the code trees of three prefix codes for a source with symbols a, b, c, and d, and probabilities 0.4, 0.3, 0.2, and 0.1, respectively. Comparing codes C_1 and C_2, we see that C_1 produces a shorter codeword for the symbol a (0 vs. 110), but C_2 produces a shorter codeword for the symbols b and c (0 vs. 10, and 10 vs. 110, respectively). Therefore, the competitive advantage of C_2 over C_1 is $P(b) + P(c) - P(a) = 0.3 + 0.2 - 0.4 = 0.1$. Since this competitive advantage is positive, we say C_2 *strictly competitively dominates* C_1. Similarly, one can verify that C_3 strictly competitively dominates C_2, and that C_1 strictly competitively dominates C_3. Thus these three codes form a cycle, where each code "beats" the previous code in the cycle in the competitive advantage sense, similar to how sports competitors can enter into a three-way standoff where each beats another cyclically in a series of competitions. This situation contrasts that of expected length minimality, in which Huffman codes (such as the code C_1) are never "beaten" by any other prefix code for the same source.

In Chapter 6 we provide several results related to the competitive advantage of prefix codes, including Huffman codes and other notable prefix codes called "Shannon-Fano" codes. Our results include characterizing for which sources competitively optimal codes exist as the source size grows to infinity, and quantifying the maximum possible competitive advantage of a prefix code over a Huffman code for the same source. Chapter 6 is a reprint of the material as it appears in: S. Congero and K. Zeger, "Competitive advantage of Huffman and Shannon-Fano codes", submitted to *IEEE Transactions on Information Theory*, November 13, 2023.

References

[1] R. Ahlswede, B. Balkenhol and L. Khachatrian, "Some properties of fix-free codes", 1^{st} *Intas Seminar on Coding Theory and Combinatorics*, Thahkadzor, Armenia, pp. 20 – 33, 1996.

[2] T. M. Cover, "On the competitive optimality of Huffman codes", *IEEE Transactions on Information Theory*, vol. 37, no. , pp. 172 – 174, January 1991.

[3] R. G. Gallager, "Variations on a theme by Huffman", *IEEE Transactions on Information Theory*, vol. 24, no. 6, pp. 668 – 8674, November 1978.

[4] D. A. Huffman, "A method for the construction of minimum-redundancy codes", *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098 – 1101, September 1952.

[5] C. E. Shannon, "A mathematical theory of communication", *Bell System Technical Journal*, vol. 27, no. 4, pp. 623 – 666, October 1948.

Chapter 2

Hexagonal Run-Length Zero Capacity Region, Part I: Analytical Proofs

Abstract

The zero capacity region for hexagonal (d, k) run-length constraints is known for many, but not all, d and k. The pairs (d, k) for which it has been unproven whether the capacity is zero or positive consist of: (i) $k = d + 2$ when $d \geq 2$; (ii) $k = d + 3$ when $d \geq 1$; (iii) $k = d + 4$ when either $d = 4$ or d is odd and $d \geq 3$; and (iv) $k = d + 5$ when $d = 4$. Here, we prove that the capacity is zero for all of case (i), and for case (ii) whenever $d \geq 7$. The method used in this paper is to reduce an infinite search space of valid labelings to a finite set of configurations that we exhaustively examine using backtracking. In Part II of this two-part series, we use automated procedures to prove that the capacity is zero in case (i) when $2 \leq d \leq 9$, in case (ii) when $3 \leq d \leq 11$, and in case (iii) when $d \in \{4, 5, 7, 9\}$, and that the capacity is positive in case (ii) when $d \in \{1, 2\}$, in case (iii) when $d = 3$, and in case (iv). Thus, the only remaining unknown cases are now when $k = d + 4$, for any odd $d \geq 11$.

2.1 Introduction

A one-dimensional run-length constraint imposes both lower and upper bounds on the number of zeros that occur between consecutive ones in a binary string. Specifically, if d and k are nonnegative integers, or ∞, then a binary string is said to satisfy a (d, k) constraint if every consecutive pair of ones in the string has at least d zeros between them and the string never has more than k zeros in a row. It is known that if $k > d$, then the number of (one-dimensional) N-bit binary strings that satisfy the (d, k) constraint grows exponentially in N (e.g., [14]) and that the logarithm (base two) of that number, divided by N, approaches a positive limit as N grows to infinity. This limit is known as the "capacity" of the constraint.

The concepts of (d, k) constraints and capacities have been generalized to two dimensions, where the one-dimensional (d, k) constraint is imposed both vertically and horizontally. Sometimes these two-dimensional constraints are referred to as "rectangular constraints". To determine the capacity of a rectangular constraint, one counts the number of binary labelings of an $N \times N$ square that satisfy the constraint, takes its logarithm, and then divides by the area N^2 of the square. It is known that this quantity approaches a limit $C_{\text{rect}}(d, k)$ (called the "capacity" again) as N grows to infinity (e.g., [18]).[1]

The *zero capacity region* for a particular type of constraint is the set of all pairs (d, k) for which the (d, k) capacity equals zero. If a particular constraint has zero capacity, then the number of valid labelings of a region does not grow exponentially fast in terms of the volume (e.g., length for 1 dimension, area for 2 dimensions, etc.) of the region.

During 1998-2013, various studies of the two-dimensional rectangular capacity were performed for the particular case $C_{\text{rect}}(1, \infty) \approx 0.587891162$ by Calkin and Wilf [5], Weeks and Blahut [34], Baxter [3], Marcus and Pavlov [23,24,28], and in [26]. This rectangular $(1, \infty)$ constraint is sometimes referred to as the "hard square model" by physicists [2], and its capacity is known to equal the rectangular capacity $C_{\text{rect}}(0, 1)$.

For two-dimensional rectangular (d, k) constraints, the zero capacity region was completely characterized in 1999 in [18], where it was shown that the capacity satisfies $C_{\text{rect}}(d, k) > 0$ if and only if $k \geq d + 2$, when $d \geq 1$ (i.e., $C_{\text{rect}}(d, k) = 0$ when $k = d + 1$). It is also known that $C_{\text{rect}}(0, k) > 0$ and $C_{\text{rect}}(k, \infty) > 0$ for all $k \geq 1$. Bounds on the two-dimensional rectangular (d, k) capacity were given in [18], by Sharov and Roth in [31], and were later improved and generalized to higher dimensions by Schwartz and Vardy in [30]. Schwartz and Bruck [29] introduced an interesting rigorous method for obtaining the capacity of general two-dimensional constrained systems, although it is not presently known how to effectively apply it to the hexagonal (d, k) case.

In 2016, Elishco, Meyerovitch, and Schwartz [10] introduced the notion of "semiconstrained systems", in

[1] Or, equivalently, one may count the number of $N \times M$ rectangles satisfying the constraint, take its logarithm, divide by the area NM, and then let both N and M tend to infinity in any manner.

which certain prescribed patterns are forbidden to appear more often than particular designated frequencies. These systems generalize (d, k) constrained systems, since (d, k) constraints require that the forbidden patterns (i.e., patterns violating the d or k constraints) must never occur.[2] Bounds and asymptotics for the capacity of semiconstrained systems were obtained in [10], and also, in 2018, by the same authors in [11], and for the multidimensional case in [12].

Other two-dimensional constraints have been studied in the literature as well. In 1961, Kasteleyn [17] counted the asymptotic number of arrangements of 1×2 tiles that cover a square lattice. In 2006, Forchhammer and Laursen [13] estimated the capacity of a two-dimensional binary code forbidding "isolated bits", i.e., a code where each 0 and 1 cannot be surrounded entirely by bits of the opposite parity. In 2010, Louidor and Marcus [21] determined the capacity of two different two-dimensional constrained systems, namely "charge constrained" and "odd constrained" systems.

Also, in 1961, Wang [32] considered finite sets of certain equal-sized squares, each of whose sides are labeled by one of a given set of colors. These squares later became known as "Wang tiles". Such Wang tiles are used to tile the plane under the constraint that adjacent tiles (horizontally and vertically) share a common color where they meet. Durand, Gamard, and Grandjean [9] in 2014, and Chen, Chen, Hu, and Lin [7] in 2016, counted the number of such Wang tilings and computed a quantity they called the "entropy", or alternatively the "spatial entropy", which is analogous to the capacity calculation described above. The authors in [7] used the phrase "spatial chaos" to describe when the spatial entropy is positive, and gave conditions on when the spatial entropy is zero. In [9], a specific aperiodic tile set (i.e., a tile set such that every tiling of the plane by tiles from this set is aperiodic) was shown to have positive spatial entropy. However, there is no known direct connection between two-dimensional (d, k) constraints and Wang tilings.

We now focus on the family of two-dimensional constraints studied in this paper. A "hexagonal" (d, k) constraint is a different type of two-dimensional run-length constraint, that imposes one-dimensional (d, k) constraints on a hexagonal lattice. Each hexagon in such a lattice has six neighbors, and thus three axes run through it. The one-dimensional constraint must be satisfied along each of the three axes for each hexagon in the lattice. An equivalent way to view the hexagonal constraint on a rectangular lattice is to impose the (d, k) constraint both horizontally and vertically, and also along one of the two diagonal directions (we will use the northeast-southwest direction, but refer to it as the "northeast diagonal") [2, p. 409] (see Figure 2.1). The same diagonal constraint direction is chosen for all squares in the lattice.

The hexagonal (d, k) capacity $C_{\text{hex}}(d, k)$ is known to be positive for certain pairs (d, k). In fact, if $C_{\text{hex}}(d, k) > 0$, then it immediately follows that $C_{\text{hex}}(d', k') > 0$ whenever either $d' < d$ or $k' > k$ (or both),

[2]The (d, k) constrained systems were called "fully constrained" systems in [10].

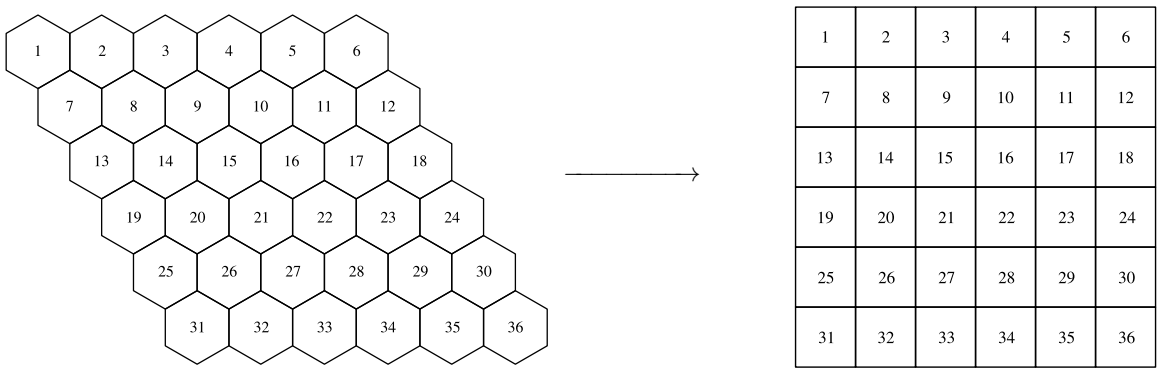


Figure 2.1. Converting a hexagonal lattice into a square lattice using northeast diagonals.

since the constraints weaken in either instance. Positive lower bounds on the hexagonal (d, k) capacity were previously proven for $d = 0$, and for all values of $d \geq 5$ for sufficiently large k (for example, $k = d + 5$ suffices), and now also for $1 \leq d \leq 4$ with our results in Part II. In what follows, we will summarize, for each $d > 0$, the smallest known k such that $C_{\text{hex}}(d, k) > 0$.

The only exactly known non-zero capacity of a hexagonal (d, k) constraint is for the case $(1, \infty)$, which is known in the physics literature as the "hard hexagon model". As with the rectangular constraint, it is easy to show that the hexagonal $(0, 1)$ and $(1, \infty)$ capacities are the same, by reversing the roles of 0s and 1s. The problem of counting the number of patterns in a bounded area that satisfy the hexagonal $(1, \infty)$ constraint was considered in the context of Ising models in physics, as early as in 1944 by Onsager [27], and in 1950 by Wannier [33]. An equivalent problem is to find the number of configurations of non-attacking kings on a chessboard with regular hexagonal cells.

In 1978, Metcalf and Yang [25] conjectured that the capacity of the hexagonal $(1, \infty)$ constraint was $\log_2 e^{1/3} \approx 0.48090$, but this was disproven in 1980 by Baxter and Tsang [4], who obtained a slightly more accurate estimate.

Baxter [1, 2], later in 1980, and then Joyce [15, 16] in 1988, performed numerous intricate calculations, which when combined determine the exact capacity[3] of the hexagonal $(1, \infty)$ constraint (the approximate value is

[3]The exact value is remarkably given by Baxter and Joyce as the logarithm, base two, of the product

$$\begin{aligned} & 4^{-1} 3^{5/4} 11^{-5/12} c^{-2} \\ & \cdot \left(1 - \sqrt{1-c} + \sqrt{2 + c + 2\sqrt{1 + c + c^2}}\right)^2 \\ & \cdot \left(-1 - \sqrt{1-c} + \sqrt{2 + c + 2\sqrt{1 + c + c^2}}\right)^2 \\ & \cdot \left(\sqrt{1-a} + \sqrt{2 + a + 2\sqrt{1 + a + a^2}}\right)^{-1/2} \end{aligned}$$

where $a = -\frac{124}{363} \cdot 11^{1/3}$, $b = \frac{2501}{11979} \cdot 33^{1/2}$, and $c = \left(\frac{1}{4} + \frac{3}{8} a \left((b+1)^{1/3} - (b-1)^{1/3}\right)\right)^{1/3}$.

$C_{\text{hex}}(1,\infty) = C_{\text{hex}}(0,1) \approx 0.4807676$, which is fairly close to the incorrect conjecture of Metcalf and Yang). As a result, one deduces that $C_{\text{hex}}(0,k) > 0$ for all $k \geq 1$.

In 2001, using the technique of finding two distinct tileable squares, Censor and Etzion [6] proved that $C_{\text{hex}}(d, d+4) > 0$ for all even $d \geq 6$. An immediate consequence is that $C_{\text{hex}}(d, d+5) > 0$ for all odd $d \geq 5$, since the hexagonal $(d, d+5)$ constraint is weaker than the $(d+1, d+5)$ constraint. In Part II of our papers, we present a tiling algorithm that automatically generates distinct tileable square labelings that demonstrate positive hexagonal (d,k) capacities for certain pairs (d,k). In particular, we prove that the capacities $C_{\text{hex}}(1,4)$, $C_{\text{hex}}(2,5)$, $C_{\text{hex}}(3,7)$, and $C_{\text{hex}}(4,9)$ are all positive.

Also, we note that the positive hexagonal (d,k) capacities obtained in [6] were for the case of $k = d+4$ when d is even and $d \geq 6$, but the proof technique does not apply to odd $d \geq 5$. In Part II, in contrast to even $d \geq 6$, we show that some of the open cases with $k = d+4$ when d is odd have zero capacity.

We next summarize the pairs (d,k) for which it was previously known that $C_{\text{hex}}(d,k) = 0$. It suffices, for each d, to give the largest k that makes $C_{\text{hex}}(d,k) = 0$. Since each rectangular (d,k) constraint is weaker than the corresponding hexagonal (d,k) constraint, it immediately follows that $C_{\text{hex}}(d,k) \leq C_{\text{rect}}(d,k)$ for all d and k. Thus, in particular, $C_{\text{hex}}(d,k) = 0$ at least whenever $C_{\text{rect}}(d,k) = 0$, namely when $k = d+1 \geq 2$. A stronger result was stated in [19], namely that $C_{\text{hex}}(d, d+2) = 0$ for all $d \geq 1$, but no proof has been published. We prove this result in Section 2.4 in Theorem 2.4.1 for $d \geq 3$, and in Theorem 2.4.3 for $d \in \{1,2\}$. In our Part II, it is also implied by the Forbidden String Algorithm for the cases $d = 3$ and $d = 5$, and by the Constant Position Algorithm when $1 \leq d \leq 9$. Even though for $d \geq 7$ the $k = d+2$ case is implied by the stronger result we prove for the $k = d+3$ case, the proof of Theorem 2.4.1 provides a relatively less complex introduction to the technique used in the stronger case. We note that the proofs we provide here of $C_{\text{hex}}(d, d+2) = 0$ when $2 \leq d \leq 6$ were neither in the previous literature, nor implied by our $k = d+3$ results in this paper.

In [20], it was stated that $C_{\text{hex}}(d, d+3) = 0$ when $d \in \{3,4,5,7,9,11\}$. In [6], Censor and Etzion considered an octagonal (d,k) constraint, which assumes the hexagonal (d,k) constraint plus an additional constraint along the northwest diagonal, and proved that the octagonal (d,k) capacity is zero whenever $k = d+3$ and $d > 0$. However, they did not give any results about whether the hexagonal (d,k) capacity is zero when $k = d+3$, but did pose it as an open question, which partially motivated our present paper. In summary, there have been an infinite number of cases for $k = d+3$, prior to our present paper, where it was unknown if the hexagonal capacity is zero. We answer this open question in completion here.

Specifically, whether $C_{\text{hex}}(d,k)$ is positive or zero has been unproven[4] for the following cases:

(i) $k = d+2$ when $d \geq 2$

[4]Some of these cases were stated in [19] and [20] and are included in Part II for archival purposes.

(ii) $k = d + 3$ when $d \geq 1$

(iii) $k = d + 4$ when either $d = 4$ or d is odd and $d \geq 3$

(iv) $k = d + 5$ when $d = 4$.

Among these cases, we prove here (in Theorem 2.4.1 and Theorem 2.4.3) that the hexagonal capacity equals zero in all of case (i), and (in Theorem 2.2.1) in case (ii) for all $d \geq 7$. In Part II, we prove that the capacity is zero in case (i) when $2 \leq d \leq 9$, in case (ii) when $3 \leq d \leq 11$, and in case (iii) when $d \in \{4, 5, 7, 9\}$, and that the capacity is positive in case (ii) when $d \in \{1, 2\}$, in case (iii) when $d = 3$, and in case (iv).

Table 2.1 summarizes the present knowledge of the zero capacity region when d is less than 19 and k is less than 25, including the results we present in Parts I and II of these papers. The results from Part I are shown surrounded by squares and the results from Part II are shown surrounded by circles. We note that four of the results turn out to be produced by both the methods in Part I and Part II, and we denote them in the table being surrounded by both a circle and a square. Proofs of the results in Part I or Part II have not previously appeared in the literature. We note that although we provide here the first published proofs of the cases where $k = d + 2$, those satisfying $d \geq 7$ are not listed as new results in the table, since they directly follow from our stronger (but more complex) $k = d + 3$ proof,

The four cases shown in our Part II are denoted by "+" signs inside circles. For any fixed d, the leftmost "+" in row d of Table 2.1 represents the smallest k for which it is known that the hexagonal (d, k) capacity is positive. Every "+" in the table represents a positive lower bound, rather than an exact capacity, except for the $(0, 1)$ case. Exact values appear difficult to obtain.

In contrast to proving that a capacity is positive, demonstrating that a capacity is zero requires new techniques, which can be very complex. One technique was used in [18] to prove that the rectangular $(d, d+1)$ capacity is zero for all $d \geq 1$. The technique showed that, asymptotically, the values of the bits stored in a linear amount of space of an $N \times N$ square determine the values of the bits in the remaining quadratic amount space in the square. In other words, the number of different valid labelings of such squares is $2^{O(N)}$, which implies the constraint has zero capacity. In contrast, for a constraint to have positive capacity, there would need to be $2^{\Omega(N^2)}$ different valid labelings of an $N \times N$ square. The same general goal, although with a significantly different approach, will be used in the present paper to show that certain hexagonal constraints have zero capacity.

Specifically, our approach in Part I to proving a particular hexagonal (d, k) capacity is zero is to show that for large enough squares of side length N, with a fixed labeling of a thin outer "frame" of width $k + 1$, at most one valid labeling of the square's interior is possible. This is accomplished by means of assuming, to the contrary, that there exist at least two valid square labelings for a given frame labeling, and then drawing (rather laborious) logical inferences which lead to a contradiction. A series of assumptions is made using a manual backtracking method,

Table 2.1. Summary of the known hexagonal (d, k) zero capacity region for small d and k. Zero and positive capacities are denoted by "0" and "+", respectively. The zeros in squares denote our contributions in the present paper (Part I), while the circled symbols are from our Part II [8], and those with both squares and circles occurred in both Parts I and II. The question marks denote remaining unsolved cases.

$d\backslash k$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
1		0	0	0	⊕	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
2			0	0	⊡	⊕	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
3				0	0	⊡	◎	⊕	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
4					0	0	⊡	◎	◎	⊕	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
5						0	0	⊡	◎	◎	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
6							0	0	⊡	◎	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
7								0	0	0	⊡	◎	+	+	+	+	+	+	+	+	+	+	+	+	+
8									0	0	0	⊡	+	+	+	+	+	+	+	+	+	+	+	+	+
9										0	0	0	⊡	◎	+	+	+	+	+	+	+	+	+	+	+
10											0	0	0	⊡	+	+	+	+	+	+	+	+	+	+	+
11												0	0	0	⊡	?	+	+	+	+	+	+	+	+	+
12													0	0	0	□	+	+	+	+	+	+	+	+	+
13														0	0	0	□	?	+	+	+	+	+	+	+
14															0	0	0	□	+	+	+	+	+	+	+
15																0	0	0	□	?	+	+	+	+	+
16																	0	0	0	□	+	+	+	+	+
17																		0	0	0	□	?	+	+	+
18																			0	0	0	□	+	+	+

which ultimately leads to the desired contradiction. This approach becomes very complex, depending on the level of pushing and popping on the stack of assumptions. Then, since an $N \times N$ square's frame contains $O(N)$ bit locations, the total number of distinct labelings of the square is $2^{O(N)}$, instead of the required $2^{\Omega(N^2)}$ for positive capacity, which proves the capacity is zero.

2.2 Preliminaries

A *square* is an $N \times N$ two-dimensional array, for some positive integer N. A *labeling* of a subset of a square assigns a 0 or 1 to each element of the subset. We will refer to horizontal, vertical, and northeast diagonal lines in a square as *rows*, *columns*, and *diagonals*, respectively, or more generally as *files*. In an $N \times N$ square, all rows and columns have length N, whereas diagonals can have lengths ranging from 1 (at two of the corners) to N. For any positive integer δ, the *frame of width* δ of a square S is the union of the first δ and last δ rows and the first δ and last δ columns of S (see Figure 2.2).

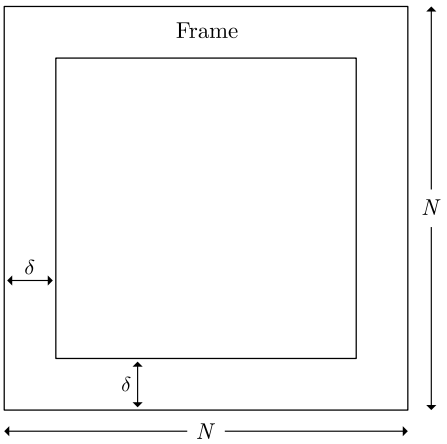


Figure 2.2. Frame of width δ in an $N \times N$ square.

A labeling l on a square S is said to *satisfy the hexagonal* (d, k) *constraint* (or is *valid*) if in every file there are at least d zeros between any two ones, and any run of 0s has length at most k.

The *capacity* of the hexagonal (d, k) constraint is defined as

$$C_{\text{hex}}(d, k) = \lim_{N \to \infty} \frac{\log_2 |L|}{N^2},$$

where L is the set of all labelings of an $N \times N$ square that satisfy the hexagonal (d, k) constraint. The capacity is known to exist for all d, k (e.g., [18]). If $C_{\text{hex}}(d, k) > 0$, then the number of valid labelings is lower bounded as $|L| = 2^{\Omega(N^2)}$.

The main result (proven in Section 2.5) of this paper is that $C_{\text{hex}}(d, d + 3) = 0$ whenever $d \geq 3$, as stated in the following theorem.

Theorem 2.2.1. *The capacity of the hexagonal* (d, k) *constraint is zero whenever* $d \geq 3$ *and* $k = d + 3$.

It has been known [20] that $C_{\text{hex}}(3, 6) = C_{\text{hex}}(4, 7) = C_{\text{hex}}(5, 8) = 0$, and it is shown in our Part II that $C_{\text{hex}}(6, 9) = 0$. We therefore restrict attention to $d \geq 7$ in the proof of Theorem 2.2.1.

An overview of the proof of this result is given in Section 2.3.1, and the actual proof is given in Section 2.5. In Section 2.4, as a preview to Section 2.5, an illustration of the proof technique is given for the simpler case of the $(d, d + 2)$ constraint.

Lemma 2.2.2. *For any nonnegative integers d, k, and δ, if the capacity of the hexagonal* (d, k) *constraint is positive, then there exists a sufficiently large square on which some pair of distinct labelings satisfying the hexagonal* (d, k) *constraint agree on the square's frame of width* δ.

Proof. Let L be the set of valid labelings of an $N \times N$ square and suppose the capacity is positive; then $|L| = 2^{\Omega(N^2)}$. Since the frame's area is $4\delta(N-\delta)$, the number of valid labelings of the frame is at most $2^{4\delta(N-\delta)}$, which grows more slowly than the number of valid labelings of the entire $N \times N$ square. Thus for large enough N, there must exist two distinct labelings of an $N \times N$ square that agree on the frame. ■

Let r and b be labelings of a square S. The *disagreement set* of r and b is

$$D_0 = \{(i,j) \in S : r(i,j) \neq b(i,j)\}.$$

For each point (i,j) in the disagreement set of labelings r and b, we say that (i,j) is *colored red* if $r(i,j) = 1$ and is *colored black* if $b(i,j) = 1$.

For any subset D of a square and any file f, if $X \in f \cap D$, then we say X *is a point of* f *in* D. If X and Y are points of a file f, then X is said to be *before* Y if X is to the left (respectively, below, or southwest) of Y, if f is a row (respectively, a column, or diagonal). If D is a subset of the disagreement set, then the *first* element of f in D is the minimum element for this ordering among points in $f \cap D$.

For any subset of the disagreement set, we define a coordinate system whose origin $(0,0)$ is the first element in the bottommost row, and assume, without loss of generality, that the origin is colored black. A *disagreement diagram* is an illustration showing the color of each point in a subset of the disagreement set. The files in a disagreement diagram are numbered as shown in Figure 2.3. The leftmost point in the bottommost nonempty row of a disagreement subset will be referred to as the *lowest-left point* of the subset.

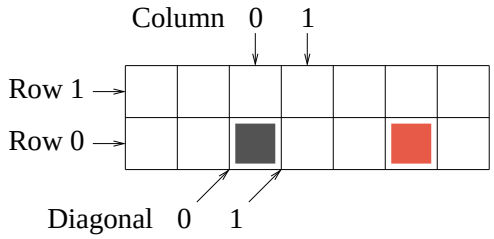


Figure 2.3. Illustration of file indexing in disagreement diagrams. The first disagreement point in the bottommost row is colored black and defined to be located at the origin $(0,0)$, and each file is indexed relative to this origin point. In particular, rows and columns increase in the north and east directions, respectively, and diagonals increase in the southeast direction (i.e., the diagonal containing a point (i,j) is indexed as $i-j$). For example, the red square in the figure is in row 0, column 3, and diagonal 3.

2.3 Description of proof technique

2.3.1 Overview

For the remainder of the paper, we set $k = d + 3$. Our goal is to prove that the hexagonal (d, k) capacity equals zero. We will assume, to the contrary, that $C_{\text{hex}}(d, k) > 0$, and attempt to derive a contradiction.

Let N be a positive integer and let r and b be any two distinct valid labelings of an $N \times N$ square, such that the labelings agree on the square's frame of width $(k + 1)$. Such labelings are guaranteed to exist for sufficiently large N by Lemma 2.2.2. The value of N could conceivably be arbitrarily large, and we know of no analytical upper bound on the size of the square being labeled.

Our goal will be to show that, in fact, r and b cannot be distinct valid labelings, due to conflicts that would arise if they were. Once established, this fact implies that a valid labeling of an $N \times N$ square is completely determined by the values of the labeling on the square's frame of width $(k+1)$. Thus, the number of possible valid labelings of an $N \times N$ square is limited to the number of possible valid labelings of the square's frame, which is of order $2^{O(N)}$. This quantity is too small to induce a positive capacity, since $2^{\Omega(N^2)}$ is needed.

In order to achieve our goal, we analyze the disagreement set of the two hypothetically different valid labelings of the square, and deduce that no such disagreement set can actually exist. As a first step toward this result, we examine the leftmost point in the bottommost row of the disagreement set, arbitrarily color it black, place it at the origin in a coordinate system, and proceed to make a series of additional assumptions about the location of a different point in the disagreement set colored red. We establish (Lemmas 2.B.2 and 2.B.3) that such a red point indeed exists, and, furthermore, that any such red point must lie within at most three neighboring positions of the origin in either direction along some row, column, or diagonal. We choose one of these three files and consider each of the six possible positions for the red point in that file, and determine whether the $(d, d + 3)$ constraint can be preserved when this point is added to the disagreement set. In other words, six "assumptions" are made for the chosen file.

2.3.2 A search tree for invalidating hypothetical labelings

We construct a *search tree* to enable us to prove that any two distinct labelings that agree on a frame of a square cannot both be valid under the hexagonal constraint. The search tree is constructed as follows.

Each node in the tree is a collection of points in a grid, which represents a *potential* disagreement subset of some disagreement set D_0 (depending on the particular hypothetical labelings r and b). If this potential diagreement subset equals an actual disagreement subset of a particular D_0 when the lowest-left points of the two sets are aligned, then the potential disgrement subset will be identified with the actual disagreement subset.

The root of T is constrained to contain exactly one point. For the disagreement set D_0 of r and b on the $N \times N$ square, the root is, by default, the origin of D_0, i.e., the root represents the first point of row 0 in D_0. On the other hand, we may choose the root's single point to be a point in D_0 other than the origin of D_0. In either case, the lowest-left point of every node in the search tree T is aligned with and identified as this non-origin point. (It will turn out that any such non-origin point at the root will always be chosen as a "pseduo-origin".) When the root's single point it not the origin, it will be explicitly noted for clarity.

Each node in the tree is a potential disagreement subset, which represents a possible sequence of assumptions. For a given disagreement subset, each of the six assumptions described in Section 2.3.1 will be an edge in the search tree. Such an edge is given by a starting disagreement subset, together with another disagreement subset that is obtained from the starting one by making the described assumption.

The tree provides a mechanism for invalidating any possible pair of hypothetical labelings r and b, by traversing a particular path for each such r and b. Such a path originates at the tree's root, and the root node itself is calibrated to a specific location within the disagreement set D_0 of the given r and b.

For any given D_0 and choice of the root of T, some of the nodes of T are disagreement subsets, and other are not. We will refer to the nodes of T as disagreement subsets, even though only some of them may be actual disagreement subsets, depending on which particular D_0 is used to search the tree. Note that the topology of the tree T is fixed, and does not vary with the choice of D_0. However, any particular choice of D_0 and the root of T induces a specific path through the tree T.

Each node in the tree T represents the assumptions made on the edges in the unique path from the root of the tree to that node. In particular, each non-root node of the tree corresponds to the set of assumptions of its parent node, together with the one added assumption corresponding to the edge from its parent node to itself.

In some cases, a particular assumption leads to a contradiction, and so that assumption can be eliminated. This elimination corresponds to a node in the tree having no out-edges, i.e., it is a leaf node. On the other hand, if no contradiction to the (d, k) constraint is observed, then the process is repeated at that node by choosing a particular file and then examining the six possible assumptions that can be made as out-edges of that node. This process is repeated at all non-leaf nodes until hopefully all paths terminate, which would result in a finite rooted tree, thus establishing the overall contradiction desired. Fortunately, this occurred and we present the discovered tree in this paper as our main result.

In this way, we build a search tree to represent the various sequences of assumptions made about the contents of the disagreement set, and use this tree to establish that the original positive capacity assumption was false. In other words, for any particular hypothetical pair of distinct labelings that agree on the frame of a square, we can show that at least one of the labelings is not valid under the hexagonal constraint by following a unique

path (determined by the disagreement set induced by the pair of labelings) through the search tree and ariving at a contradiction at a leaf node.

Specifically, the discovered tree contains 50 internal nodes and 151 leaf nodes. Of the 151 leaf nodes, 110 of them can be eliminated quickly by immediate conflicts, i.e., the set of assumptions associated with each node contradicts the hexagonal (d, k) constraint, under the original positive capacity assumption. This leaves only 41 for more careful analysis. All but 9 of these 41 can be readily classified according to three types of relatively easy disagreement patterns. The remaining 9 are particular non-standard, more complicated, "special conflicts" that must be handled separately, but do indeed cause contradictions as well. We note that this method was implemented by hand, not with a computer.

In order to speed up the general tree building technique described above, we observed that at eight leaf nodes and one internal node, it was possible to reduce the complexity of the tree growing process using a concept of "pseudo-origins", which is described in Section 2.5.

We also note that the detailed construction of the tree used in the proof does not depend on any particular choice of the square size N, but does depend in many places on the fact that $k = d + 3$.

2.3.3 Constructing the search tree

The search tree is denoted by T, and its nodes are disagreement subsets. The set of disagreement points corresponding to any node in the tree is a proper subset of the set of disagreement points corresponding to each of its children in the tree. We depict the disagreement subsets using red and black squares, where a red square in a certain position means that position is labeled 1 by r, and a black square in a certain position means that position is labeled 1 by b. Note that no position in a node of T can contain both a red and black square, since these positions are in the disagreement set of r and b by assumption.

By Lemma 2.B.3, there are at most six possible locations for the first red disagreement point of row 0, but by symmetry we need only consider the three locations to the right of $(0, 0)$. Therefore, the root node has branches leading to these three possible positions of the first red disagreement point of row 0 (see Figure 2.4). We use these three disagreement subsets as root nodes for three subtrees, labeled T_1, T_2, and T_3 (see Figures 2.5–2.7).

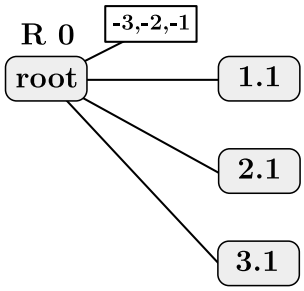


Figure 2.4. Search Tree T_0.

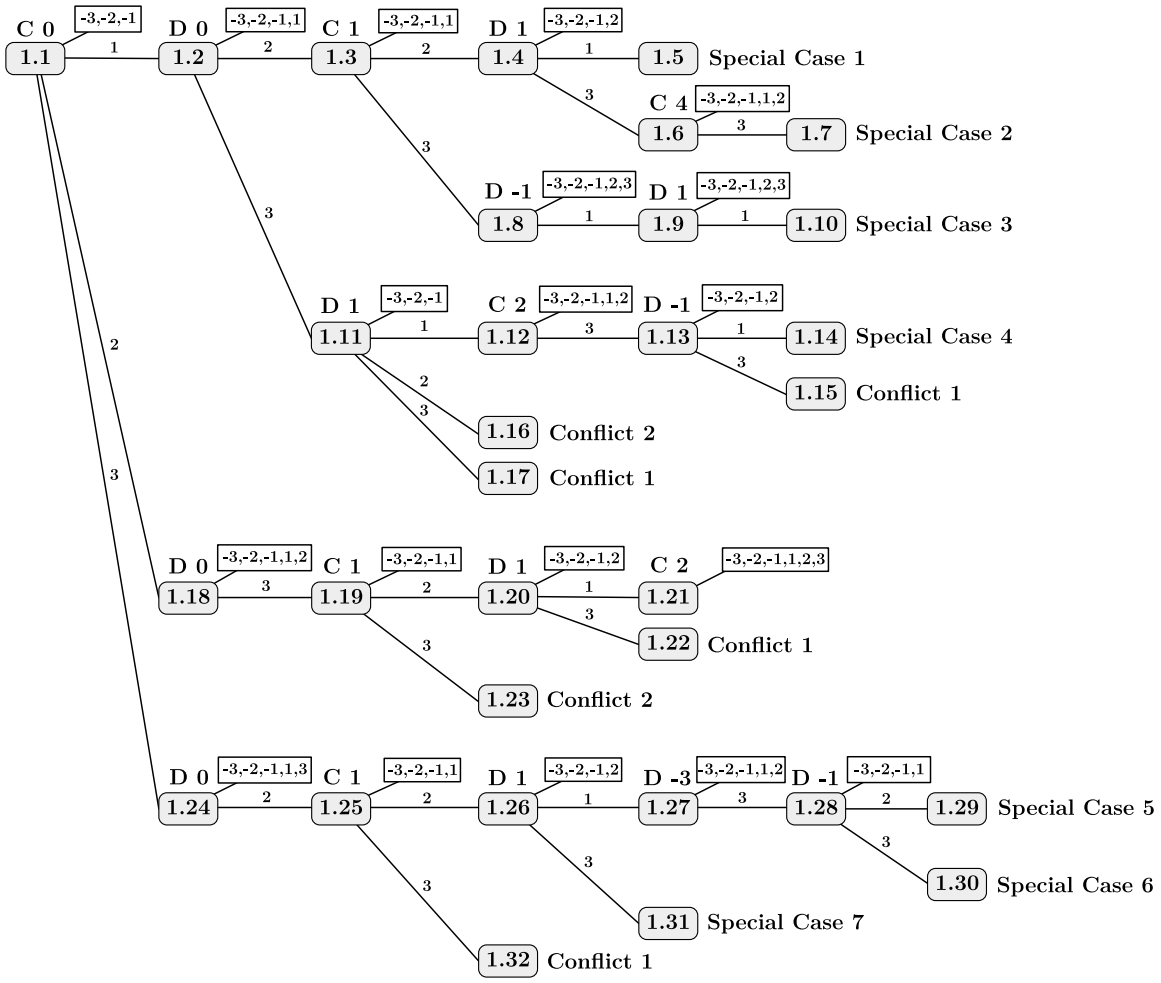


Figure 2.5. Search Tree T_1. Internal nodes are labeled as either R (rows), C (columns), or D (diagonals) to indicate the disagreement subset files used to make further assumptions regarding the possible (d, k) validity of the labelings r and b.

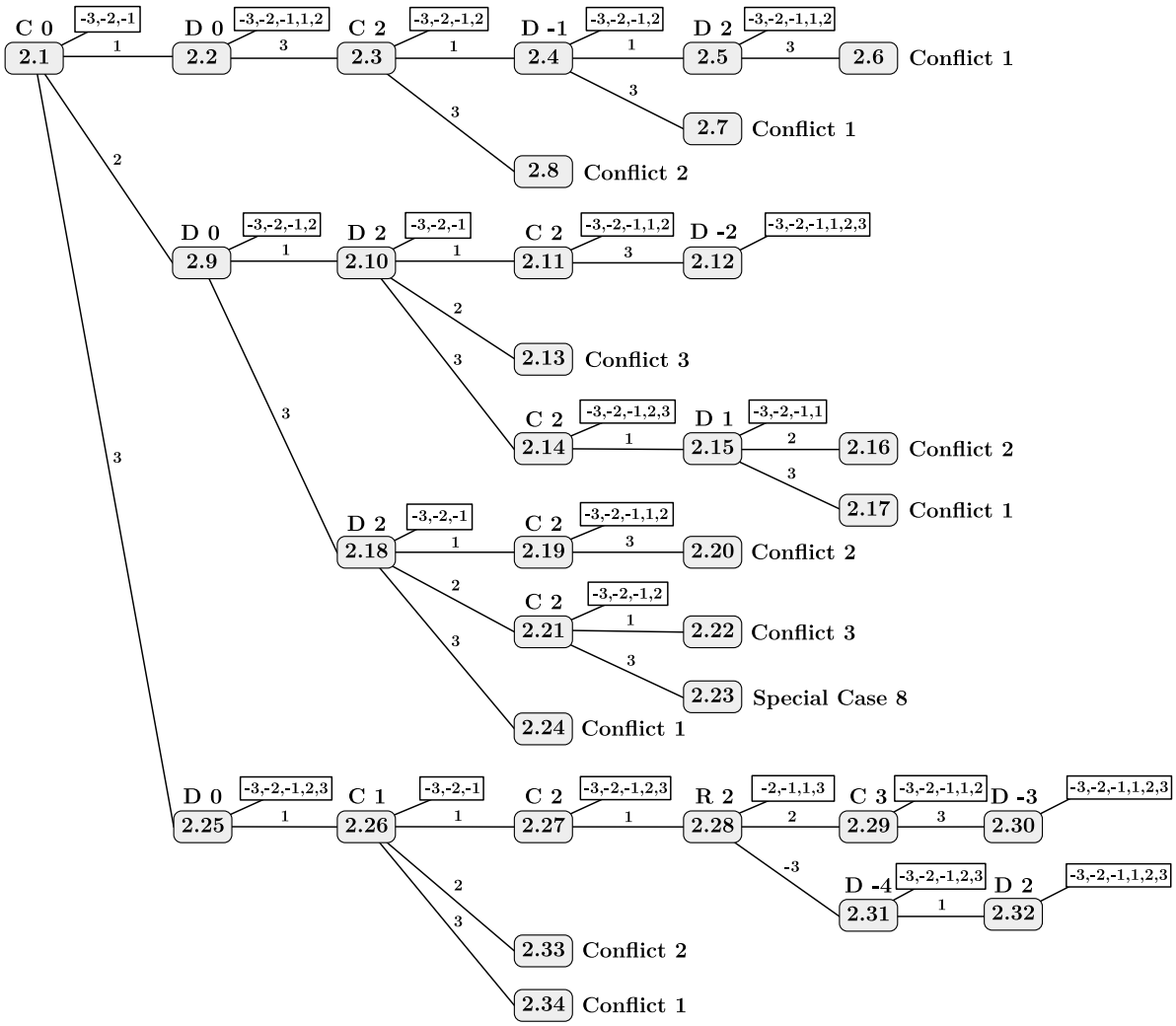


Figure 2.6. Search Tree T_2. Internal nodes are labeled as either R (rows), C (columns), or D (diagonals) to indicate the disagreement subset files used to make further assumptions regarding the possible (d, k) validity of the labelings r and b.

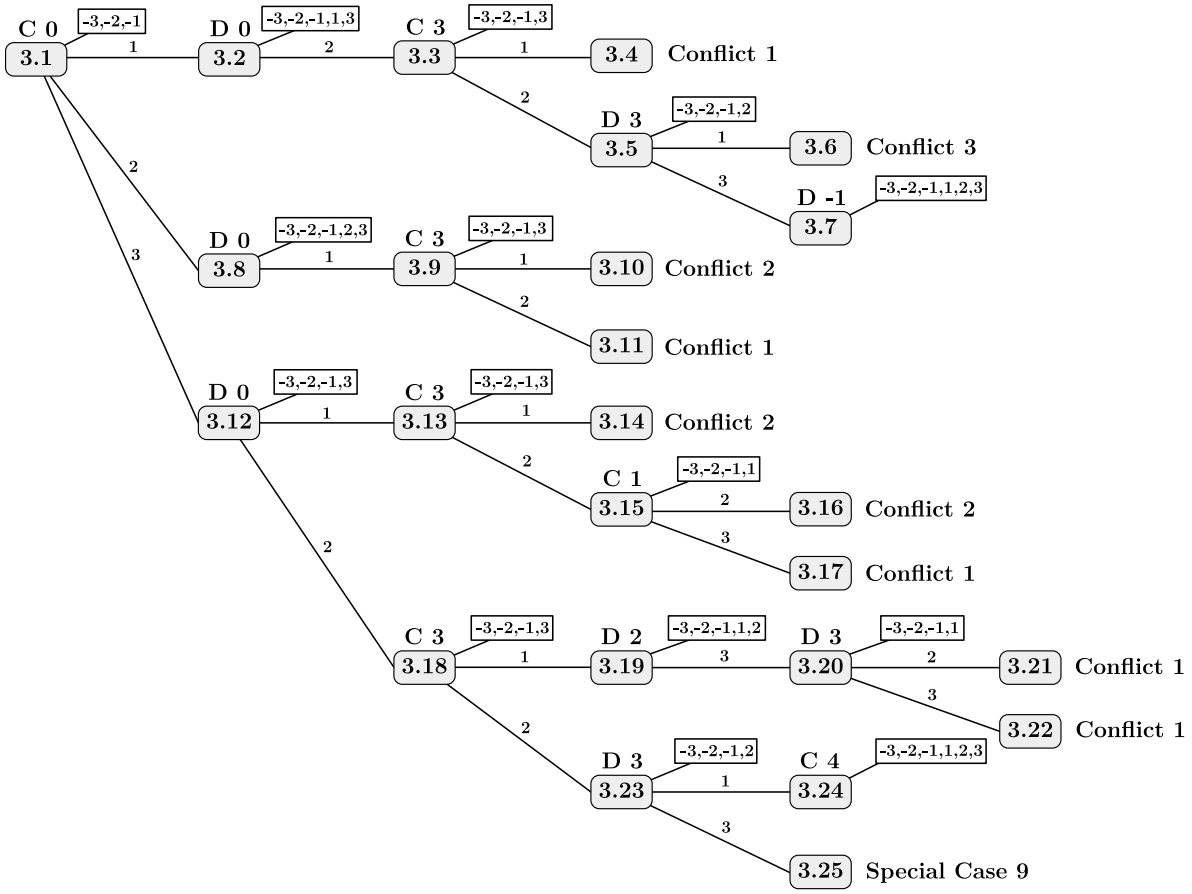
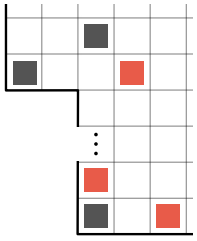
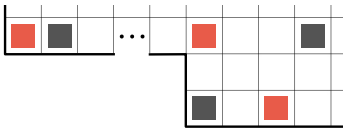


Figure 2.7. Search Tree T_3. Internal nodes are labeled as either R (rows), C (columns), or D (diagonals) to indicate the disagreement subset files used to make further assumptions regarding the possible (d, k) validity of the labelings r and b.

Let D be a subset of the disagreement set D_0 of the two distinct labelings r and b, and let f be a file. We say f is *D-minimal* if f intersects D and the first point x of f in D is also the first point of f in the disagreement set D_0 with the same color as x (see Figure 2.8).



(a) For the disagreement subset D_1 consisting of the three bottommost disagreement points, column 0 is D_1-minimal. On the other hand, for the disagreement subset D_2 consisting of the three topmost disagreement points, column 2 is not D_2-minimal.



(b) For the disagreement subset D_1 consisting of the two leftmost disagreement points, row 0 is D_1-minimal. On the other hand, for the disagreement subset D_2 consisting of the four rightmost disagreement points, row 2 is not D_2-minimal.

Figure 2.8. Examples of files that either are, or are not, D-minimal. In both figures, the labelings agree at any points below or to the left of the solid black border.

If a conflict cannot immediately be found in a node, then children are added by an exhaustive procedure. First, we find a file (if it exists) that is minimal with respect to the current disagreement subset, and that contains only one point of the disagreement subset. Then, we add children corresponding to each possible location in that file for the first disagreement point of the other color, which is guaranteed to exist by Lemma 2.B.2. We denote these different positions by their distance offset Δ from the first disagreement point in the given file. By Lemma 2.B.3, we have $-3 \leq \Delta \leq 3$. If such a file does not exist (which happens just once in the search tree), then the node is treated as a special conflict (see Special Conflict 1 in Lemma 2.B.6).

If a conflict can indeed be found in a node (either by commonly occurring configurations or by a special argument), then this node is made a leaf node of the search tree.

This process is continued until each branch terminates in a leaf node. If all leaves of a search tree violate the constraint, then the disagreement subset shown in the root node of the tree cannot occur, where the lowest-left point of the disagreement subset is the lowest-left point (or, possibly, another type of disagreement point we will call a "pseudo-origin") of the full disagreement set of the two valid labelings.

Frequently, certain Δ values can be immediately eliminated due to a conflict with other points in the

disagreement subset, such as when two positions labeled 1 are positioned closer than distance d apart. Such invalid disagreement subsets are not shown as explicit nodes in the search tree, but are instead displayed as a collection of eliminated delta values grouped in a box branching from a node.

An illustration of how internal tree nodes are handled is given in Section 2.4. We also provide a guide to reading the tree diagrams in Figure 2.9.

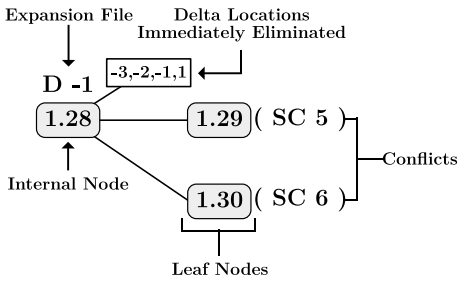


Figure 2.9. Illustration explaining aspects of the search tree diagrams given in Figures 2.4–2.7.

In Subsection 2.B.1 of Appendix 2.B, conflicts in the leaf nodes are discussed in detail, along with various useful lemmas.

2.4 $C_{\text{hex}}(d, d+2) = 0$ whenever $d \geq 1$

We illustrate our general proof technique with a relatively simple and already-known (but unpublished) case of zero hexagonal (d, k) capacity (e.g., see [19]), namely when $k = d + 2$. Traversing the steps of this simplified proof facilitates comprehension of the more complex general result, since they share many common ideas.

The following theorem covers all $d \geq 3$. The remaining cases of $d = 1$ and $d = 2$ are proven later in this section in Theorem 2.4.3. We note that Theorem 2.4.1 also follows from the stronger, but more complex, result that we prove in Theorem 2.2.1, for the cases when $d \geq 7$.

2.4.1 $C_{\text{hex}}(d, d+2) = 0$ whenever $d \geq 3$

Theorem 2.4.1. *The capacity of the hexagonal (d, k) constraint is zero whenever $d \geq 3$ and $k = d + 2$.*

Proof. Suppose, to the contrary, that $C_{\text{hex}}(d, d+2) > 0$. Then, by Lemma 2.2.2, for sufficiently large N, there exist two distinct labelings, r and b, of an $N \times N$ square that agree on the square's frame of width $k + 1$. Let D_0 be the disagreement set of r and b. The points of the square are assigned integer coordinates with the lowest-left point of D_0 denoted by p and located at the origin $(0, 0)$. Without loss of generality, suppose the disagreement point p is colored black, i.e., $b(0, 0) = 1$ and $r(0, 0) = 0$.

We make finite sequences of assumptions about the contents of D_0 that exhaust all possible scenarios, using a depth first search on a tree (see Figure 2.10) that we build for this purpose. We show that every path through this tree leads to a contradiction, implying that the original assumption of two different valid labelings was false. Thus, there cannot exist any nonempty disagreement set D_0, so, in fact, any two labelings that agree on the frame also agree on the rest of the square. This fact limits the number of possible valid labelings of squares to a growth rate which is too small to sustain a positive hexagonal $(d, d+2)$ capacity.

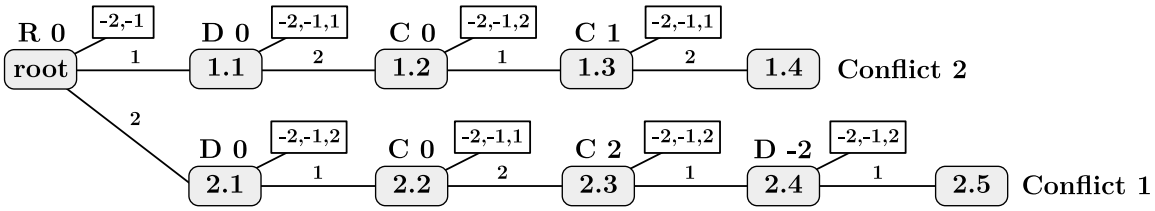


Figure 2.10. Search tree for the proof of Theorem 2.4.1 with the hexagonal $(d, d+2)$ constraint. Each tree node corresponds to one of the diagrams, 1.1 – 1.4 or 2.1 – 2.5, in Figure 2.11, and the colored squares in those diagrams represent assumed values of the labelings r and b, under the assumption that the labelings r and b disagree. The rectangles above and to the right of internal nodes list the Δ values corresponding to disagreement subsets that immediately result in a contradiction (i.e., these subsets are leaves of the tree). Every path from the root to a leaf results in a contradiction, thus disproving the assumption that $C_{\text{hex}}(d, d+2) > 0$.

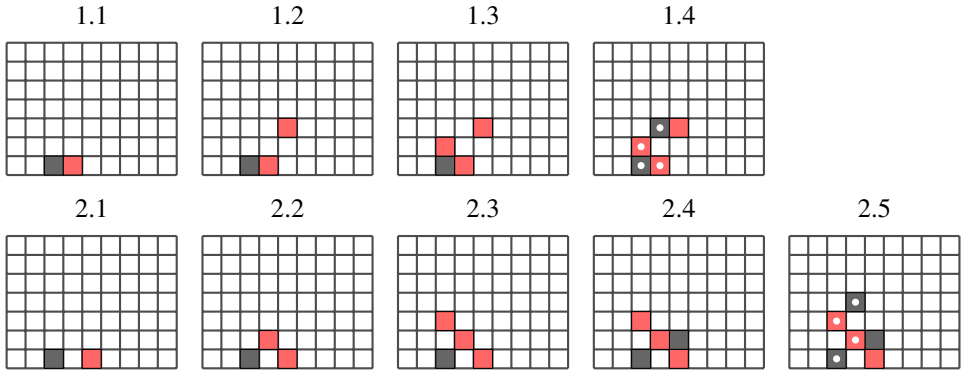


Figure 2.11. The figures show disagreement subsets corresponding to nodes of the search tree in Figure 2.10 for Theorem 2.4.1. The leftmost black point in the lowest row of each diagram has coordinates $(0, 0)$.

The same proof used in Lemma 2.B.3 also shows that for the $(d, d+2)$ constraint, since p is the first point of the lowest row of D_0, there must be a red point $q \in D_0$ within ± 2 positions of p in row 0 (instead of ± 3 as in the $(d, d+3)$ constraint). However, since p is the lowest-left disagreement point, q cannot be to the left of p, so q must be located either at $(1, 0)$ or $(2, 0)$. These two possible arrangements of the black point p and the red point q

are displayed in disagreement diagrams 1.1 and 2.1, respectively[5].

We next show that both arrangements of p and q shown in disagreement diagrams 1.1 and 2.1 lead to at least one of the labelings r or b violating the hexagonal $(d, d+2)$ constraint. To this end, we consider every possible arrangement of the points of D_0 that can arise from the assumptions made in the cases shown in diagrams 1.1 and 2.1. A point of the disagreement subset is selected that is the first point of a D-minimal file, and for which there is not yet a point of the other color within ± 2 positions. Analogous to the $(d, d+3)$ case, such a point of the other color is guaranteed to exist in one of these locations by Lemma 2.B.3.

We then assume each of these four possibilities, one at a time, and show that each leads to a contradiction. For some of these assumptions a contradiction to r and b being different valid labelings appears immediately, while other assumptions are more complicated. In such cases, we add the assumption to the disagreement subset and repeat the process for this new, augmented disagreement subset. This corresponds to adding a new node to the tree with an edge from the previous disagreement subset node.

Each disagreement subset in the tree (in Figure 2.10) is listed in what follows, with the first few being described in detail. Specifically, the coordinates of the first disagreement point of its file are given, as well as the name of the file (i.e., the row, column, or diagonal), and the coordinates of all points within ± 2 positions of the first point within that file. These points are labeled -2, -1, 1, and 2 in Figure 2.12 for cases 1.1, 1.2, and 1.3. For each of these points, an explanation is given of the contradiction that arises from it being included in the disagreement set, or else a new arrangement is considered for further exploration.

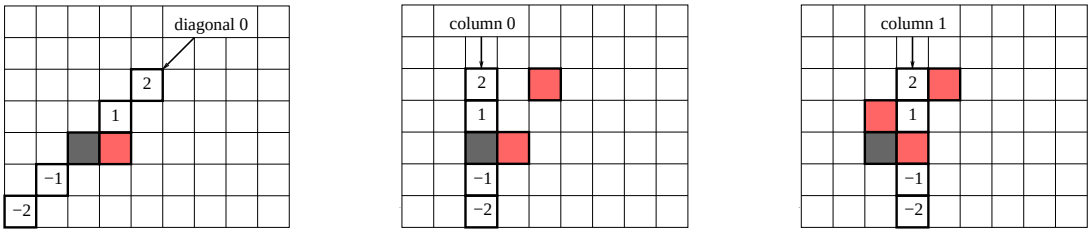


Figure 2.12. Diagrams 1.1, 1.2, and 1.3 from Figure 2.11 with labeled Δ locations.

- 1.1: point $(0, 0)$; diagonal 0; Δ locations $(-2, -2), (-1, -1), (1, 1), (2, 2)$.

  This case corresponds to disagreement diagram 1.1, and we consider the four listed neighbors of $(0, 0)$ along the diagonal stemming from $(0, 0)$ within ± 2 of $(0, 0)$ (see Figure 2.12). Since $(0, 0)$ is colored black, any disagreement point at one of the listed Δ locations must be red. However, there cannot be a disagreement point at $(-2, -2)$ or $(-1, -1)$, since that would contradict the point at $(0, 0)$ being the lowest-

[5]All disagreement diagrams for Theorem 2.4.1 are found in Figure 2.11.

left disagreement point. Also, there cannot be a red disagreement point at $(1,1)$, for then the red point at $(1,0)$ would vertically violate the d constraint in r (since $d \geq 3$). There is no immediate contradiction to coloring a disagreement point at $(2,2)$ red, so we address this possibility in case 1.2.

- 1.2: point $(0,0)$; column 0; Δ locations $(0,-2),(0,-1),(0,1),(0,2)$.

  Since $(0,0)$ is colored black, any disagreement point at one of the listed Δ locations is colored red. However, there cannot be a disagreement point at $(0,-2)$ or $(0,-1)$, since $(0,0)$ is the lowest-left disagreement point. Also, there cannot be a red disagreement point at $(0,2)$, for then the red point at $(2,2)$ would horizontally violate the d constraint in r (since $d \geq 3$). There is no immediate contradiction to coloring a disagreement point at $(0,1)$ red, so we address this possibility in case 1.3.

- 1.3: point $(1,0)$; column 1; Δ locations $(1,-2),(1,-1),(1,1),(1,2)$.

  The chosen file is the column immediately to the right of the point $(0,0)$. Since $(1,0)$ is colored red, any disagreement point at one of the listed Δ locations is colored black. However, there cannot be a disagreement point at $(1,-2)$ or $(1,-1)$, since $(0,0)$ is the lowest-left disagreement point. Also, there cannot be a black disagreement point at $(1,1)$, for then the black point at $(0,0)$ would diagonally violate the d constraint in b (since $d \geq 3$). There is no immediate contradiction to coloring a disagreement point at $(1,2)$ black, so we address this possibility in case 1.4.

- 1.4: Conflict 2.

  The four hollow circles indicate the squares of D_0 that are used to obtain a contradiction in this case. Whereas these squares of D_0 are under the hexagonal $(d, d+2)$ constraint, their arrangement is analogous to an example of a Conflict 2 arrangement shown in Figure 2.15(d) for the hexagonal $(d, d+3)$ constraint.

  A slight modification of Lemma 2.B.5 yields a conflict in this case with the hexagonal $(d, d+2)$ constraint, i.e., at least one of the labelings r and b must not be valid. Since the arrangement of disagreement points in 1.4 does not allow both r and b to be valid labelings, the arrangement of disagreement points in 1.1 also does not allow both r and b to be valid labelings.

- 2.1: point $(0,0)$; diagonal 0; Δ locations $(-2,-2),(-1,-1),(1,1),(2,2)$.

- 2.2: point $(0,0)$; column 0; Δ locations $(0,-2),(0,-1),(0,1),(0,2)$.

- 2.3: point $(2,0)$; column 2; Δ locations $(2,-2),(2,-1),(2,1),(2,2)$.

- 2.4: point $(0, 2)$; diagonal -2; Δ locations $(-2, 0), (-1, 1), (1, 3), (2, 4)$.

- 2.5: Conflict 1 (see Figure 2.14(a) for the analogous version of this conflict for the $k = d + 3$ case).

Since the two arrangements of disagreement points shown in 1.1 and 2.1 do not allow r and b to be valid labelings, the proof is complete.

■

2.4.2 $C_{\textbf{hex}}(d, d + 2) = 0$ when $d \in \{1, 2\}$

If m and b are real numbers and l is a labeling of Z^2, then the set $\{(x, y) \in Z^2 : y = mx + b\}$ is called a *line of* 1*s* of slope m and intercept b if $l(x, y) = 1$ for all (x, y) in the set.

In the proof of the following lemma, we say that a line of 1s *intersects* a string 10^z1 *on the left* (respectively, *right*) if the string is horizontal and its leftmost (respectively, rightmost) 1 is on the line of 1s.

We thank Zsolt Kukorelly for some of the ideas in the following lemma.

Lemma 2.4.2. *If a labeling satisfies the hexagonal* $(2, 4)$ *constraint and has a line of* 1*s with slope* -1*,* $1/2$*, or* 2 *that intersects the string* 10^31 *or* 10^41*, then the labeling consists entirely of parallel lines of* 1*s.*

Proof. We consider each of the three slopes m separately. In each case where a line of 1s is assumed to intersect a string on the left or right, the 1 in the string that lies on the line of 1s will be assumed, without loss of generality, to lie at the origin. Let l denote the binary labeling of points in Z^2.

- Suppose $m = -1$.

  A line of 1s cannot intersect 10^31 on the left or right, for otherwise 101 would occur diagonally.

  If the line of 1s intersects 10^41 on the left (respectively, right), then a line of 1s is forced, with slope -1 and intercept $b = 5$ (respectively, $b = -5$).

  To see this, note that $l(0, 0) = l(5, 0) = l(1, -1) = 1$ and $l(2, -1) = l(3, -1) = l(4, -1) = l(5, -1) = 0$, which implies $l(6, -1) = 1$ to prevent 0^5 horizontally, so by induction half of the line of 1s, $\{(x, y) \in Z^2 : x + y = 5\}$, is forced downward (i.e. when $y \leq 0$). Also, $l(4, 0) = l(4, -1) = l(4, -2) = l(4, -3) = 0$, so $l(4, 1) = 1$ to prevent 0^4 vertically. This implies that the half line of 1s extends upward to give an entire line of 1s. A symmetric argument gives the result when the line of 1s intersects 10^41 on the right.

- Suppose $m = 1/2$.

  A line of 1s cannot intersect 10^31 on the left or right, for otherwise 101 would occur vertically. To see this,

note that $l(0,0) = l(4,0) = l(2,-2) = 1$ implies that 101 occurs diagonally from $(2,-2)$ to $(4,0)$. And similarly on the other side of the line of 1s.

If the line of 1s intersects 10^41 on the left (respectively, right), then a line of 1s is forced, with slope $1/2$ and intercept $b = -(5/2)$ (respectively, $b = (5/2)$).

To see this, note that $l(3,1) = l(4,1) = l(5,1) = l(6,1) = 0$ implies that $l(7,1) = 0$ to prevent 0^5 horizontally, so by induction half of the line of 1s, $\{(x,y) \in Z^2 : y = (x/2) - (5/2)\}$, is forced upward (i.e. when $y \geq 0$). Also, since $l(3,2) = l(3,1) = l(3,0) = l(3,-2) = 0$, we must have $l(3,-1) = 1$ to prevent 0^5 vertically. This implies that the half line of 1s extends downward to give an entire line of 1s. A symmetric argument gives the result when the line of 1s intersects 10^41 on the right.

- Suppose $m = 2$.

  If the line of 1s intersects 10^31 on the left (respectively, right), then two lines of 1s are forced, whose slopes are 2. One of them has intercept $b = -8$ (respectively, $b = 8$), and the other has intercept either $b = -3$ or $b = -5$ (respectively, $b = 3$ or $b = 5$).

  To see this, note that $l(2,2) = l(3,2) = l(4,2) = l(6,2) = 0$ implies $l(5,2) = 1$, and $l(0,-2) = l(1,-2) = l(2,-2) = l(4,-2) = 0$ implies $l(3,-2) = 1$, in both cases to prevent 0^5 horizontally. This forces a line of 1s, namely, $\{(x,y) \in Z^2 : y = 2x - 8\}$. If additionally $l(2,1) = 1$, then it is easy to see that the line of 1s, $\{(x,y) \in Z^2 : y = 2x - 3\}$, is forced, but alternatively if $l(2,1) = 0$, then the line of 1s, $\{(x,y) \in Z^2 : y = 2x - 5\}$, is forced. A symmetric argument gives the result when the line of 1s intersects 10^41 on the right.

  If the line of 1s intersects 10^41 on the left (respectively, right), then two lines of 1s are forced, with slopes 2, and intercepts $b = -5$ and $b = -10$ (respectively, $b = 5$ and $b = 10$).

  To see this, note that $l(5,3) = 0$ to avoid 0^5 diagonally from $(1,1)$ to $(5,5)$, so that $l(6,4) = 0$ to avoid 0^5 horizontally from $(2,2)$ to $(6,2)$, and then $l(4,1) = 0$ to avoid 0^5 diagonally from $(2,0)$ to $(6,4)$. But $l(1,-2) = l(3,0) = l(4,1) = l(5,2) = 0$ implies $l(2,-1) = 1$ to prevent 0^5 diagonally. It then follows that $l(1,2) = l(6,2) = l(3,1) = l(-1,-2) = l(4,-2) = 1$, and by induction we get the following two lines of 1s: $\{(x,y) \in Z^2 : y = 2x - 5\}$ and $\{(x,y) \in Z^2 : y = 2x - 10\}$.

Each of these three cases shows that starting with a line of 1s forces new lines of 1s of the same slope to its left and to its right, provided the original line of 1s intersected either 10^31 or 10^41. If, instead, the line of 1s intersected 10^21 on the left (respectively, right), then it would force a line of 1s through the rightmost (respectively, leftmost) 1 in 10^21. This is because if any other 1 in the line of 1s was the leftmost bit in 10^31 or 10^41, then as

previously shown it would force a line of 1s through the rightmost 1 in that string, contradicting the assumed string 10^21 that intersects the line of 1s.

All of the locations between the original line of 1s and each of these new lines is labeled by 0. By induction, if this process is continued, one concludes that the labeling of the entire plane consists only of parallel lines of 1s.

■

Theorem 2.4.3. *The capacity of the hexagonal (d, k) constraint is zero when $d \in \{1, 2\}$ and $k = d + 2$.*

Proof. When $d = 1$ and $k = 3$, the string 101 is forbidden horizontally, for otherwise the string 0^4 would occur horizontally above it. Thus $C_{\text{hex}}(1, 3) = C_{\text{hex}}(2, 3) \leq C_{\text{rect}}(2, 3) = 0$.

Under the $(2, 4)$ constraint, the only possible zero runs are 0^2, 0^3, and 0^4.

If a valid labeling does not have any runs 0^3 or 0^4, then all zero runs are 0^2, and there are only three possibilities for the labeling of each row, and each such labeling determines the labeling everywhere else.

Next suppose there is at least one run 0^3 or 0^4 in any hexagonal $(2, 4)$ labeling. We will next show that every valid $(2, 4)$ labeling has at least one line of 1s with slope either -1, $1/2$, or 2.

We will first consider the case when 10^31 appears somewhere and then the case when 10^41 appears somewhere. Without loss of generality, we will assume such strings are horizontal and start at the origin.

- Assume $l(0, 0) = l(4, 0) = 1$.

  Then $l(3, 2) = 0$ to prevent 0^6 horizontally from $(0, 1)$ to $(5, 1)$. Since $l(0, 2) = l(2, 2) = l(3, 2) = l(4, 2) = l(6, 2) = 0$, we must have $l(1, 2) = l(5, 2) = 1$, to prevent 0^5 horizontally. The original two assumptions are thus still true if shifted by $(1, 2)$, and, by symmetry about row 0, if shifted by $(-1, -2)$ as well. Then, by induction, we deduce that the following line of 1s is forced: $\{(x, y) \in Z^2 : y = 2x\}$.

- Assume $l(0, 0) = l(5, 0) = l(4, 2) = 1$.

  Then $l(3, 1) = l(4, 1) = l(5, 1) = l(6, 1) = 0$, so $l(2, 1) = l(7, 1) = 1$, to prevent 0^5 horizontally. Also, $l(2, -1) = l(3, 0) = l(4, 1) = l(5, 2) = 0$, so $l(6, 3) = l(1, -2) = 1$ to prevent 0^5 diagonally. The original three assumptions are thus still true if shifted by $(2, 1)$, i.e., $l(2, 1) = l(7, 1) = l(6, 3) = 1$. Futhermore, since $l(0, 0) = l(5, 0) = l(1, -2) = 1$, the original assumptions are true if rotated about row 0. Then, by this symmetry and induction, we deduce that the following line of 1s is forced: $\{(x, y) \in Z^2 : y = x/2\}$.

- Assume $l(0, 0) = l(5, 0) = l(3, 2) = 1$.

  This implies $l(0, 1) = l(1, 1) = l(2, 1) = l(3, 1) = 0$ so $l(-1, 1) = l(4, 1) = 1$. Since $l(2, 2) = l(2, 1) = l(2, 0) = l(2, -1) = 0$ we must have $l(2, 3) = l(2, -2) = 1$. The original three assumptions are thus still

true if shifted by $(-1, 1)$, i.e., $l(-1, 1) = l(4, 1) = l(2, 3) = 1$. Furthermore, since $l(0, 0) = l(5, 0) = l(2, -2) = 1$, the original assumptions are true if rotated about row 0. Thus, by this symmetry and induction, we deduce that the following line of 1s is forced: $\{(x, y) \in Z^2 : y = -x\}$.

- Assume $l(0, 0) = l(5, 0) = 1$ and $l(4, 2) = l(3, 2) = 0$.

  Then $l(2, 2) = l(3, 2) = l(4, 2) = l(5, 2) = 0$, which implies $l(1, 2) = l(6, 2) = 1$. Also $l(2, 1) = 0$ to prevent 0^5 vertically from $(4, -1)$ to $(4, 3)$, and $l(4, 1) = 0$ to prevent 0^5 vertically from $(2, -1)$ to $(2, 3)$. Thus, $l(3, 1) = 1$ to prevent 0^5 horizontally from $(0, 1)$ to $(4, 1)$. This forces $l(4, 3) = 1$ to prevent 0^5 vertically from $(4, -1)$ to $(4, 3)$, so $l(5, 4) = l(4, 4) = 0$. The original four assumptions are thus still true if shifted by $(1, 2)$. Also, since $l(2, 0) = l(2, 1) = l(2, 2) = l(2, 3) = 0$, we get $l(2, -1) = 1$, which implies $l(1, -2) = l(2, -2) = 0$. Therefore, since $l(0, 0) = l(5, 0) = 1$, the original assumptions are true if rotated about row 0. Thus, by this symmetry and induction, we deduce that the following line of 1s is forced: $\{(x, y) \in Z^2 : y = 2x\}$.

We have now shown that all possible strings 10^31 and 10^41 force a line of 1s with slope -1, $1/2$, or 2. So, by Lemma 2.4.2 the entire labeling consists of parallel lines of 1s. This means that the labeling of any row of a rectangle induces one of at most three possible labelings of the entire rectangle, corresponding to the three possible slopes of the parallel lines of 1s. Then, since there are at most $3 \cdot 2^N$ valid $N \times N$ square labelings that contain at least one of 10^31 or 10^41, and there are only three valid $N \times N$ square labelings containing only 10^21, we have $C_{\text{hex}}(2, 4) \leq \lim_{N\to\infty} \frac{1}{N^2} \log_2(3 \cdot 2^N + 3) = 0$. ■

2.5 Main result: $C_{\text{hex}}(d, d + 3) = 0$ whenever $d \geq 3$

In this section we establish that $C_{\text{hex}}(d, d + 3) = 0$ whenever $d \geq 3$ (Theorem 2.2.1). Of these infinite cases, as noted in Section 2.1, it has been previously shown [20] that $C_{\text{hex}}(d, d+3) = 0$ when $d \in \{3, 4, 5, 7, 9, 11\}$ and also when $d = 6$ in our Part II. In what follows we prove the result for all $d \geq 7$, which suffices to complete the proof.

The proof of Theorem 2.2.1 relies directly on Lemmas 2.2.2, 2.B.8, and 2.B.11, the latter two of which are derived in this section from Lemmas 2.B.6, 2.B.9, and 2.B.10.

Since the lowest-left point of the disagreement set D_0 of the two distinct labelings r and b is defined to lie at the origin $(0, 0)$, the labelings r and b agree at certain positions, such as:

(i) at any point in row 0 to the left of $(0, 0)$.

(ii) at any point (x, y) satisfying $x \leq 4$ and $y \leq -1$.

Similarly, we call a point $p = (x, y) \in D_0$ a *pseudo-origin* if the labelings r and b agree in the following positions:

(i) at any point in row y to the left of p.

(ii) at any point (x', y') satisfying $x' \leq x + 4$ and $y' \leq y - 1$.

In the construction of the search tree T, certain files of disagreement subsets are particularly useful in the analysis. For example, each edge of T corresponds to a specific file used for expansion, and this file is marked above the node in the tree diagrams illustrated in Figures 2.4–2.7. Such files in this category are used to examine six possible assumptions for the position of a particular disagreement point, corresponding to the six locations described in Lemma 2.B.3. Another important category of files used in the construction of T consists of those used to achieve conflicts at the leaf nodes. These are enumerated in Table 2.2. We call these two categories of files critical, as described in the following definition.

Definition 2.5.1. A file f is *critical* for a disagreement subset D if f corresponds to an edge from the node D in the tree T or is used to demonstrate a conflict in the leaf D of the tree T.

It is noted that if a file is critical for disagreement subset D, then D contains at least one point in that file.

In the exhaustive search of disagreement subsets using the search tree T, we often find conflicts with the constraint relatively close to the lowest-left point of a disagreement subset. Specifically, the indices of rows, columns, and diagonals that are critical for some disagreement subset in the constructed search tree T are fortunately at most 2, 4, and 4, respectively. This upper bound for the critical columns and diagonals motivates our definition of pseudo-origin, and leads to the following lemma.

Lemma 2.5.2. *Let r and b be distinct labelings of a square that satisfy the hexagonal $(d, d + 3)$ constraint with $d \geq 7$ and agree on the square's frame of width $k + 1$. If x is a pseudo-origin and D is a potential disagreement set in the search tree T with root x, then row 0 and any critical columns and diagonals for D are D-minimal.*

Proof. First note that if the two distinct labelings r and b agree in all positions of a file up to a point p, then any disagreement point within the first 8 positions after p in the file is the first disagreement point of its color in the file. This is because two points of the same color must be separated by at least $d \geq 7$ positions, and so two such points cannot be contained within 8 consecutive positions.

Since x is a pseudo-origin, the two labelings agree in the region containing all points to the left of x, and all points below the row containing x in columns or diagonals with index at most 4. Then by inspecting every potential disagreement subset D in the search tree T, it can be verified that for row 0 or any column or diagonal with index at most 4 that intersects D, the first disagreement points of both colors of such a file f in D occur within

Table 2.2. The critical files used to demonstrate conflicts at leaf nodes in the search tree T.

Leaf node	critical files
1.5	Columns 0,1,2; Diagonals -1,0,1
1.7	Columns 0,1,4; Diagonals -1,0,1
1.10	Columns 0,1,2; Diagonals -1,0,1
1.14	Columns 0,1,2; Diagonals -1,0,1
1.15	Diagonals -1,0
1.16	Diagonals 0,1
1.17	Diagonals 0,1
1.21	Column 2
1.22	Diagonals 0,1
1.23	Columns 0,1
1.29	Columns 0,1,2; Diagonals -3,-1
1.30	Diagonals -3,-1,0,1
1.31	Column 0; Diagonal 1
1.32	Columns 0,1
2.6	Diagonals 0,2
2.7	Diagonals -1,0
2.8	Diagonals -1,0
2.12	Diagonal -2
2.13	Row 2
2.16	Diagonals 1,2
2.17	Diagonals 1,2
2.20	Columns 2,3
2.22	Row 2
2.23	Columns 0,2; Diagonals 0,2
2.24	Diagonals 0,2
2.30	Diagonal -3
2.32	Diagonal 2
2.33	Columns 0,1
2.34	Columns 0,1
3.4	Rows 0,1
3.6	Row 1
3.7	Diagonal -1
3.10	Rows 0,1
3.11	Rows 0,2
3.14	Rows 0,1
3.16	Columns 0,1
3.17	Columns 0,1
3.21	Rows 0,2
3.22	Diagonals 2,3
3.24	Column 4
3.25	Column 0; Diagonal 3

the first 7 positions after the last agreement point of f. Therefore, by the first paragraph, these first disagreement points of each such file in D are the first disagreement points of each such file in D_0. Thus any such file (including, in particular, row 0) is D-minimal, and since any critical column or diagonal has index at most 4, the lemma is proved. ■

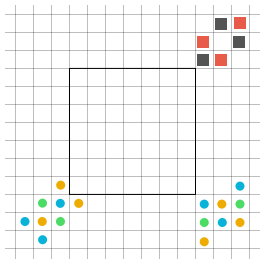
In the proof of Lemma 2.B.6, use is made of various diagrams in Figure 2.13. Before stating the lemma, we explain in detail how these diagrams are used.

In each special conflict diagram in Figure 2.13, the red and black squares indicate disagreement points as previously discussed. In contrast, the circles represent certain important agreement points of the labelings r and b. The circles in each diagram are contained in some critical column or diagonal of the disagreement subset in the same diagram, and these critical files are listed in each diagram's caption. Specifically, the circles in a given column or diagonal denote the possible locations of the last agreement point labeled 1 in that file before any point in the disagreement subset. Additionally, among the circles corresponding to critical columns, the values of r and b do not change within a given color of circle, and the same holds true for the circles corresponding to critical diagonals.

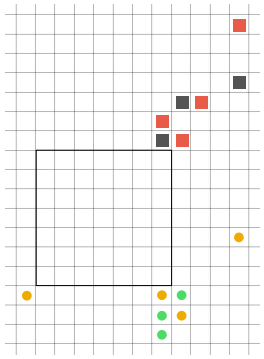
Each critical file contains exactly two disagreement points, one red and one black. The lowest-left black square is always a pseudo-origin, so the two labelings r and b agree at all points below its row (i.e., row 0) when the column index is 4 or less. The colored circles in the diagrams lie in these agreement regions and are associated with critical files.

In any critical file, to satisfy the k constraint, a point labeled 1 must occur within the $(k+1)$ positions before the second of the two shown disagreement points in that file. But since $d \geq 7$ (by assumption) and the largest row index of any disagreement point in a critical file is 6, the d constraint implies this point labeled 1 must occur in the agreement region of the two labelings. Then by the d constraint applied to the first of the two shown disagreement points in the critical file, this agreement point labeled 1 must occur at least d positions away from this first disagreement point. Since $k = d + 3$, this leaves a window of at most 3 positions for such an agreement point labeled 1, and since $d \geq 7$ by assumption, exactly one of the positions in this window must be labeled 1. The circles in each critical file occupy the positions in this window. However, the various configurations of circles do not allow the unique circle labeled 1 in each critical file to be arbitrarily chosen while satisfying the hexagonal (d, k) constraint.

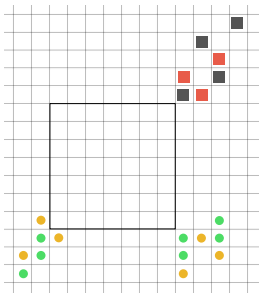
By logical deduction in each Special Conflict diagram, one can verify that, of the possible labelings of all of the circles in the critical columns, either 0, 1, or 2 labelings satisfy the d constraint. If zero such labelings satisfy the d constraint, we depict the circles as hollow with no coloring. If exactly one labeling satisfies the d constraint,



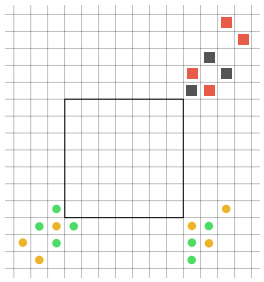
(a) Special Conflict 1 (for 1.5). critical files:
(columns: 0, 1, 2)
(diagonals: -1, 0, 1)



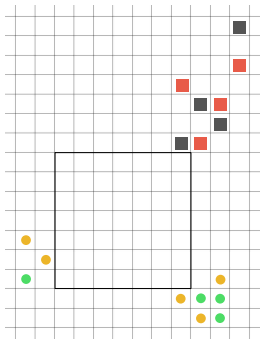
(b) Special Conflict 2 (for 1.7). critical files:
(columns: 0, 1, 4)
(diagonals: 1)



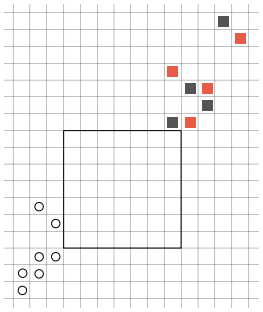
(c) Special Conflict 3 (for 1.10). critical files:
(columns: 0, 1, 2)
(diagonals: -1, 0, 1)



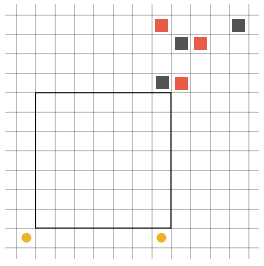
(d) Special Conflict 4 (for 1.14). critical files:
(columns: 0, 1, 2)
(diagonals: -1, 0, 1)



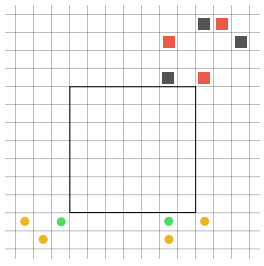
(e) Special Conflict 5 (for 1.29). critical files:
(columns: 0, 1, 2)
(diagonals: -3, -1)



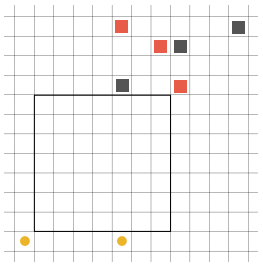
(f) Special Conflict 6 (for 1.30). critical files:
(diagonals: -3, -1, 0, 1)



(g) Special Conflict 7 (for 1.31). critical files:
(columns: 0)
(diagonals: 1)



(h) Special Conflict 8 (for 2.23). critical files:
(columns: 0, 2)
(diagonals: 0, 2)



(i) Special Conflict 9 (for 3.25). critical files:
(columns: 0)
(diagonals: 3)

Figure 2.13. Special Conflicts. The lowest-left point of disagreement subset D is at position $(0, 0)$ and colored black. By the assumption that all critical files for D are D-minimal, both assumed-to-be-valid labelings agree in all colored circles. A square of side length $d = 7$ helps visualize the conflicts, but any larger value of d results in the same conflicts for the same reasons given.

we color circles orange to represent a label of 1, and green to represent a label of 0. If two labelings satisfy the d constraint, which happens only in Special Conflict 1, it suffices to color the circles labeled 1 orange in one labeling and blue in the other labeling. In both cases, we color circles green when the point is labeled 0 in every labeling. We then repeat this colorization process for the circles in critical diagonals.

As a result, Special Conflict 1 contains two possible labelings for both critical columns and diagonals, and Special Conflict 6 contains zero possible labelings of critical diagonals. The other 7 Special Conflicts contain a unique labeling for the circles in critical columns, and a unique labeling for the circles in critical diagonals.

Following this coloration, violations of the d constraint or k constraint could occur due to the possible labelings of the circles in the critical columns and diagonals. We describe such a violation in each Special Conflict, and deduce a contradiction.

To illustrate the coloration process, consider the diagram corresponding to Special Conflict 2. The diagram's caption indicates that the four critical files are columns 0,1, and 4, together with diagonal 1. Considering the columns first, the single circle in column 4 must be labeled 1, which then implies the bottom circle in column 1 must be labeled 1 to avoid violating the d constraint along a diagonal, and this in turn implies the top circle in column 0 must also be labeled 1. In the only critical diagonal, the single circle must be labeled 1. This fixes the coloring of the circles based on the labelings. Then, a conflict can be seen in the row immediately below the black $d \times d$ square outline, since the two orange circles in that row have only $(d-1)$ positions labeled 0 between them.

The figures assume $d = 7$ for these special conflicts, as seen in the 7×7 square outline, but, in fact, any value of $d \geq 7$ causes a conflict for the same reasons (to be given in Subsection 2.B.2 of Appendix 2.B) by considering a larger square outline with side length d. Specifically, the $d \times d$ square outline shown in each image can be used to apply the arguments for any $d \geq 7$ by examining the positions of the circles relative to each other.

Some technical lemmas used in the proof of the main result are given in Subsection 2.B.2 of Appendix 2.B.

The main result then readily follows from Lemmas 2.2.2, 2.B.8, and 2.B.11.

Proof of Theorem 2.2.1. In Part II of this two-part series [8], we prove that $C_{\text{hex}}(d, d+3) = 0$ when $3 \leq d \leq 11$, which overlaps with the cases $7 \leq d \leq 11$ shown here.

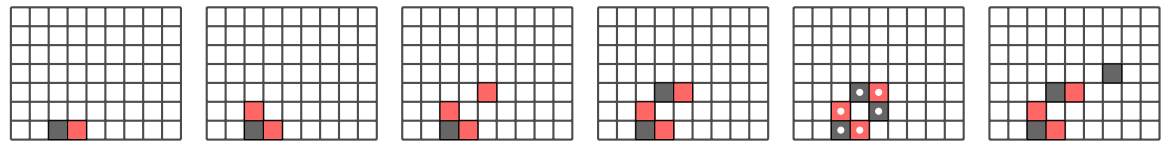
Let $d \geq 7$ and $k = d + 3$ and suppose to the contrary that $C_{\text{hex}}(d, k) > 0$. Then by Lemma 2.2.2, there exist two distinct valid labelings r and b of a sufficiently large $N \times N$ square that agree on the square's frame of width $k + 1$.

By Lemma 2.B.11, there exists a pseudo-origin x in the disagreement set of r and b with the row minimality property (see Definition 2.B.7 in Appendix 2.B). Then by Lemma 2.B.8, there is a conflict with the constraint in either labeling r or b, so it cannot be true that both labelings are valid, a contradiction. Thus, $C_{\text{hex}}(d, d+3) = 0$ for all $d \geq 7$. ■

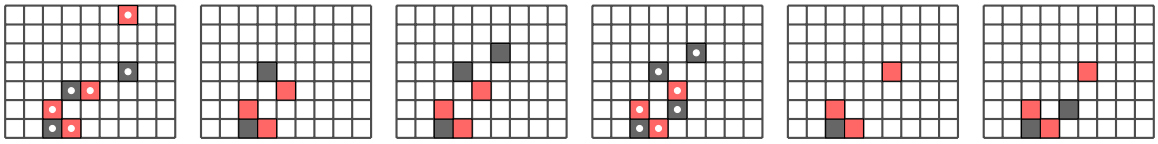
2.A Disagreement Subsets

The figures below (labeled 1.1-1.32, 2.1-2.34, 3.1-3.25) show disagreement subsets corresponding to nodes of the search tree for the main result in Section 2.5.

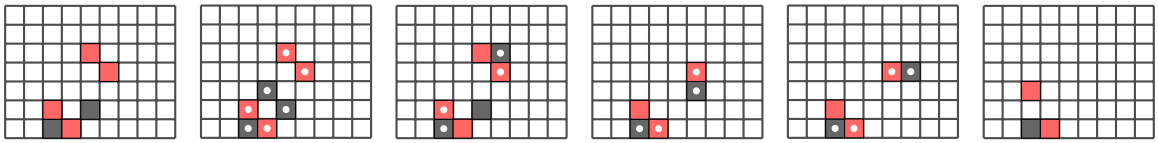
1.1 1.2 1.3 1.4 1.5 1.6



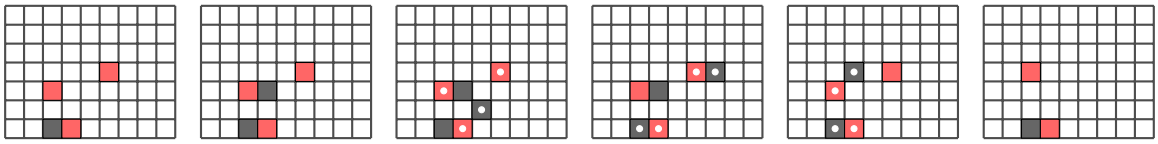
1.7 1.8 1.9 1.10 1.11 1.12



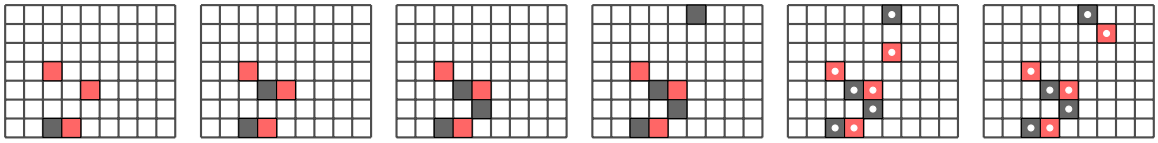
1.13 1.14 1.15 1.16 1.17 1.18



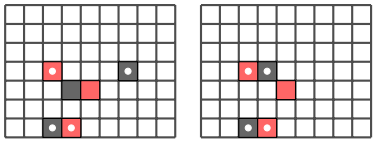
1.19 1.20 1.21 1.22 1.23 1.24



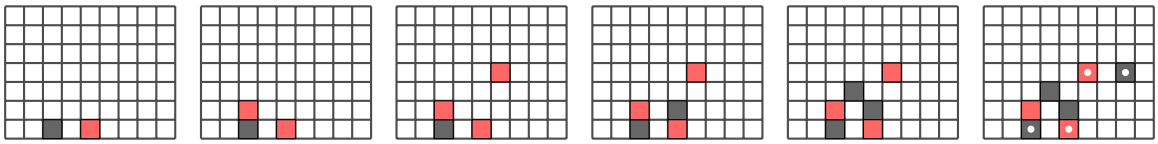
1.25 1.26 1.27 1.28 1.29 1.30



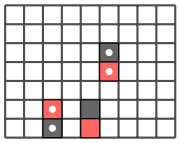
1.31 1.32



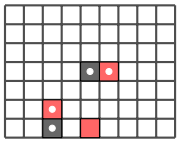
2.1 2.2 2.3 2.4 2.5 2.6



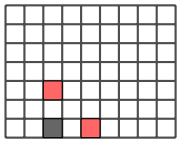
2.7



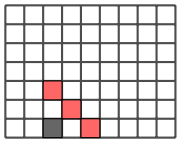
2.8



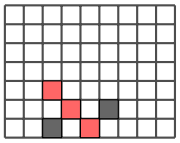
2.9



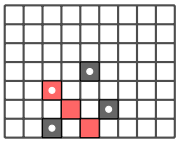
2.10



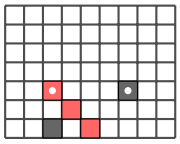
2.11



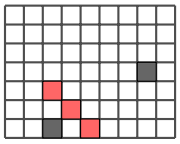
2.12



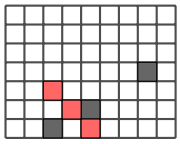
2.13



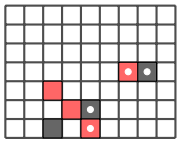
2.14



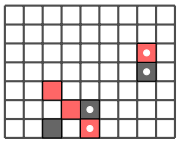
2.15



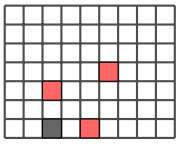
2.16



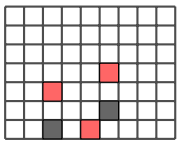
2.17



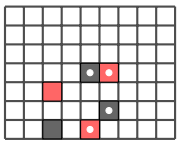
2.18



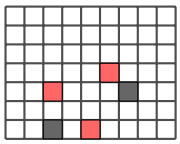
2.19



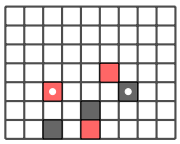
2.20



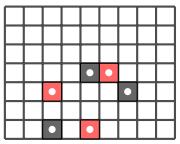
2.21



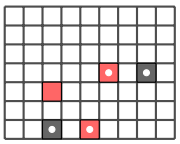
2.22



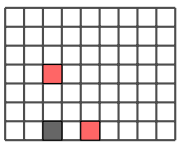
2.23



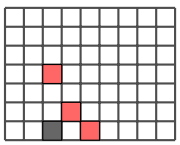
2.24



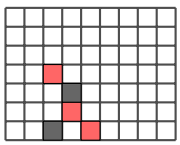
2.25



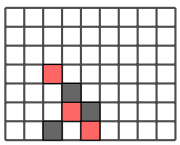
2.26



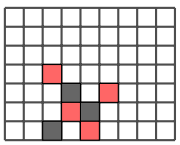
2.27



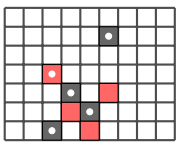
2.28



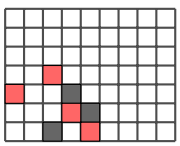
2.29



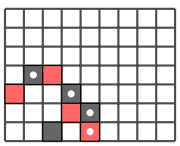
2.30



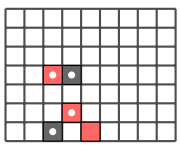
2.31



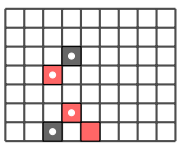
2.32



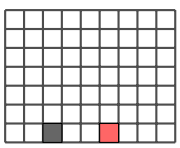
2.33



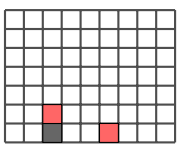
2.34



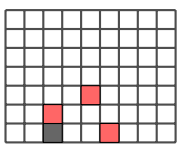
3.1



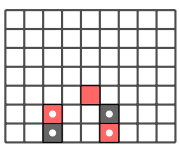
3.2



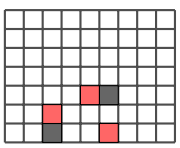
3.3



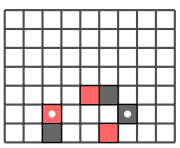
3.4



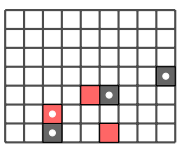
3.5



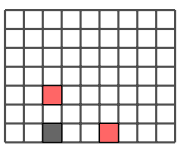
3.6



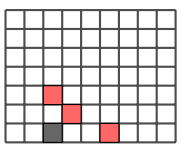
3.7



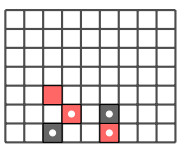
3.8



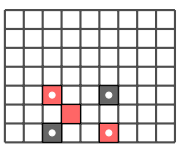
3.9



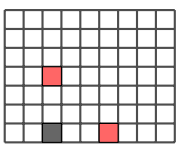
3.10



3.11



3.12



3.13

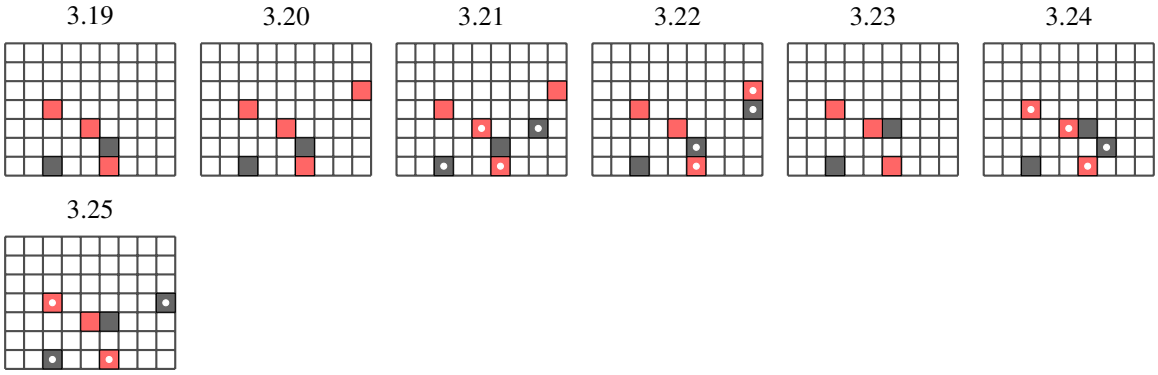
3.14

3.15

3.16

3.17

3.18



2.B Lemmas

2.B.1 Conflicts in leaf nodes

In this subsection we include lemmas that are used in the proof of the main theorem. In particular, we focus on establishing conflicts at leaf nodes that help complete our proof by contradiction.

Lemma 2.B.1. *Let r and b be distinct labelings of a square that satisfy the hexagonal (d, k) constraint and agree on the square's frame of width $\delta \geq k + 1$. Then every file of length at least $2(k + 1)$ has at least two positions where both r and b are labeled 1. In particular, at least one of these positions comes before every disagreement point in the file, and another of these positions comes after every disagreement point in the file.*

Proof. Let f be a file of length at least $2(k + 1)$. In both r and b, one of the first $(k + 1)$ positions of f must be labeled 1 or else the labelings would violate the k constraint. But since r and b agree on the frame, which has width $(k + 1)$, r and b must agree at such a position labeled 1 within the first $(k + 1)$ positions of f. Similarly for the last $(k + 1)$ positions. ■

The following lemma shows that if a file contains a disagreement point labeled 1 by one labeling, then the file also contains a disagreement point labeled 1 by the other labeling.

Lemma 2.B.2. *Let r and b be distinct labelings of a square that satisfy the hexagonal $(d, d + 3)$ constraint with $d \geq 3$ and agree on the square's frame of width $k + 1$. For each file f of the square, if f contains a disagreement point of a particular color, then f also contains a disagreement point of the other color.*

Proof. Suppose f is a file that intersects the disagreement set such that x is the first disagreement point of f. Without loss of generality, suppose x is black. Suppose there does not exist a red disagreement point in f. Then for any point $z \in f$, $b(z) = 0$ implies $r(z) = 0$.

Since $b(x) = 1$, the d constraint implies there are at least d positions on both sides of x in f labeled 0 by b. Therefore, there are also at least d positions on either side of x in f labeled 0 by r. Then since $r(x) = 0$, r must

have a run of at least $2d+1$ consecutive 0s in f. But $d \geq 3$ implies $2d+1 > d+3 = k$, and so the run of $2d+1$ consecutive 0s in r violates the k constraint, a contradiction. ■

As previously mentioned, the quantity Δ denotes the distance offset from the first disagreement point in a given file to the point of the other color (guaranteed to exist by Lemma 2.B.2) in the same file. The following lemma establishes that if the file is D-minimal, then $-3 \leq \Delta \leq 3$.

Lemma 2.B.3. *Let r and b be distinct labelings of a square that satisfy the hexagonal $(d, d+3)$ constraint with $d \geq 5$ and agree on the square's frame of width $k+1$. Let D be a subset of the disagreement set D_0, and let f be a file that is D-minimal. Let $x \in D$ be the first point of f in D. Then the first point of f in D_0 of opposite color to x is located in one of six possible locations, namely within ± 3 positions from x in f.*

Proof. Without loss of generality, let x be colored black. Since f is D-minimal, x is the first black disagreement point of f in D_0. By Lemma 2.B.2, there exists a red disagreement point in f; let y be the first such point. Without loss of generality, we can assume y is after x in f (if not, we can switch the roles of red and black).

Consider the last position z before x where $r(z) = 1$ and $b(z) = 1$. There must exist at least one such z by Lemma 2.B.1. By the d constraint in labeling b, there must be d positions labeled 0 between z and x, and by the k constraint in labeling r, there cannot be more than $d+3$ zeros between z and y. Therefore, x and y cannot be separated by more than 3 positions. ■

Given a disagreement subset $D \subseteq D_0$, the following lemma shows that for a file f that is not D-minimal, there exist at least two points (of different colors) of f in D_0 before the first point of f in D.

Lemma 2.B.4. *Let r and b be distinct labelings of a square that satisfy the hexagonal $(d, d+3)$ constraint with $d \geq 7$ and agree on the square's frame of width $k+1$. Let D_0 be the disagreement set of r and b, and let $D \subseteq D_0$. Suppose there exists a file f intersecting D that is not D-minimal, and let z be the first point of f in D. Then there exist at least two points (of different colors) of f in D_0 before z.*

Proof. Without loss of generality, suppose z is colored black. Since f is not D-minimal, z is not the first black point of f in D_0. Therefore, there exists another point x before z that is the first black disagreement point of f. Furthermore, since $d \geq 7$, x is at least 7 positions before z.

By Lemmas 2.B.2 and 2.B.3, there exists a red disagreement point y that is at most 3 positions away from x, which guarantees y occurs before z. Thus, x and y are two points of different colors of f in D_0 before z. ■

The proof of Lemma 2.B.4 in fact applies to the stronger case where $d \geq 3$, but we need only $d \geq 7$ for our analysis.

A disagreement subset D may contain an arrangement of points that causes at least one of r or b to violate the hexagonal (d, k) constraint, provided that certain files containing these points are D-minimal. We call such arrangements *conflicts*. In particular, the following three types of conflicts arise often in our proofs, and we will refer to them as Conflict 1, Conflict 2, and Conflict 3. Examples of these conflicts are shown in Figure 2.14.

- Conflict 1.

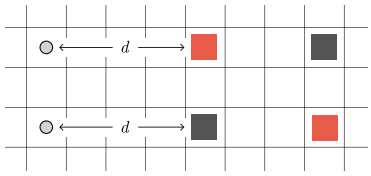
  In this arrangement, the first disagreement points of two parallel files in a disagreement subset D are arranged as shown, for example, in Figure 2.14a. In addition, these files must be D-minimal and separated by fewer than d files.

- Conflict 2.

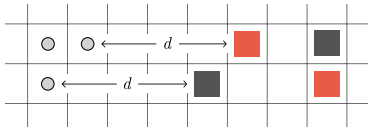
  In this arrangement, the first disagreement points of two adjacent D-minimal files in a disagreement subset D are arranged as shown, for example, in Figure 2.14b. Figure 2.15 provides a full catalog of possible Conflict 2 arrangements, as well as arrangements of disagreement points that may resemble Conflict 2, but are not.

- Conflict 3.

  In this arrangement, the first disagreement points of opposite color of a D-minimal file in a disagreement subset D are separated by 3 positions.



(a) Conflict 1.



(b) Conflict 2.

Figure 2.14. Examples of Conflicts 1 and 2 for the hexagonal $(d, d + 3)$ constraint.

The following lemma shows that the arrangements of points in Conflict 1, Conflict 2, and Conflict 3 all do indeed cause at least one of the labelings r and b to violate the hexagonal (d, k) constraint.

Lemma 2.B.5. *Let D be a subset of the disagreement set of two distinct valid labelings r and b of an $N \times N$*

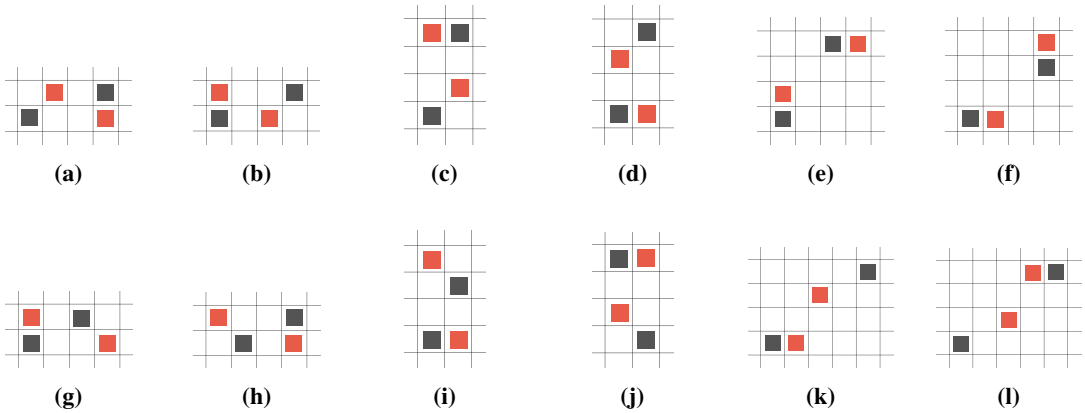


Figure 2.15. Diagrams (a)–(f) show Conflict 2 arrangements. The same arrangements with the colors red and black switched also cause conflicts Diagrams (g)–(l) show arrangements that do not immediately cause conflicts. All arrangements apply to the hexagonal $(d, d+3)$ constraint.

square that agree on the square's frame of width $k+1$. Then the arrangements of points in Conflict 1, Conflict 2, and Conflict 3 each cause at least one of r and b to violate the hexagonal (d, k) constraint.

Proof.

Each of the three types of conflicts previously defined are examined to establish the lemma.

– Conflict 1.

In Figure 2.14a, let the bottom two points be in row 0 and the upper two points be in row i, and suppose $i \leq d$. By Lemma 2.B.1, in each of rows 0 and i there exists a point before the displayed points where both labelings equal 1. Let the last positions where both labelings equal 1 before the displayed points in rows 0 and i be denoted p_0 and p_i, respectively.

Since rows 0 and i are D-minimal and the displayed points are the first disagreements points of their rows in D (as required by Conflict 1), the displayed points are the first disagreement points of their rows in the disagreement set of the two labelings. Therefore, the only possible column that can contain p_0 and p_i is the column that is separated by exactly d columns from the column containing the leftmost disagreement point in each row. However, since row 0 and row i are separated by $i-1 < d$ rows, p_0 and p_i are separated by fewer than d rows. Therefore this arrangement causes a conflict with the d constraint.

Similar arguments show the lemma in cases where the disagreement points are in columns or diagonals instead of rows.

– Conflict 2.

In Figure 2.14b, let the bottom two points be in row 0 and the upper two points be in row 1. By Lemma 2.B.1, in each of rows 0 and 1 there exists a point before the displayed points where both labelings equal 1. Let the last positions where both labelings equal 1 before the displayed points in rows 0 and 1 be denoted p_0 and p_1, respectively.

Since rows 0 and i are D-minimal and the displayed points are the first disagreements points of their rows in D (as required by Conflict 2), the displayed points are the first disagreement points of their rows in the disagreement set of the two labelings. Therefore, the only possible column that can contain p_0 is the column that is separated by exactly d columns from the column containing the leftmost disagreement point in row 0. Also, the only possible columns that can contain p_1 are the columns that are separated by exactly d or exactly $(d+1)$ columns from the column containing the leftmost disagreement point in row 1. However, both of these positions for p_1 cause a conflict with the d constraint, since p_0 would be either vertically or diagonally adjacent to p_1. Therefore this arrangement causes a conflict with the d constraint.

Similar arguments show the lemma in cases displayed in diagramss (a)–(f) in Figure 2.15.

– Conflict 3.

This arrangement of points causes a conflict since the first disagreement points of opposite color of a D-minimal file in a disagreement subset D can be separated by at most 2 positions, by Lemma 2.B.3.

■

Example 1. Configuration 3.10 in Appendix 2.A.
The four points of r and b that are involved in the conflict are labeled with white dots. They are arranged according to Conflict 2 described in Lemma 2.B.5, so the arrangement shown in this configuration causes at least one of r and b to violate the hexagonal (d, k) constraint.

Example 2. Configuration 3.25 in Appendix 2.A.
This configuration does not contain a commonly occurring conflict, and so we treat it as a special conflict in Lemma 2.B.6.

2.B.2 Lemmas for Main Result in Section 2.5

Lemma 2.B.6. *Let r and b be distinct labelings of a square that satisfy the hexagonal $(d, d+3)$ constraint with $d \geq 7$ and agree on the square's frame of width $k+1$. Then all disagreement subsets shown as Special Conflicts in Figure 2.13 cause at least one of r or b to violate the hexagonal $(d, d+3)$ constraint.*

Proof. In each Special Conflict diagram in Figure 2.13, let the row directly below the square outline be row i (rows decrease moving downward).

Special Conflict 1. Under the given assumptions, there are two possible valid labelings of the circles in the critical columns, and also two possible valid labelings of the circles in the critical diagonals. The green circles denote positions labeled 0 in all possible labelings,

For the circles in critical columns, either orange circles are labeled 0 and blue circles are labeled 1, or vice versa. The same property holds for critical diagonals. However, the labeling associated with orange column circles does not have to agree with the labeling associated with orange diagonal circles (similarly for blue).

To demonstrate that there is a conflict caused by this disagreement subset, we show that all four pairings of these orange and blue arrangements of positions labeled by 1s generate a conflict.

- *(Diagonal Orange = 1, Column Orange = 1)* The 1 in row $(i+1)$ has a run of $(d+5)$ positions labeled 0 to the right. This violates the $k = d + 3$ constraint.
- *(Diagonal Orange = 1, Column Blue = 1)* The two 1s in row i are separated by $(d-1)$ positions. This causes a conflict with the d constraint.
- *(Diagonal Blue = 1, Column Orange = 1)* The two 1s in row $(i-1)$ are separated by $(d+4)$ positions labeled 0. This causes a conflict with the $k = d + 3$ constraint.
- *(Diagonal Blue = 1, Column Blue = 1)* The 1 in row $(i+1)$ has a run of $(d+5)$ positions labeled 0 to the left. This violates the $k = d + 3$ constraint.

Special Conflict 2. The two 1s in row i (i.e., the two orange circles immediately below the square outline) are separated by a distance of $(d-1)$. This causes a conflict with the d constraint.

Special Conflict 3. The 1 in row $(i+1)$ has a run of $(d+5)$ positions labeled 0 to the right. This violates the $k = d + 3$ constraint.

Special Conflict 4. The 1 in row $(i-2)$ has a run of $(d+4)$ positions labeled 0 to the right. This violates the $k = d + 3$ constraint.

Special Conflict 5. The 1 in row $(i+2)$ has a run of $(d+4)$ positions labeled 0 to the right. This violates the $k = d + 3$ constraint.

Special Conflict 6. Diagonals -1 and -3 each contain a single position that must be labeled 1, denoted by hollow circles. If one were to try labeling these positions with 1s, then three of the hollow circles below (in diagonals 0 and 1) would be labeled with 0s, by the d constraint. But then diagonals 0 and 1 would each contain a

single remaining position that must be labeled 1, and these positions are vertically adjacent. Labeling both of these positions by a 1 would therefore cause a conflict with the d constraint.

Special Conflict 7. The two 1s in row i are separated by a distance of $(d-1)$. This causes a conflict with the d constraint.

Special Conflict 8. The two 1s in row $(i-1)$ are separated by a distance of $(d-1)$. This causes a conflict with the d constraint.

Special Conflict 9. The two 1s in row i are separated by a distance of $(d-3)$. This causes a conflict with the d constraint.

■

Definition 2.B.7. We say that a pseudo-origin $x \in D_0$ has the *row minimality property* if for any $D \subseteq D_0$ in the search tree T with root x, every critical file for D is D-minimal.

Lemma 2.B.8. *Let r and b be distinct labelings of a square that satisfy the hexagonal $(d, d+3)$ constraint with $d \geq 7$ and agree on the square's frame of width $k+1$. Suppose there exists a pseudo-origin x in the disagreement set of r and b with the row minimality property. Then at least one of r and b conflicts with the constraint.*

Proof. We traverse the unique path (determined by the disagreement set) of the search tree T with root x. Since x has the row minimality property, every critical file for each disagreement subset in the search tree T with root x is minimal for that disagreement subset.

At each non-leaf node, we choose one particular file and consider the six possible cases (i.e., $\Delta = \pm 1, \pm 2, \pm 3$) required by Lemma 2.B.3. Note that the file chosen for each node is displayed above the node in the tree diagrams in Figures 2.4–2.7, and the disagreement subsets corresponding to the nodes are shown in Appendix 2.A. Certain values of Δ corresponding to relatively easy conflicts are shown in boxes above the nodes, but for the remaining values of Δ, out-edges are shown leading to other nodes in the tree.

At six particular leaf nodes (namely, 1.21, 2.12, 2.30, 2.32, 3.7, 3.24), we choose a file and show that all six possible values of Δ lead to conflicts. At the remaining leaf nodes, we establish a conflict in the given disagreement subset by using the disagreement points in more than one file. Thus, in any case, all leaf nodes in the tree T lead to contradictions with the hexagonal $(d, d+3)$ constraint, and therefore the original assumption of positive capacity cannot be true.

The conflicts established at the leaf nodes rely on the fact that all files in Table 2.2 for a disagreement subset D are D-minimal, since the pseudo-origin x has the row minimality property. The examination of the six possible Δ values in a given file at tree nodes also relies on pseudo-origin x having the row minimality property, since D-minimality of the file is required in Lemma 2.B.3.

Section 2.3.3 describes the steps used to exhaustively build these search trees in more detail, and Subsection 2.B.1 describes the verification of contradictions at the tree leaves, as proven in Lemma 2.B.5 and Lemma 2.B.6. In particular, Lemma 2.B.6 verifies the contradictions to the hexagonal (d, k) constraint for the 7 special conflicts of tree T_1, the one special conflict of T_2, and the one special conflict of T_3. ■

The hypothesis of Lemma 2.B.8 assumes the existence of a pseudo-origin $x \in D_0$ with the row minimality property. The following lemmas establish that such a point x indeed exists. They rely on the previously stated fact that the values of the indices of the rows, columns, and diagonals that are critical for some disagreement subset in the search tree T are upper bounded by 2, 4, and 4, respectively.

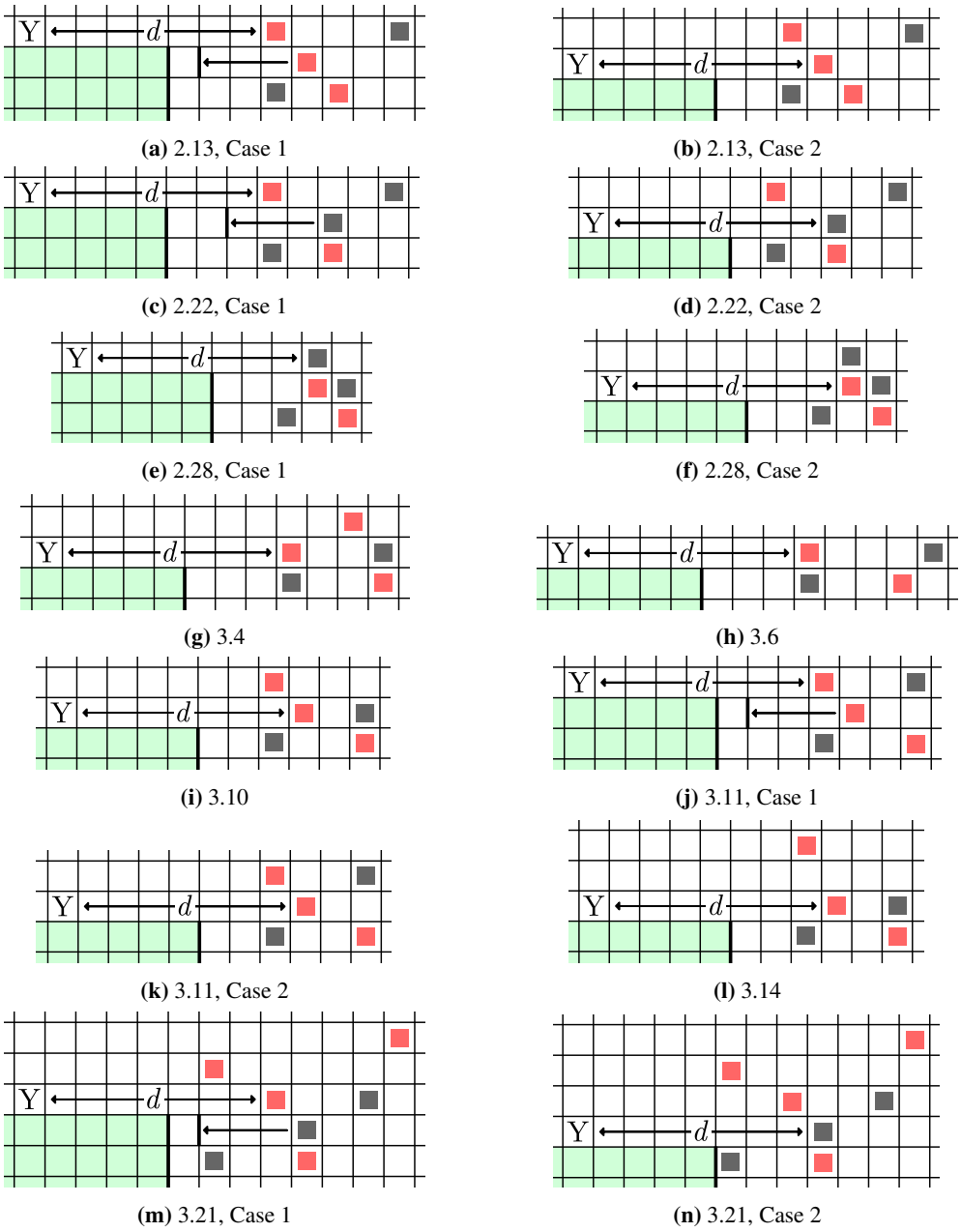
Lemma 2.B.9. *Let r and b be distinct labelings of an $N \times N$ square with disagreement set D_0 that satisfy the hexagonal $(d, d+3)$ constraint, with $d \geq 7$, and which agree on the square's frame of width $k+1$. Let c_m be the column index of the leftmost point of D_0 in row m. Let $D \subseteq D_0$ be a disagreement subset in the search tree T whose root is a pseudo-origin. Suppose row j is critical for D, but is not D-minimal, and let (i, j) be the leftmost point of row j in D. Then the leftmost point of row j in D_0 is a pseudo-origin if $i - 4 < c_m$ whenever $0 \leq m \leq j-1$.*

Proof. By Lemma 2.B.4 there exist both red and black points in row j in D_0 to the left of the point (i, j). Let y be the leftmost point of row j in D_0, and let z be the leftmost point of the other color between y and (i, j). If y and (i, j) are colored the same, then the column index of y satisfies $c_j \leq i - d - 1$ by the d constraint. If y and (i, j) are colored differently, then the column index of z is less than or equal to $i - d - 1$ by the d constraint. Therefore, in either case, since c_j is less than the column index of z, we have $c_j \leq i - d - 1 \leq i - 8$.

Since y is the leftmost disagreement point of row j in D_0, the two labelings agree at all points to the left of y in row j. Whenever $0 \leq m \leq j-1$, the integer c_m is the column index of the leftmost point of row m in D_0, and so the two labelings agree at all points in each row m to the left of column c_m. Therefore, since $c_j + 4 \leq i - 4 < c_m$ whenever $0 \leq m \leq j-1$, the two labelings agree at points with row index m, for $0 \leq m \leq j-1$, and with column or diagonal index less than or equal to $c_j + 4$. The two labelings agree in these columns and diagonals at all points below row 0 as well, since the lowest-left point of D is a pseudo-origin. Therefore, y is a pseudo-origin. ■

The following lemma shows that for any $x \in D_0$ that is a pseudo-origin without the row minimality property, there exists another pseudo-origin that is above and to the left of x in D_0. This property will be exploited in an inductive argument in Lemma 2.B.11.

Lemma 2.B.10. *Let r and b be distinct labelings of an $N \times N$ square with disagreement set D_0 that satisfy the hexagonal $(d, d+3)$ constraint, with $d \geq 7$, and which agree on the square's frame of width $k+1$. Let $x \in D_0$*



(a) 2.13, Case 1 **(b)** 2.13, Case 2

(c) 2.22, Case 1 **(d)** 2.22, Case 2

(e) 2.28, Case 1 **(f)** 2.28, Case 2

(g) 3.4 **(h)** 3.6

(i) 3.10 **(j)** 3.11, Case 1

(k) 3.11, Case 2 **(l)** 3.14

(m) 3.21, Case 1 **(n)** 3.21, Case 2

Figure 2.16. Images depicting the cases in Lemma 2.B.10. The point labeled Y is the rightmost point of its row that could be a pseudo-origin. The green area indicates the region where the labelings agree, which shows that the point labeled Y satisfies the requirements of being a pseudo-origin (as long as the labelings agree at any point to the left of the point labeled Y). The value of d used in the figures is 7, but any larger value of d would push the point labeled Y even farther to the left.

be a pseudo-origin without the row minimality property. Then there exists a pseudo-origin in D_0 that is above and to the left of x.

Proof. For any row m, let c_m be the column index of the leftmost point of row m in D_0. By Lemma 2.5.2, since x is a pseudo-origin, row 0 and any critical columns and diagonals in any $D \subseteq D_0$ in the search tree T with root x are D-minimal. Therefore, since x does not have the row minimality property, there exists $D \subseteq D_0$ in the search tree T with root x but for which there exists a critical row that is not D-minimal. The nodes in the search tree where rows are critical (as opposed to columns or diagonals) are cases 2.13, 2.22, 3.4, 3.6, 3.10, 3.11, 3.14, 3.21, and in the expansion from 2.28 (in Appendix 2.A).

In each of the subsets $D \subseteq D_0$ in the following itemized cases, row 0 is D-minimal because the lowest-left point of D is a pseudo-origin, and so the labelings r and b agree to the left of that point. Therefore, by Lemma 2.B.9, to show a row $j > 0$ has a pseudo-origin, it suffices to check that $i - 4 < c_m$, whenever $0 \leq m \leq j - 1$, where i is the column index of the leftmost point of row j in D.

Figure 2.16 can be used for visualization in the following cases.

- **Configurations 2.13 and 2.22**

  Let D be the disagreement subset in one of configurations 2.13 or 2.22. The critical row for D is row 2, so suppose row 2 is not D-minimal. There are two cases to consider: row 1 is D-minimal, or row 1 is not D-minimal.

  - *Case 1*

    If row 1 is D-minimal, then the point in row 1 (i.e., at position (1,1)) in D is one of the leftmost two points of row 1 in D_0. So either this point is the leftmost disagreement point of row 1, or the leftmost disagreement point of row 1 is at most 3 positions to the left of this point. In either situation (and in either choice of configuration), $c_1 \geq -2$. The column index i of the leftmost point of row 2 in D is 0, and $c_0 = 0$. So $i - 4 = -4 < -2 \leq c_1$ and $i - 4 < c_0$, and so the leftmost point of row 2 in D_0 is a pseudo-origin by Lemma 2.B.9.

  - *Case 2*

    Alternatively, suppose row 1 is not D-minimal. The column index i of the leftmost point of row 1 in D (in either choice of configuration) is at most 2, and $c_0 = 0$. So $i - 4 \leq -2 < 0 = c_0$, and so the leftmost point of row 1 in D_0 is a pseudo-origin by Lemma 2.B.9.

- **Expansion from configuration 2.28**

  When we add children to the search tree from 2.28 by expanding on row 2, we are assuming that row 2 is

D-minimal. So suppose row 2 is not D-minimal. There are two cases to consider: row 1 is D-minimal, or row 1 is not D-minimal.

- *Case 1*

  If row 1 is D-minimal, then the shown points in row 1 constitute the leftmost two points of row 1 in D_0. Therefore, $c_1 = 1$. The column index i of the leftmost point of row 2 in D is 1, and $c_0 = 0$. So $i - 4 = -3 < 1 = c_1$ and thus $i - 4 < c_0$, and so the leftmost point of row 2 in D_0 is a pseudo-origin by Lemma 2.B.9.

- *Case 2*

  Alternatively, suppose row 1 is not D-minimal. The column index i of the leftmost point of row 1 in D is 1, and $c_0 = 0$. So $i - 4 = -3 < 0 = c_0$, and so the leftmost point of row 1 in D_0 is a pseudo-origin by Lemma 2.B.9.

- **Configurations 3.4, 3.10, and 3.14**

  Let D be the disagreement subset in one of configurations 3.4, 3.10, or 3.14. The critical rows for D are row 0 and row 1. Row 0 is D-minimal, so suppose row 1 is not D-minimal. The column index i of the leftmost point of row 1 in D is at most 1 in any of the three configurations, and $c_0 = 0$. So $i - 4 \leq -3 < 0 = c_0$, and so the leftmost point of row 1 in D_0 is a pseudo-origin by Lemma 2.B.9.

- **Configuration 3.6**

  Let D be the disagreement subset in configuration 3.6. The only critical row for D is row 1, so suppose row 1 is not D-minimal. The column index i of the leftmost point of row 1 in D is 0, and $c_0 = 0$. So $i - 4 = -4 < 0 = c_0$, and so the leftmost point of row 1 in D_0 is a pseudo-origin by Lemma 2.B.9.

- **Configuration 3.11**

  Let D be the disagreement subset in configuration 3.11. The critical rows for D are rows 0 and 2. Row 0 is D-minimal, so suppose row 2 is not D-minimal. There are two cases to consider: row 1 is D-minimal, or row 1 is not D-minimal.

  - *Case 1.*

    If row 1 is D-minimal, then the point in row 1 in D is one of the leftmost two points in row 1 in D_0. So either this point is the leftmost disagreement point of row 1, or the leftmost disagreement point of row 1 is at most 3 positions to the left of this point in row 1. In either situation, $c_1 \geq -2$. The column index i of the leftmost point of row 2 in D is 0, and $c_0 = 0$. So $i - 4 = -4 < -2 \leq c_1$ and $i - 4 < c_0$, and so the leftmost point of row 2 in D_0 is a pseudo-origin by Lemma 2.B.9.

– *Case 2*.

Alternatively, suppose row 1 is not D-minimal. The column index i of the leftmost point of row 1 in D is 1, and $c_0 = 0$. So $i - 4 = -3 < 0 = c_0$, and so the leftmost point of row 1 in D_0 is a pseudo-origin by Lemma 2.B.9.

- **Configuration 3.21**

Let D be the disagreement subset in configuration 3.21. The critical rows for D are rows 0 and 2. Row 0 is D-minimal, so suppose row 2 is not D-minimal. There are two cases to consider: row 1 is D-minimal, or row 1 is not D-minimal.

– *Case 1*.

If row 1 is D-minimal, then the point in row 1 in D is one of the leftmost two points in row 1 in D_0. So either this point is the leftmost disagreement point of row 1, or the leftmost disagreement point of row 1 is at most 3 positions to the left of this point. In either situation, $c_1 \geq 0$. The column index i of the leftmost point of row 2 in D is 2, and $c_0 = 0$. So $i - 4 = -2 < c_0 \leq c_1$, and so the leftmost point of row 2 in D_0 is a pseudo-origin by Lemma 2.B.9.

– *Case 2*.

Alternatively, suppose row 1 is not D-minimal. The column index i of the leftmost point of row 1 in D is 3, and $c_0 = 0$. So $i - 4 = -1 < 0 = c_0$, and so the leftmost point of row 1 in D_0 is a pseudo-origin by Lemma 2.B.9.

■

Lemma 2.B.11. *Let r and b be distinct labelings of an $N \times N$ square with disagreement set D_0 that satisfy the hexagonal $(d, d+3)$ constraint, with $d \geq 7$, and which agree on the square's frame of width $k+1$. Then there exists a pseudo-origin in D_0 with the row minimality property.*

Proof. Let x be the lowest-left point of D_0, and without loss of generality suppose its color is black and that it is located at position $(0, 0)$. Clearly x is a pseudo-origin since x is the lowest-left point of D_0.

Suppose x does not have the row minimality property. Then by Lemma 2.B.10, there exists another pseudo-origin that is above and to the left of x in D_0. If this process is repeated, then either a pseudo-origin with the row minimality property will be found in a finite number of steps (since D_0 has height less than N), or else the highest row containing a pseudo-origin of D_0 will be reached. The pseudo-origin with greatest row index must have the row minimality property, or else Lemma 2.B.10 would show the existence of another pseudo-origin in a higher row, a contradiction. ■

Chapter 2 is a reprint of the material as it appears in: S. Congero and K. Zeger, "Hexagonal run-length zero capacity region—Part I: Analytical proofs", *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 130-152, January 2022.

References

[1] R. J. Baxter, "Hard hexagons: exact solution" *Journal of Physics A* vol. 13, pp. 1023 – 1030, 1980.

[2] R. J. Baxter, *Exactly Solved Models in Statistical Mechanics*, Academic Press, 1982.

[3] R. J. Baxter, "Planar lattice gases with nearest-neighbour exclusion," *Annals Combinatorics 3*, vol. 191, pp. 191 – 203, 1999.

[4] R.J. Baxter and S.K. Tsang, "Entropy of hard hexagons," *J. Phys. A (Math. Gen)* vol. 13, pp. 1023 – 1030, 1980.

[5] N. J. Calkin and H. S. Wilf, "The number of independent sets in a grid graph," *SIAM Journal of Discrete Math*, vol. 11, pp. 54 – 60, February 1998.

[6] K. Censor and T. Etzion, "The positive capacity region of two-dimensional run-length-constrained channels," *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 5128 – 5140, November 2006.

[7] J.-Y. Chen, Y.-J. Chen, W.-G. Hu, and S.-S. Lin, "Spatial chaos of Wang tiles with two symbols." *Journal of Mathematical Physics*, vol. 57, no. 2, February 2016.

[8] S. Congero and K. Zeger, "Hexagonal run-length zero capacity region—Part II: Automated proofs," *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 153-177, January 2022.

[9] B. Durand, G. Gamard, and A. Grandjean, "Aperiodic tilings and entropy," *International Conference on Developments in Language Theory,* pp. 166 – 177, Springer, 2014.

[10] O. Elishco, T. Meyerovitch, and M. Schwartz, "Semiconstrained systems," *IEEE Transactions on Information Theory*, vol. 62, no. 4, pp. 1688 – 1702, April 2016.

[11] O. Elishco, T. Meyerovitch, and M. Schwartz, "On encoding semiconstrained systems," *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 2474 – 2484, April 2018.

[12] O. Elishco, T. Meyerovitch, and M. Schwartz, "On independence and capacity of multidimensional semi-constrained systems," *IEEE Transactions on Information Theory*, vol. 64, no. 10, pp. 6461 – 6483, October 2018.

[13] S. Forchhammer and T. V. Laursen, "A model for the two-dimensional no isolated bit constraint," *Proceedings of the IEEE International Symposium on Information Theory*, Seattle, WA, pp. 1189 – 1193, July 2006.

[14] K. S. Immink, *Codes for Mass Data Storage Systems*, second edition, Shannon Foundation Publishers, Eindhoven, The Netherlands, 2004.

[15] G. S. Joyce, "On the hard hexagon model and the theory of modular functions," *Philosophical Transactions of the Royal Society of London A*, vol. 325, pp. 643 – 702, 1988.

[16] G. S. Joyce, "Exact results for the activity and isothermal compressibility of the hard-hexagon model," *Journal of Physics A: Mathematical and General*, vol. 21, pp. L983 – L988, 1988.

[17] P. W. Kasteleyn, "The statistics of dimers on a lattice. I. The number of dimer arrangements on a quadratic lattice," *Physica*, vol. 27, pp. 1209 – 1225, 1961.

[18] A. Kato and K. Zeger, "On the capacity of two-dimensional run length constrained channels," *IEEE Transactions on Information Theory*, vol. 45, no. 4, pp. 1527 – 1540, July 1999.

[19] Zs. Kukorelly and K. Zeger, "The capacity of some hexagonal (d, k) constraints", *IEEE International Symposium on Information Theory (ISIT)*, Washington, D.C., p. 64, June 2001

[20] Zs. Kukorelly and K. Zeger, "Automated theorem proving for hexagonal run length constrained capacity computation", *IEEE International Symposium on Information Theory (ISIT)*, Seattle, Washington, pp. 1199 – 1203, July 2006.

[21] E. Louidor and B. H. Marcus, "Improved lower bounds on capacities of symmetric 2D constraints using Rayleigh quotients," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1624 – 1639, April 2010.

[22] B. Marcus, "Capacity of higher dimensional constrained systems", In *Coding Theory and Applications* vol. 3, pp 3 – 21, edited by R. Pinto, P. R. Malonek, and P. Vettori, Springer, 2015.

[23] B. Marcus and R. Pavlov, "Approximating entropy for a class of $\mathbb{Z}^2$ Markov random fields and pressure for a class of functions on $\mathbb{Z}^2$ shifts of finite type," *Ergodic Theory and Dynamical Systems*, vol. 33, no. 1, pp. 186 – 220, 2013.

[24] B. Marcus and R. Pavlov, "Computing bounds for entropy of stationary $\mathbb{Z}^d$ Markov random fields," *SIAM Journal on Discrete Mathematics*, vol. 27, no. 3, pp. 1544 – 1558, 2013.

[25] B.D. Metcalf and C.P. Yang, "Degeneracy of anti-ferromagnetic Ising lattices at critical magnetic field and zero temperature," *Physical Review B* vol. 18, pp. 2304 – 2307, 1978.

[26] Zs. Nagy and K. Zeger, "Capacity bounds for the three-dimensional run length limited channel," *IEEE Transactions on Information Theory*, vol. 46, pp. 1030 – 1033, May 2000.

[27] L. Onsager, "Crystal statistics. I. A two-dimensional model with an order-disorder transition," *Physical Review*, vol. 65, nos. 3 and 4, pp. 117 – 149, 1944.

[28] R. Pavlov, "Approximating the hard square entropy constant with probabilistic methods," *Annals of Probability*, vol. 40, no. 6, pp. 2362 – 2399, 2012.

[29] M. Schwartz and J. Bruck, "On the capacity of the precision-resolution system," *IEEE Transactions on Information Theory*, vol. 56, no. 3, pp. 1028 – 1037, March 2010.

[30] M. Schwartz and A. Vardy, "New bounds on the capacity of multidimensional run-length constraints," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4373 – 4382, July 2011.

[31] A. Sharov and R. M. Roth, "Two-dimensional constrained coding based on tiling," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1800 – 1807, April 2010.

[32] H. Wang, "Proving theorems by pattern recognition – II", *The Bell System Technical Journal*, vol. 40, no. 1, pp. 1 – 41, January 1961.

[33] G.H. Wannier, "Antiferromagnetism. The triangular Ising net," *Physical Review*, vol. 79, no. 2, pp. 357 – 364, 1950.

[34] W. Weeks and R. E. Blahut, "The capacity and coding gain of certain checkerboard codes", *IEEE Transactions on Information Theory*, vol. 44, pp. 1193 – 1203, May 1998.

Chapter 3

Hexagonal Run-Length Zero Capacity Region, Part II: Automated Proofs

Abstract

The zero capacity region for hexagonal (d, k) run-length constraints is known for many, but not all, d and k. The pairs (d, k) for which it has been unproven whether the capacity is zero or positive consist of: (i) $k = d + 2$ when $d \geq 2$; (ii) $k = d + 3$ when $d \geq 1$; (iii) $k = d + 4$ when either $d = 4$ or d is odd and $d \geq 3$; and (iv) $k = d + 5$ when $d = 4$. Here, we prove the capacity is zero in case (i) when $2 \leq d \leq 9$, in case (ii) when $3 \leq d \leq 11$, and in case (iii) when $d \in \{4, 5, 7, 9\}$. We also prove the capacity is positive in case (ii) when $d \in \{1, 2\}$, in case (iii) when $d = 3$, and in case (iv). The zero capacities for $k = d + 4$ are the first and only known cases equal to zero when $k - d > 3$. All of our results are obtained by developing three algorithms that automatically and rigorously assist in proving either the zero or positive capacity results by efficiently searching large numbers of configurations. The proofs involve either upper bounding the number of paths through certain large directed graphs, finding forbidden strings, or building distinct tileable square labelings. Some of the proofs examine over 20 billion cases using a supercomputer. In Part I of this two-part series, it is proven that the capacity is zero for all of case (i), and for case (ii) whenever $d \geq 7$. Thus, the only remaining unknown cases are now when $k = d + 4$, for any odd $d \geq 11$.

3.1 Introduction

This paper is Part II of a two-part series. Some of the background, motivation, and basic definitions will be repeated here so that the presentation can be followed in a self-contained manner. The reader is referred to Part I [7] for other details.

A one-dimensional run-length constraint imposes both lower and upper bounds on the number of zeros that occur between consecutive ones in a binary string. Specifically, if d and k are nonnegative integers, or ∞, then a binary string is said to satisfy a (d, k) constraint if every consecutive pair of ones in the string has at least d zeros between them and the string never has more than k zeros in a row. We will assume throughout that $k < \infty$. It is known that if $k > d$, then the number of (one-dimensional) N-bit binary strings that satisfy the (d, k) constraint grows exponentially in N (e.g., [9]) and that the logarithm (base two) of that number, divided by N, approaches a positive limit as N grows to infinity. This limit is known as the "capacity" of the constraint.

The concepts of (d, k) constraints and capacities have been generalized to two dimensions, where the one-dimensional (d, k) constraint is imposed both vertically and horizontally. Sometimes these two-dimensional constraints are referred to as "rectangular constraints". To determine the capacity of a rectangular constraint, one counts the number of binary labelings of an $N \times N$ square that satisfy the constraint, takes its logarithm, and then divides by the area N^2 of the square. It is known that this quantity approaches a limit $C_{\text{rect}}(d, k)$ (called the "capacity" again) as N grows to infinity (e.g., [12]).

The *zero capacity region* for a particular type of constraint is the set of all pairs (d, k) for which the (d, k) capacity equals zero. If a particular constraint has zero capacity, then the number of valid labelings of a region does not grow exponentially fast in terms of the volume (e.g., length for 1 dimension, area for 2 dimensions, etc.) of the region.

Various estimates of the two-dimensional rectangular capacity were obtained for the particular case $C_{\text{rect}}(1, \infty) \approx 0.587891162$ by Calkin and Wilf [5], Weeks and Blahut [23], Baxter [3], Pavlov [18], and in [16]. This rectangular $(1, \infty)$ constraint is sometimes referred to as the "hard square model" by physicists [2], and its capacity is known to equal the rectangular capacity $C_{\text{rect}}(0, 1)$.

For two-dimensional rectangular (d, k) constraints, the zero capacity region was completely characterized in [12], where it was shown that the capacity satisfies $C_{\text{rect}}(d, k) > 0$ if and only if $k \geq d + 2$, when $d \geq 1$ (i.e., $C_{\text{rect}}(d, k) = 0$ when $k = d + 1$). It is also known that $C_{\text{rect}}(0, k) > 0$ and $C_{\text{rect}}(k, \infty) > 0$ for all $k \geq 1$. Bounds on the two-dimensional rectangular (d, k) capacity were given in [12], by Sharov and Roth in [20], and were later improved and generalized to higher dimensions by Schwartz and Vardy in [19].

A two-dimensional "hexagonal" constraint imposes one-dimensional (d, k) constraints along the 3 pri-

mary directions on a hexagonal lattice. An equivalent way to view the hexagonal constraint on a rectangular lattice is to impose the (d, k) constraint both horizontally and vertically, and also along one of the two diagonal directions (we will use the northeast-southwest direction, but refer to it as the "northeast diagonal") [2, p. 409].

The hexagonal (d, k) capacity, denoted by $C_{\text{hex}}(d, k)$, is the limit as $N \to \infty$ of the logarithm base two of the number of $N \times N$ squares satisfying the hexagonal (d, k) constraint, divided by the area N^2 of the square.

The hexagonal (d, k) capacity $C_{\text{hex}}(d, k)$ is known to be positive for certain pairs (d, k). In fact, if $C_{\text{hex}}(d, k) > 0$, then it immediately follows that $C_{\text{hex}}(d', k') > 0$ whenever either $d' < d$ or $k' > k$ (or both), since the constraints weaken in either instance. Positive lower bounds on the hexagonal (d, k) capacity were previously proven for $d = 0$, and for all values of $d \geq 5$ for sufficiently large k (for example, $k = d + 5$ suffices), and now also for $1 \leq d \leq 4$ with our results in this paper. In what follows, we will summarize, for each $d > 0$, the smallest known k such that $C_{\text{hex}}(d, k) > 0$.

The only exactly known non-zero capacity of a hexagonal (d, k) constraint is for the case $(1, \infty)$, which is known in the physics literature as the "hard hexagon model". As with the rectangular constraint, it is easy to show that the hexagonal $(0, 1)$ and $(1, \infty)$ capacities are the same, by reversing the roles of 0s and 1s. The problem of counting the number of patterns in a bounded area that satisfy the hexagonal $(1, \infty)$ constraint was considered in the context of Ising models in physics, by Onsager [17], and Wannier [22]. An equivalent problem is to find the number of configurations of non-attacking kings on a chessboard with regular hexagonal cells.

Metcalf and Yang [15] conjectured that the capacity of the hexagonal $(1, \infty)$ constraint was $\log_2 e^{1/3} \approx 0.48090$, but this was disproven by Baxter and Tsang [4], who obtained a slightly more accurate estimate. Baxter [1,2], and Joyce [10,11] performed numerous intricate calculations, which when combined determine the exact hexagonal $(1, \infty)$ capacity.

Using the technique of finding two distinct tileable squares, Censor and Etzion [6] proved that $C_{\text{hex}}(d, d + 4) > 0$ for all even $d \geq 6$. An immediate consequence is that $C_{\text{hex}}(d, d + 5) > 0$ for all odd $d \geq 5$, since the hexagonal $(d, d + 5)$ constraint is weaker than the $(d + 1, d + 5)$ constraint. In this paper, we present a tiling algorithm that automatically generates distinct tileable square labelings that demonstrate positive hexagonal (d, k) capacities for certain pairs (d, k). In particular, we prove that the capacities $C_{\text{hex}}(1, 4)$, $C_{\text{hex}}(2, 5)$, $C_{\text{hex}}(3, 7)$, and $C_{\text{hex}}(4, 9)$ are all positive.

Also, we note that the positive hexagonal (d, k) capacities obtained in [6] were for the case of $k = d + 4$ when d is even and $d \geq 6$, but the proof technique does not apply to odd $d \geq 5$. In this paper, in contrast to even $d \geq 6$, we show that some of the open cases with $k = d + 4$ when d is odd have zero capacity.

Whether or not $C_{\text{hex}}(d, k)$ is positive or zero has been unproven[1] for the following cases:

[1] Some of these cases were stated in [13] and [14] and are included here for archival purposes.

(i) $k = d + 2$ when $d \geq 2$

(ii) $k = d + 3$ when $d \geq 1$

(iii) $k = d + 4$ when either $d = 4$, or d is odd and $d \geq 3$

(iv) $k = d + 5$ when $d = 4$.

Among these open cases, we prove in Part I that the hexagonal (d, k) capacity equals zero in all of case (i), and in case (ii) whenever $d \geq 7$. Here, in Part II, we prove that the capacity is zero in case (i) when $2 \leq d \leq 9$ and case (ii) when $3 \leq d \leq 11$ (in Corollary 3.2.11 and Corollary 3.3.4), and in case (iii) when $d \in \{4, 5, 7, 9\}$ (in Corollary 3.2.11), and that the capacity is positive in case (ii) when $d \in \{1, 2\}$, in case (iii) when $d = 3$, and in case (iv) (in Theorem 3.4.1).

Table 3.1 summarizes the present knowledge of the zero capacity region when d is less than 19 and k is less than 25, including the results we present in Parts I and II of these papers. The results from Part I are shown surrounded by squares and the results from Part II are shown surrounded by circles. We note that four of the results turn out to be produced by both the methods in Part I and Part II, and we denote them in the table being surrounded by both a circle and a square. Proofs of the results in Part I and Part II have not previously appeared in the literature. We note that although we provide here and in Part I the first published proofs of the cases where $k = d + 2$, those satisfying $d \geq 7$ are not listed as new results in the table, since they directly follow from our stronger (but more complex) $k = d + 3$ proof in Part I.

We also note that no positive capacities in the table were previously proven on any of the four rows corresponding to $d = 1, 2, 3, 4$, but our results in Theorem 3.4.1 imply that the entire row to the right of each of our newly added "+" entries is filled with positive capacities.

One way to demonstrate that a particular hexagonal (d, k) capacity is positive is to exhibit at least two rectangular labelings of the same dimensions that can validly tile the plane in any configuration. For example, for some fixed d and k, suppose A and B are two different $M \times N$ rectangles filled with 0s and 1s that each satisfy the hexagonal (d, k) constraint, and such that $d < k < M \leq N$. If each of the 16 possible assignments of rectangles

1	2
3	4

Figure 3.1. Tiling configuration for demonstrating positive capacities with the Rectangle Tiling Algorithm. If each of two different binary labelings A and B of an $M \times N$ rectangle can arbitrarily occupy each of the four positions shown without violating the hexagonal (d, k) constraint, then the constraint has positive capacity.

A and B to the four possible positions shown in Figure 3.1 causes the resulting $2N \times 2M$ rectangle to satisfy the (d, k) hexagonal constraint, then arbitrary tilings of the plane by rectangles A and B also satisfy the same

Table 3.1. Summary of the known zero hexagonal (d, k) capacity region for small d and k. Zero and positive capacities are denoted by "0" and "+", respectively. The circled symbols denote our contributions in the present paper (Part II) and the zeros in squares are from our Part I paper [7], and those with both squares and circles occurred in both Parts I and II. The question marks are remaining unsolved cases.

$d\backslash k$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
1		0	0	0	⊕	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
2			0	0	⊚	⊕	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
3				0	0	⊚	⊜	⊕	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
4					0	0	⊚	⊜	⊜	⊕	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
5						0	0	⊚	⊜	⊜	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
6							0	0	⊚	⊜	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
7								0	0	0	⊚	⊜	+	+	+	+	+	+	+	+	+	+	+	+	+
8									0	0	0	⊚	+	+	+	+	+	+	+	+	+	+	+	+	+
9										0	0	0	⊚	⊜	+	+	+	+	+	+	+	+	+	+	+
10											0	0	0	⊚	+	+	+	+	+	+	+	+	+	+	+
11												0	0	0	⊚	?	+	+	+	+	+	+	+	+	+
12													0	0	0	□	+	+	+	+	+	+	+	+	+
13														0	0	0	□	?	+	+	+	+	+	+	+
14															0	0	0	□	+	+	+	+	+	+	+
15																0	0	0	□	?	+	+	+	+	+
16																	0	0	0	□	+	+	+	+	+
17																		0	0	0	□	?	+	+	+
18																			0	0	0	□	+	+	+

constraint. In such a case, since the rectangles A and B each have area MN and the four positions can be chosen freely as either A or B, the capacity is at least $1/(MN)$, which is positive.

We create a Rectangle Tiling Algorithm (discussed in Section 3.4) using this technique, and show in Theorem 3.4.1 that $C_{\text{hex}}(1,4)$, $C_{\text{hex}}(2,5)$, $C_{\text{hex}}(3,7)$, and $C_{\text{hex}}(4,9)$ are all positive. For each of these cases, a square is determined that can take on two distinct labelings which can arbitrarily tile the plane without violating the constraint. These squares are depicted in Figures 3.16–3.19, in which a variable x can be set as either 0 or 1 to obtain two distinct tileable squares (note that $\overline{x} = 1 - x$). Similarly, for $d = 0$, while the fact that $C_{\text{hex}}(0,1) > 0$ follows from elaborate derivations of Baxter [1] and Joyce [10,11], a much simpler proof is given in Theorem 3.4.1 using two distinct tileable squares.

Prior to this present paper, for the case of $k = d + 4$, some hexagonal (d, k) capacities were known to be positive but none were known to equal zero. Specifically, it was previously known [**?**, 10, 11] that $C_{\text{hex}}(0,1) > 0$, and it is shown here (Theorem 3.4.1) that $C_{\text{hex}}(1,4) > 0$, $C_{\text{hex}}(2,5) > 0$, and $C_{\text{hex}}(3,7) > 0$, from which it follows that $C_{\text{hex}}(d, d+4) > 0$ whenever $0 \leq d \leq 3$. Additionally, it was proven by Censor and Etzion [6] that $C_{\text{hex}}(d, d+4) > 0$ for all even $d \geq 6$. This knowledge left as open problems whether $C_{\text{hex}}(d, d+4)$ is positive or zero in the $d = 4$ or odd $d \geq 5$ cases.

In our present paper, we demonstrate the first known cases for $k = d + 4$ where the hexagonal capacity is zero, specifically whenever $d \in \{4, 5, 7, 9\}$. The fact that the $k = d + 4$ cases alternate between zero and positive hexagonal capacities from $d = 5$ to $d = 10$ contrasts the case of $k = d + 3$ where we showed (in Part I) that $C_{\text{hex}}(d, d+3) = 0$ for all $d \geq 3$. Also, in the rectangular constraint case, there is no such alternation between zero and positive capacities [12].

Here, in Part II, one of our approaches to proving particular hexagonal (d, k) capacities are zero is to show that the number of valid labelings of an $N \times N$ square grows like $2^{O(N)}$ as $N \to \infty$, whereas one would need $2^{\Omega(N^2)}$ to assure a positive capacity. To accomplish this, we create the Constant Position Algorithm (defined formally in Section 3.2.4), which assists in the proof by showing that a valid hexagonal (d, k) labeling of any $(k + 1)$ consecutive rows in an $N \times N$ square allows only a small number of choices for validly labeling the row immediately above the $(k + 1)$ consecutive rows. The exact number of such valid labelings of that new row is shown to be bounded, as $N \to \infty$. This in turn implies that there are only $2^{O(N)}$ valid labelings of the square as $N \to \infty$, which yields zero capacity. These facts are established by constructing two directed graphs representing allowable sequences of labelings of a $(k + 1) \times (k + 1)$ square that slides horizontally or vertically, one row or column at a time, respectively.

In order to upper bound the number of paths through these two directed graphs, we show that it suffices to determine which pairs of paths through the horizontal graph have vertically compatible vertices. To aid this au-

tomated search, we show the useful fact that there is no loss of generality in restricting paths to strongly connected components of the graphs (Lemma 3.2.8). The computation yields an upper bound on the number of ways that a valid labeling of a $(k+1) \times N$ horizontal strip can be extended upwards by one row, and, consequently, gives an upper bound on the number of valid labelings of an $N \times N$ square.

It is shown that if a certain constant position property, defined in Section 3.2.4, exists for a particular hexagonal (d, k) constraint, then the upper bound is tight enough to make a positive capacity impossible for that constraint (Theorem 3.2.9).

We use our Constant Position Algorithm to efficiently verify if the constant position property holds for the (d, k) cases of concern. The algorithm reduces the complexity of such a search from what we assess to be "virtually impossible" with today's technology, down to "manageable", using a multi-node supercomputer with extensive memory capabilities. When the algorithm was implemented and executed, over 20 billion separate cases were examined, and as a result we directly observed that the constant position property indeed holds for 11 new cases. Namely, assisted by the Constant Position Algorithm, we prove in Corollary 3.2.11 that $C_{\text{hex}}(d, k) = 0$ whenever

$$(d, k) \in \{(3, 6), (4, 7), (4, 8), (5, 8), (5, 9), (6, 9), (7, 10), (7, 11), (8, 11), (9, 12), (9, 13)\}.$$

In fact, we also observed that the constant position property holds more generally for all cases of zero hexagonal (d, k) capacity when $d \leq 9$.

In addition to the Constant Position Algorithm and the Rectangle Tiling Algorithm, we also introduce a third automated proof technique, which we call the Forbidden String Algorithm. The Forbidden String Algorithm, discussed in Section 3.3, proves some hexagonal (d, k) capacities are zero by showing that certain binary strings can never occur in large validly labeled rectangles. It uses the fact that $C_{\text{hex}}(d, k) = C_{\text{hex}}(d+1, k)$ if 10^d1 is forbidden, and $C_{\text{hex}}(d, k) = C_{\text{hex}}(d, k-1)$ if 10^k1 is forbidden.

The emphasis in this paper will be on the Constant Position Algorithm, as this algorithm takes the most effort to describe, and gives the best results. After that, we will discuss the Forbidden String Algorithm and the Rectangle Tiling Algorithm.

Some preliminary definitions are now provided. A *rectangle* is an $M \times N$ two-dimensional array, where M is the number of rows and N is the number of columns. If $M < N$ (respectively, $M > N$), then the rectangle is called a *horizontal strip* (respectively, a *vertical strip*). A *labeling* of a set is an assignment $l(x) \in \{0, 1\}$ to each element x of the set. A labeling l of a rectangle is said to *satisfy the hexagonal* (d, k) *constraint* (or is *valid*) if along every row, column, and northeast diagonal, the number of 0s between any two 1s is at least d, and no run

of 0s is longer than k. A *position* in an $M \times N$ rectangle is a relative location (x, y), indexed by integer-valued Cartesian coordinates. The ith row (respectively, column) of such a rectangle consists of positions with second (respectively, first) coordinate i. For example, the bottom-left element of any rectangle is at position $(1, 1)$ and the top-right (i.e., in row M and column N) element is at position (N, M).

If l is a labeling, then $l(x, y)$ will denote the value of the labeling l at position (x, y).

Let $L(M, N)$ denote the set of valid labelings of an $M \times N$ rectangle. The *capacity* of the hexagonal (d, k) constraint is defined as

$$C_{\text{hex}}(d, k) = \lim_{M,N\to\infty} \frac{\log_2 |L(M, N)|}{MN}$$

and is known to exist for all d and k by subadditivity (e.g., [12]). Note that if $C_{\text{hex}}(d, k) > 0$, then the number of valid labelings is lower bounded as $|L(M, N)| = 2^{\Omega(MN)}$. For the remainder of the paper, it is assumed that d and k are fixed and all logarithms are base 2.

3.2 Constant Position Algorithm for proving zero hexagonal (d, k) capacity

The Constant Position Algorithm is an automated computer search that examines as many as billions of cases in order to assist the rigorous proof of certain zero hexagonal (d, k) capacity cases. Our proof relies on a peculiar property, which we call the constant position property and describe in Section 3.2.4, that turns out to be a sufficient condition for the hexagonal (d, k) capacity to be zero (see Theorem 3.2.9).

The algorithm tries to determine if a given hexagonal (d, k) constraint has zero capacity. The main idea is to check how many valid labelings there are of a horizontal $(k + 1) \times N$ strip, given the (valid) labeling of the horizontal $(k + 1) \times N$ strip immediately below it (i.e., shifted one row down). If the number of such labelings of the upper horizontal strip is small enough, no matter which labeling is used for the lower horizontal strip, then the number of valid labelings of an $N \times N$ square can be upper bounded tightly enough to avoid $2^{\Omega(N^2)}$ growth, thus proving the (d, k) constraint has zero capacity.

In order to efficiently determine if the labelings of such shifted horizontal strips are severely constrained, it is convenient to create a directed graph, whose nodes are valid labelings of a $(k+1) \times (k+1)$ square, and whose directed edges convey whether one square labeling can overlay the other, but shifted one column to the right. We also create an analogous graph for vertical sliding compatibility. We examine labelings corresponding to walks through the constructed (horizontal) graph that do not change from one graph component to another. We prove that this is, in fact, possible to do without sacrificing generality (see Lemma 3.2.8). As a result, the Constant Position Algorithm below focuses on horizontal $(k + 1) \times N$ labelings that each correspond to a walk in a single arbitrary

component of the (horizontal) graph.

We note that as k grows, the number of valid labelings of $(k+1) \times (k+1)$ squares grows rapidly, which makes storage and processing computationally difficult. Thus, the computational complexity of performing the Constant Position Algorithm increases, and at some point becomes infeasible, even with massive computing power (typically when $k \geq 11$). However, for every computationally feasible case, our results establish that $C_{\text{hex}}(d,k) = 0$ if and only if the constant position property holds, We suspect the "only if" direction of this assertion may be true for even more complex cases, but we leave it as an open question in Conjecture 3.2.12.

3.2.1 Definitions

Let Λ be the set of valid labelings of a $(k+1) \times (k+1)$ square. Define two directed graphs

$$G_h = (\Lambda, E_h)$$
$$G_v = (\Lambda, E_v)$$

such that $(x,y) \in E_h$ if and only if the x-labeling of the rightmost k columns of the $(k+1) \times (k+1)$ square is identical to the y-labeling of the leftmost k columns of the square, and such that $(x,y) \in E_v$ if and only if the x-labeling of the topmost k rows of the square is identical to the y-labeling of the bottommost k rows of the square. Any such graphs will be called *label graphs*.

Figure 3.2 illustrates labelings of two 3×3 squares and how they can form the vertices of a single edge in G_h. Thus if there is an edge from x to y in G_h, then the valid labeling x can be extended rightward by one

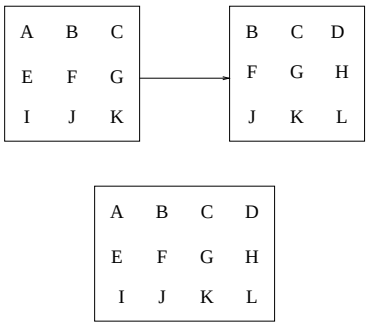


Figure 3.2. Illustration of an edge in G_h. Two 3×3 labeled squares (the vertices) are overlaid to form a 3×4 horizontal strip. The letters represent binary values according to some valid labeling.

column to a valid labeling of a $(k+1) \times (k+2)$ rectangle, using the y-labeling of the rightmost column of the $(k+1) \times (k+1)$ rectangle. Similarly, if there is an edge from x to y in G_v, then the valid x-labeling can be extended upward one row using the y-labeling of the topmost row of the $(k+1) \times (k+1)$ rectangle. In particular,

if (λ_1, λ_2) is an edge in G_v, then labeling $\lambda_2 \in \Lambda$ is said to be *vertically compatible above* labeling $\lambda_1 \in \Lambda$.

The label graph G_h is defined so that a labeling of a $(k+1) \times N$ horizontal strip is valid if and only if the sequence of labelings of $(k+1) \times (k+1)$ squares, obtained by sliding one column at a time from left to right in the strip, is a (directed) walk in the graph G_h. Similarly, the label graph G_v is defined so that a labeling of a $M \times (k+1)$ vertical strip is valid if and only if the sequence of labelings of squares, obtained by sliding one row at a time from bottom to top in the strip, is a (directed) walk in the graph G_v. In these cases, the walk in G_h is said to *correspond to* the labeling of the horizontal strip, and the walk in G_v is said to *correspond to* the labeling of the vertical strip.

Define $L_{G_1,G_2}(M, N)$ to be the set of labelings l of an $M \times N$ rectangle such that the restriction of l to any $(k+1) \times N$ horizontal strip corresponds to a walk through G_1 and the restriction of l to any $M \times (k+1)$ vertical strip corresponds to a walk through G_2. The two label graphs G_1 and G_2 are said to *generate* $L_{G_1,G_2}(M, N)$.

One basic fact that we will repeatedly rely on is that any labeling of an $M \times N$ square is valid if and only if the labeling is valid on every $(k+1) \times (k+1)$ subsquare of the $M \times N$ square. This is due to the fact that any violation of the k constraint (i.e., the existence of a string 0^{k+1}) along a horizontal, vertical, or diagonal would have to occur in a $(k+1) \times (k+1)$ subsquare (as would any violation of the d contraint). A consequence (shown in Lemma 3.2.5) is that $L_{G_h,G_v}(M, N) = L(M, N)$.

A directed graph is called *strongly connected* if there exist directed paths from every vertex to every other vertex in the graph. A strongly connected subgraph K of G is *maximal* if no other strongly connected subgraph of G contains K. Maximal strongly connected subgraphs of G will be referred to as *components*. The components of G partition the vertices of G, but not necessarily the edges. A key property that we will make use of is that any walk through a directed graph that leaves a particular component can never return to that component.

A component is called *cyclic* if it contains a directed cycle. In G_h, any component K is cyclic if and only if a bi-infinite walk through K corresponds to a labeling of a $(k+1) \times \infty$ horizontal strip. In fact, the only non-cyclic components are single vertices without self-loops, and thus the lone vertex in a non-cyclic component can appear in the labeling of a $(k+1) \times N$ horizontal strip at most once. Analogous statements apply to cyclic and non-cyclic components in G_v.

If $d > 0$ and $k < \infty$, then the graphs G_h and G_v cannot contain any self-loops, since the vertex in any self-loop would necessarily contain either an all-0 or all-1 labeling of a row or column of a $(k+1) \times (k+1)$ square. More generally, all cycles in G_h and G_v must have at least $d+1$ edges, and thus all cyclic components of G_h and G_v must contain at least $d+1$ vertices.

The following example illustrates the types of components that can occur in G_h or G_v.

Example 3.2.1. The directed graph in Figure 3.3 has components whose vertex sets are $\{1, 2, 3\}$, $\{4\}$, $\{5, 6\}$,

and $\{7, 8, 9\}$, and whose edge sets consist of all edges that connect vertices within each vertex set. Of these components, all are cyclic except the component whose vertex set is $\{4\}$.

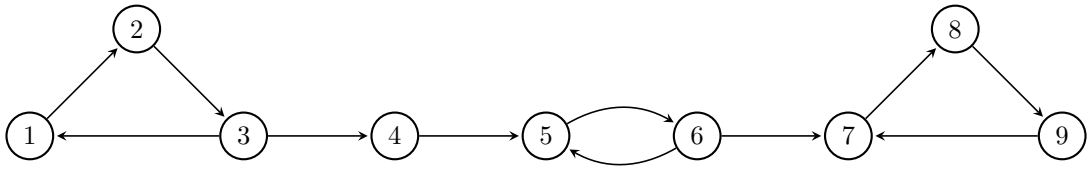


Figure 3.3. Graph used in Example 3.2.1.

In the following example, we use the hexagonal $(3, 4)$ constraint to illustrate the correspondence between walks through the graph G_h and sequences of labelings of $(k+1) \times (k+1)$ squares in $(k+1) \times M$ horizontal strips for an actual valid example labeling. The capacity of the hexagonal $(3, 4)$ constraint is known to be zero (it is implied by the known zero capacity of the rectangular $(3, 4)$ constraint), and it results in a relatively small label graph G_h.

Example 3.2.2. Figure 3.4 illustrates the directed graph G_h for the hexagonal $(3, 4)$ constraint. The graph G_h consists of 7 components: three disjoint 5-cycles, and four isolated vertices. Each vertex is a labeling of a 5×5 square. Figure 3.5 shows an example of a valid labeling of a 15×20 rectangle. It can be seen that the labeling of any horizontal strip of width 5 corresponds to a walk through one of the 3 cycles in the graph.

Let R be the set of all $(k+1) \times (k+1)$ squares in an $M \times N$ rectangle. Each square in R lies entirely in one particular $(k+1) \times N$ horizontal strip within the $M \times N$ rectangle. Each such horizontal strip contains $(N-k)$ squares of size $(k+1) \times (k+1)$, and for any labeling of the $M \times N$ rectangle, the labeling of each such $(k+1) \times (k+1)$ square within the strip belongs to exactly one component of G_h. In fact, when scanning a strip from left to right, if the labeling of a $(k+1) \times (k+1)$ square lies in a particular component, but the labeling of the next square to its right (i.e., sliding one column rightward) in the strip does not lie in the same component, then the former component can never be revisited within that strip.

Given a fixed valid labeling l of an $M \times N$ rectangle, we identify, for each horizontal strip and for each component of G_h, the first (i.e., leftmost) occurrence in the strip of a $(k+1) \times (k+1)$ square labeling from the component. We use the notation $T_h(l)$ to denote the set of all such leftmost squares in all strips of the $M \times N$ rectangle. Specifically, for any $l \in L_{G_h, G_v}(M, N)$, define

$$T_h(l) = \{r \in R : \text{the labeling of each } (k+1) \times (k+1) \text{ square to the left of } r \text{ in the same strip belongs to a different component of } G_h \text{ as the labeling of } r\}.$$

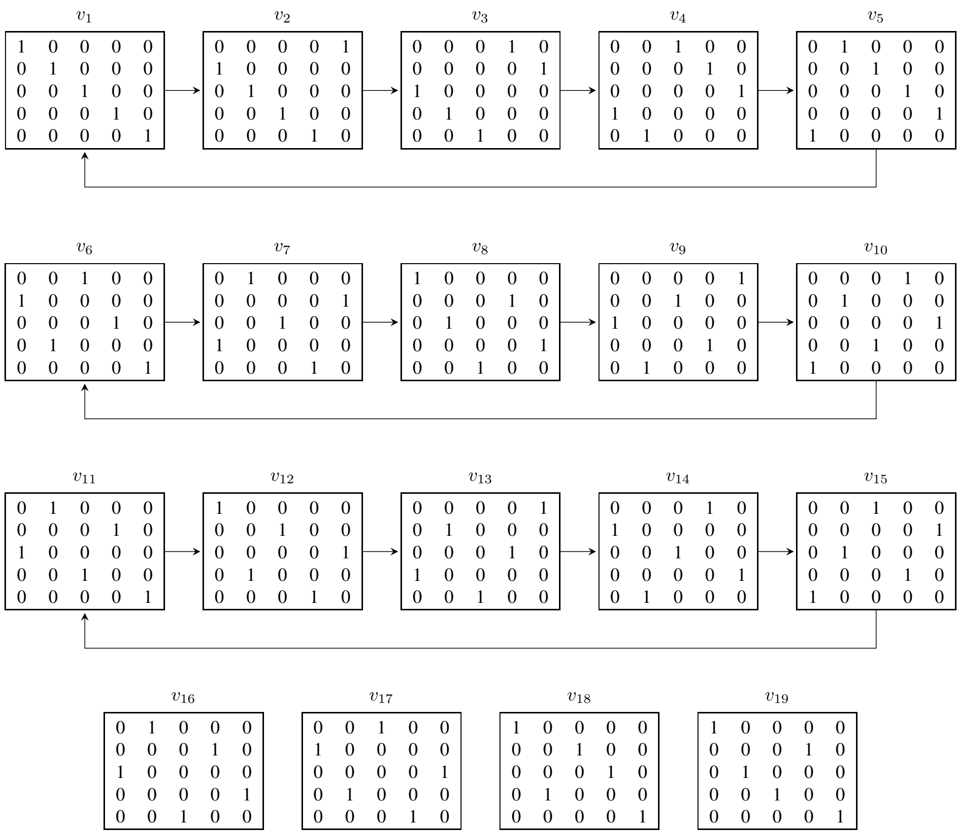


Figure 3.4. The graph G_h for the hexagonal $(3, 4)$ constraint. The graph G_h consists of 3 disjoint 5-cycles and 4 isolated vertices, where each vertex is a valid labeling of a 5×5 square.

0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

Figure 3.5. A labeling of a 15×20 rectangle satisfying the hexagonal $(3, 4)$ constraint. The labeling of each 5×20 horizontal strip in the rectangle corresponds to a walk through the second component of G_h listed in Figure 3.4. For example, the labeling of the horizontal strip consisting of the top 5 rows of the rectangle corresponds to the sequence of 16 vertices that starts at v_6 and continues through the cycle in the second component of G_h, i.e., $v_6, v_7, v_8, v_9, v_{10}, v_6, \ldots$.

Note that every $(k+1) \times (k+1)$ square whose labeling under l is in a non-cyclic component of G_h necessarily lies in $T_h(l)$.

For any fixed labeling $l \in L_{G_h,G_v}(M,N)$, each $(k+1) \times N$ horizontal strip in the $M \times N$ rectangle corresponds to a walk through the graph G_h. Each node in such a walk belongs to exactly one component of G_h. If (a,b) is a directed edge in such a walk and a and b belong to different components of G_h, then the rectangle being labeled by b is in $T_h(l)$. Thus, $T_h(l)$ is the set of all locations of component transitions in G_h for the horizontal strips of the $M \times N$ rectangle.

We define the equivalence relation $\sim$ between labelings $l_1, l_2 \in L_{G_h,G_v}(M,N)$ so that $l_1 \sim l_2$ iff $T_h(l_1) = T_h(l_2)$, and denote the associated equivalence class of any $l \in L_{G_h,G_v}(M,N)$ by $[l]$.

The Constant Position Algorithm is described in Section 3.2.4 in terms of 6 Steps, and in this section we offer a preview in order to convey the gist of how it works. The following definition will be used throughout the remainder of the paper.

Definition 3.2.3. For a given hexagonal (d,k) constraint, let $H_1, \ldots, H_\alpha$ and $V_1, \ldots, V_\beta$, respectively, denote the components of the directed graphs G_h and G_v. Also, define the disjoint graph unions

$$H = \bigcup_{i=1}^{\alpha} H_i \quad \text{and} \quad V = \bigcup_{i=1}^{\beta} V_i.$$

The graphs H and V are obtained from the graphs G_h and G_v, respectively, by removing all edges between different components. For some pairs (d,k), it may happen that none of the components of G_h (respectively, G_v) are connected by edges to any other components (e.g., see Example 3.2.2), in which case $H = G_h$ (respectively, $V = G_v$).

3.2.2 Preview of Step 5 of the algorithm

An explicit description of the Constant Position Algorithm is given in Section 3.2.4. Here, we motivate Step 5, a key part of the algorithm.

Consider a valid labeling of a $(k+2) \times (k+1)$ vertical rectangle within an $N \times N$ square. Denote the labelings of the lower and upper $(k+1) \times (k+1)$ squares within this vertical rectangle by λ and λ', respectively, and let λ and λ' be nodes common to both H and V. Note that λ' is vertically compatible above λ, i.e., the ordered pair (λ, λ') is a directed edge in V. Since the lower and upper squares overlap in k rows, the number of possibilities for the pair (λ, λ') is limited.

Denote by ρ the top row of the $(k+2)\times(k+1)$ rectangle. For any given labeling λ, all possible labelings λ' that are vertically compatible above λ may agree with each other in certain locations of ρ. By observing numerous

such labelings for the (d, k) pairs of interest, we discovered that for every fixed pair of components that λ and λ' could belong to in H, there always appears to be at least one position in ρ that gets labeled the same by every λ' that is vertically compatible above a given λ.

In other words, whereas the noted position in ρ depends only on the fixed pair of components of λ and λ', the value of the labeling at that position depends additionally on the specific choice of λ within its component. However, the value is constant for each λ' in its component that is vertically compatible above λ. The existence of such a position seems to arise due to the smallness of the difference $(k - d)$ in our cases of interest, and provides a crucial step in our proof that certain hexagonal capacities are zero.

3.2.3 Preview of Lemma 3.2.8

Lemma 3.2.8 provides an important reduction in the amount of computational complexity needed to determine whether the constant position property holds for the particular hexagonal (d, k) constraint being examined. Specifically, it allows one to restrict attention to searching the connected components of two directed graphs rather than the entire graphs. We now provide a preview and summary of the proof of this lemma.

Let $l \in L_{G_h,G_v}(N, N)$. Then the labeling of each $(k+1) \times N$ horizontal strip induced by l corresponds to a walk through G_h containing exactly $(N - k)$ nodes, and there are at most α vertices whose previous vertex in the walk is from a different component (since there are α components in G_h). These at most α vertices in a component are the first occurrences in the walk of a vertex from the component they lie in, and are called "transition vertices". Similarly, there are exactly $(N - k)$ horizontal strips of size $(k + 1) \times N$ in the $N \times N$ square, so there are at most $\alpha(N - k) \leq \alpha N$ transition vertices among all of the $(N - k)$ horizontal strips. We call the squares labeled by the transition vertices "transition squares".

In Lemma 3.2.7, we show that there are not too many possible arrangements of these transition squares in the $N \times N$ square. More specifically, in each $(k + 1) \times N$ horizontal strip, any of the at most α transition squares can be appear in at most $(N-k)$ locations or not at all. Thus there are at most $(N-k+1)^\alpha$ possible arrangements of the transition squares in such a strip. Since there are $(N - k)$ horizontal strips, it follows that there are at most

$$(N - k + 1)^{\alpha(N-k)} \leq 2^{\alpha N \log N} = 2^{o(N^2)}$$

possible arrangements of the transition squares in the entire $N \times N$ square.

Now suppose the number of valid labelings of an $N \times N$ square is large enough (asymptotically) to yield a positive capacity, i.e., $|L_{G_h,G_v}(N, N)| = 2^{\Omega(N^2)}$. Then, by the previous argument, there exists a particular arrangement of these transition squares for which there are many labelings in $L_{G_h,G_v}(N, N)$ (specifically $2^{\Omega(N^2)}$)

that induce this exact arrangement of transition squares in the $N \times N$ square. We show in Lemma 3.2.8 that for such a fixed particular arrangement, there exists a smaller subsquare of the $N \times N$ square that:

(i) is disjoint from the transition squares, and

(ii) has many valid labelings.

Specifically, this subsquare has side length of order $\sqrt{N}$, and has $2^{\Omega(N)}$ valid labelings.

To find such a subsquare, we partition the $N \times N$ square into a grid of about N/σ^2 subsquares with side lengths approximately $\sigma\sqrt{N}$, and we can (and do) choose σ so that the percentage of these subsquares that intersect transition squares stays arbitrarily small as N grows.

The subsquares that intersect transition squares have a total area equal to the number of subsquares (approximately $(k+1)^2\alpha N$) times the subsquare area (approximately $\sigma^2 N$). Thus, the number of valid labelings of all the subsquares that intersect transition squares is at most $2^{(k+1)^2\alpha\sigma^2 N^2}$, whose growth rate as $N \to \infty$ can be made arbitrarily less than that of the capacity growth rate, $2^{C_{\text{hex}}(d,k)N^2}$, by choosing σ small enough. This fact implies that the growth rate of the number of valid labelings of all the subsquares that do *not* intersect transition squares can be made asymptotically arbitrarily close to $2^{C_{\text{hex}}(d,k)N^2}$. Since there are only N/σ^2 subsquares in total, we can show that at least one subsquare that does not intersect any transition square must have about $2^{C_{\text{hex}}(d,k)N}$ valid labelings.

Such a subsquare satisfying (i) and (ii) above can be found for any side length $\sqrt{N}$ by considering larger and larger $N \times N$ squares, and so $|L_{H,G_v}(\sqrt{N},\sqrt{N})|$ is asymptotically $2^{C_{\text{hex}}(d,k)N}$, and thus $|L_{H,G_v}(N,N)|$ is asymptotically $2^{C_{\text{hex}}(d,k)N^2}$. Applying the same argument to transitions between components of G_v that occur in labelings of $L_{H,G_v}(N,N)$ shows that $|L_{H,V}(N,N)|$ is also asymptotically $2^{C_{\text{hex}}(d,k)N^2}$. In other words, in Lemma 3.2.8 we show the capacity of the hexagonal (d,k) constraint is unchanged even if we constrain all walks through G_h and G_v, corresponding to labelings of $(k+1) \times N$ horizontal and $N \times (k+1)$ vertical strips, to stay within a single component each. This theorem justifies restricting attention to the components of G_h and G_v in Steps 3, 4, and 5 of the Constant Position Algorithm.

3.2.4 Algorithm description

We say that an ordered pair of components (H_a, H_b) of G_h is *vertically semi-compatible* if at least one vertex in H_b is vertically compatible above at least one vertex in H_a.

Definition 3.2.4. A hexagonal (d,k) constraint has the *constant position property* if for every pair (H_a, H_b) of vertically semi-compatible components from G_h, there exists $j \in \{1,\ldots,k+1\}$, such that for each labeling λ_a in H_a, the value at position $(j, k+1)$ of every labeling λ_b in H_b vertically compatible above λ_a is constant (note: the value can vary with λ_a).

The constant position property is the key observation that allows us to successfully find new hexagonal (d, k) capacities that equal zero. This property is illustrated in Figure 3.6.

It is assumed that the bottom-most $(k+1) \times N$ horizontal strip of the $M \times N$ rectangle contains $(k+1) \times (k+1)$ square labelings from the component H_a in G_h, and that the overlapping $(k+1) \times N$ horizontal strip one row higher contains $(k+1) \times (k+1)$ square labelings from the component H_b in G_h. The four $(k+1) \times (k+1)$ squares shown in the bottom strip of Figure 3.6 are assumed to have labelings λ_a and λ'_a, as indicated. Any location denoted by x or y is at position $(j, k+1)$ of the $(k+1) \times (k+1)$ squares that are shifted one row up from the squares labeled by λ_a or λ'_a. If the constraint has the constant position property, then the values at each such x are the same no matter which $(k+1) \times (k+1)$ labeling from H_b is chosen, and similarly for the values of each such y, although the value of y may differ from the value of x.

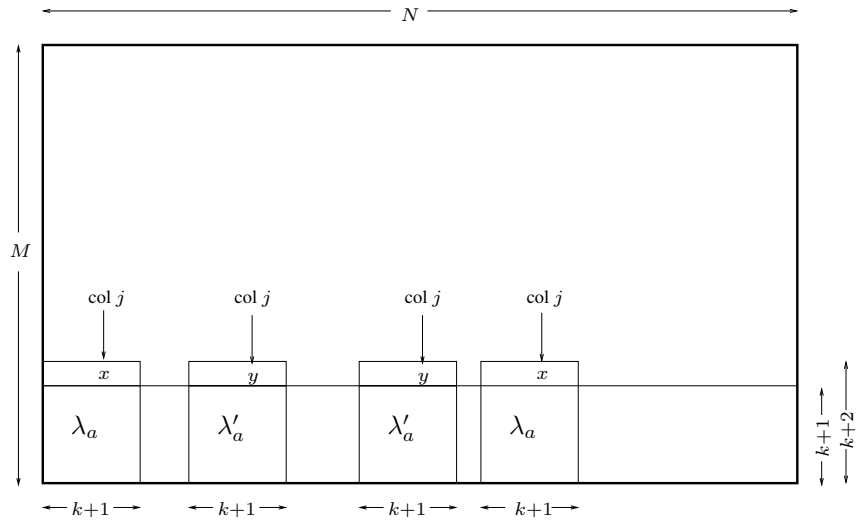


Figure 3.6. Illustration of the constant position property.

The Constant Position Algorithm is not guaranteed to find all such zero capacity constraints, but has proven effective for all cases within its computational complexity capabilities (see Theorem 3.2.13).

Constant Position Algorithm

Step 1: Create the set Λ of valid labelings of a $(k+1) \times (k+1)$ square.

Step 2: Create label graphs $G_h = (\Lambda, E_h)$ and $G_v = (\Lambda, E_v)$.

Step 3: Determine the components of G_h and G_v, i.e., $H_1, \ldots, H_\alpha$ and $V_1, \ldots, V_\beta$.

Step 4: Identify all pairs (H_a, H_b) of vertically semi-compatible components.

Step 5: Determine if the hexagonal (d, k) constraint has the constant position property.

Step 6: If the determination in Step 5 is true, then output "success"; otherwise, output "failure".

3.2.5 Lemmas for zero capacity proof

In this section, four lemmas are given that lead to the main technical result, Theorem 3.2.9 in Section 3.2.6, which asserts that the hexagonal (d, k) capacity is zero whenever the constant position property occurs.

Lemma 3.2.5. $L_{G_h,G_v}(M, N) = L(M, N)$.

Proof. Suppose $l \in L(M, N)$, i.e., l is a valid labeling of an $M \times N$ rectangle. For any $(k + 1) \times N$ horizontal strip, the sequence of labelings of $(k + 1) \times (k + 1)$ squares, sliding left to right one column at a time within the strip, forms a walk through G_h, since these $(k + 1) \times (k + 1)$ square labelings are necessarily valid and each successive pair of labelings corresponds to an edge in G_h by the construction of G_h. Similarly, labelings of vertical strips correspond to walks in G_v.

Conversely, if $l \notin L(M, N)$, then either the d constraint or the k constraint is violated somewhere in the $M \times N$ rectangle, which implies that the labeling of some $(k + 1) \times (k + 1)$ subsquare is not valid. Therefore the labelings of the horizontal and vertical strips containing this subsquare do not correspond to walks through G_h and G_v, respectively. Thus $l \notin L_{G_h,G_v}(M, N)$. ■

The label graphs G_h and G_v contain components, some of which may be reachable from other components. Any edge that connects a vertex from one component to another component cannot itself lie in any component. Thus, a label graph consists of a disjoint union of components together with connecting edges not belonging to any component. If we remove all such connecting edges of the label graphs G_h and G_v, then we are left with the disjoint unions of components of G_h and G_v. For any fixed $M \times N$ rectangle, those disjoint unions of components generate a subset of the labelings of the rectangle that are valid under the hexagonal (d, k) constraint (i.e., $L_{H,V}(M, N) \subseteq L(M, N)$). Thus, using the number of such valid labelings generated by the reduced label graphs to estimate the hexagonal (d, k) capacity will yield a number less than or equal to $C_{\text{hex}}(d, k)$ (i.e., $|L_{H,V}(M, N)| \leq |L(M, N)|$). Lemma 3.2.8 below however, shows that, in fact, the capacity $C_{\text{hex}}(d, k)$ is still obtained with equality by only counting the number of valid labelings in this reduced case (i.e., $|L_{H,V}(M, N)|$ and $|L(M, N)|$ have the same growth rate as $M, N \to \infty$).

First we provide two technical lemmas to aid in the proof of Lemma 3.2.8. In particular, the following lemma gives: (i) a linear (in the number of rows of an $M \times N$ rectangle) upper bound on the number of component transition locations in $T_h(l)$; and (ii) an upper bound on the number of different transition sets $T_h(l)$ that can occur across all $l \in L_{G_h,G_v}(M, N)$.

Lemma 3.2.6. *Suppose label graphs G_h and G_v generate a set of labelings $L_{G_h,G_v}(M,N)$ of an $M \times N$ rectangle. Then the following hold:*

(i) $|T_h(l)| \leq \alpha M$ *for any* $l \in L_{G_h,G_v}(M,N)$,

(ii) $|L_{G_h,G_v}(M,N)/\sim| \leq 2^{\alpha \log(N+1)M}$.

Proof. Let $l \in L_{G_h,G_v}(M,N)$ be a labeling of an $M \times N$ rectangle generated by G_h and G_v. Then, in particular, the labeling of each $(k+1) \times N$ horizontal strip of l corresponds to a walk through G_h. From the definition of $T_h(l)$, a $(k+1) \times (k+1)$ square r can be included in $T_h(l)$ only if the labeling of each $(k+1) \times (k+1)$ square to the left of r in the same strip belongs to a component of G_h other than the component containing the labeling of r. Thus there can be at most α (i.e., the number of components in G_h) $(k+1) \times (k+1)$ squares of any horizontal strip included in $T_h(l)$, and therefore $|T_h(l)| \leq \alpha M$.

Now for the second part. In a given $(k+1) \times N$ strip there are at most $N+1$ options for the position of a $(k+1) \times (k+1)$ square $r \in T_h(l)$, where the one extra option corresponds to labelings from a component not appearing in the horizontal strip at all. Thus there are at most $(N+1)^\alpha$ ways to arrange the at most α possible such squares from $T_h(l)$ that can appear in a given horizontal strip. Since the number of $(k+1) \times N$ horizontal strips is at most M, there are no more than $(N+1)^{\alpha M}$ ways to position all of the squares in $T_h(l)$. Therefore, $|L_{G_h,G_v}(M,N)/\sim| \leq 2^{\alpha \log(N+1)M}$. ■

Recall that two labelings of an $M \times N$ rectangle are equivalent under the relation $\sim$ if the labelings transition within horizontal strips from one component of G_h to another at the same locations. The following lemma helps us prove Lemma 3.2.8 by allowing us to to restrict attention to one equivalence class of such labelings.

Lemma 3.2.7. *If $C_{hex}(d,k) > 0$, then for any $\epsilon > 0$, there exist constants M_1 and N_1 such that for all $M > M_1$ and $N > N_1$, there exists an equivalence class $[l] \in L_{G_h,G_v}(M,N)/\sim$ whose size is at least $2^{(C_{hex}(d,k)-\epsilon)MN}$.*

Proof. Let $\epsilon > 0$. Since

$$C_{\text{hex}}(d,k) = \lim_{M,N\to\infty} \frac{\log |L_{G_h,G_v}(M,N)|}{MN} > 0$$

we can find constants M_0 and N_0 such that $|L_{G_h,G_v}(M,N)| \geq 2^{(C_{\text{hex}}(d,k)-\frac{\epsilon}{2})MN}$ for all $M > M_0$ and $N > N_0$. By Lemma 3.2.6, $|L_{G_h,G_v}(M,N)/\sim| \leq 2^{\alpha \log(N+1)M}$, where α is the number of components in G_h, and thus

for any $M > M_0$ and $N > N_0$,

$$\begin{aligned} 2^{(C_{\text{hex}}(d,k)-\frac{\epsilon}{2})MN} &\leq |L_{G_h,G_v}(M,N)| \\ &= \sum_{[l]\in L_{G_h,G_v}(M,N)/\sim} |[l]| \\ &\leq 2^{\alpha\log(N+1)M} \max_{[l]\in L_{G_h,G_v}(M,N)/\sim} |[l]|. \end{aligned}$$

Therefore,

$$\begin{aligned} \max_{[l]\in L_{G_h,G_v}(M,N)/\sim} |[l]| &\geq 2^{(C_{\text{hex}}(d,k)-\frac{\epsilon}{2})MN-\alpha\log(N+1)M} \\ &= 2^{(C_{\text{hex}}(d,k)-\frac{\epsilon}{2}-\alpha\frac{\log(N+1)}{N})MN}. \end{aligned}$$

Taking N large enough gives $\alpha \cdot \frac{\log(N+1)}{N} \leq \frac{\epsilon}{2}$, proving the lemma. ■

The following lemma shows that, when enumerating the valid labelings of an $M \times N$ rectangle in order to calculate the hexagonal (d,k) capacity, it suffices to count only those labelings for which the labeling of each $(k+1)\times N$ horizontal strip corresponds to a path in a single component of G_h, and the labeling of each $M\times(k+1)$ vertical strip corresponds to a path in a single component of G_v.

Lemma 3.2.8.

$$C_{hex}(d,k) = \lim_{M,N\to\infty} \frac{\log|L_{H,V}(M,N)|}{MN}.$$

Proof. First suppose $C_{\text{hex}}(d,k) = 0$. Then since H and V are subgraphs of G_h and G_v, respectively, for all M and N we have $L_{H,V}(M,N) \subseteq L_{G_h,G_v}(M,N)$, and thus $|L_{H,V}(M,N)| \leq |L_{G_h,G_v}(M,N)|$. In this case,

$$0 \leq \lim_{M,N\to\infty} \frac{\log|L_{H,V}(M,N)|}{MN} \leq \lim_{M,N\to\infty} \frac{\log|L_{G_h,G_v}(M,N)|}{MN} = 0.$$

Thus we will assume $C_{\text{hex}}(d,k) > 0$. Note that the limits on the right and left sides in the statement of the theorem exist by [12]. Let $\epsilon \in (0,2)$. Then by Lemma 3.2.7, there exist M_1, N_1 such that for all $M > M_1$ and all $N > N_1$, there exists an equivalence class $[l] \in L_{G_h,G_v}(M,N)/\sim$ such that $|[l]| \geq 2^{(C_{\text{hex}}(d,k)-\frac{\epsilon}{3})MN}$.

In what follows, we may assume $M = N > \max\{M_1, N_1\}$. We will use floor functions to ensure the

dimensions of the required subsets of an $N \times N$ square are integers. Let σ be a positive real number such that

$$\sigma \leq \sqrt{\frac{\epsilon}{2(k+1)^2\alpha}} \tag{3.1}$$

(where α is the number of components in G_h) and consider an $f(N) \times f(N)$ square ψ, where $f(N) = \left\lfloor \sigma\sqrt{N} \right\rfloor$ (assume N is large enough so that $f(N) \geq 1$). Thus,

$$f(N)^2 \leq \sigma^2 N. \tag{3.2}$$

Define $s(N) = f(N)\left\lfloor \frac{1}{\sigma}\sqrt{N} \right\rfloor$ to be the side length of a large square subset, which occupies nearly all of the $N \times N$ square. Such an $s(N) \times s(N)$ square can be tiled in the obvious way by using $\left\lfloor \frac{1}{\sigma}\sqrt{N} \right\rfloor^2$ disjoint copies of ψ. Let Ψ denote the set of these disjoint subsquares of the $s(N) \times s(N)$ square. Thus,

$$|\Psi| = \left\lfloor \frac{1}{\sigma}\sqrt{N} \right\rfloor^2 \leq N/\sigma^2. \tag{3.3}$$

Note that the number of positions of the $N \times N$ square that are not covered by any $\psi \in \Psi$ is $N^2 - s(N)^2$.

The number of component transitions that occur among all the horizontal strips in the $N \times N$ square is the size of the transition set $T_h(l)$, whereas the number of $f(N) \times f(N)$ subsquares is the size of Ψ. We will demonstrate that there are more of these subsquares than there are component transitions, and in fact, there is always at least one subsquare which does not overlap any $(k+1) \times (k+1)$ square associated with a component transition in G_h. Any such subsquare in Ψ will be called *safe*, and otherwise, *unsafe*.

Since α is the number of components of G_h, by Lemma 3.2.6, we have $|T_h(l)| \leq \alpha N$, for all $l \in L_{G_h,G_v}(M,N)$. That means there are at most $(k+1)^2\alpha N$ of the N^2 positions in the $N \times N$ square that are contained in some $(k+1) \times (k+1)$ square of $T_h(l)$. Each such position can be located in at most one of the (disjoint) squares in Ψ. Thus, the number of safe squares in Ψ is at least

$$\begin{aligned}
|\Psi| - (k+1)^2\alpha N &= \left\lfloor \frac{1}{\sigma}\sqrt{N} \right\rfloor^2 - (k+1)^2\alpha N \\
&> \left(\frac{1}{\sigma}\sqrt{N} - 1\right)^2 - (k+1)^2\alpha N \\
&\geq \left(\sqrt{2(k+1)^2\alpha N/\epsilon} - 1\right)^2 - (k+1)^2\alpha N \qquad \text{[from (3.1)]} \\
&> 0
\end{aligned} \tag{3.4}$$

for sufficiently large N (since $\epsilon < 2$). Therefore, there exists at least one safe square in Ψ.

In the remainder of the proof, we will show that at least one safe square has many valid labelings. Note that there are at most

$$2^{((k+1)^2\alpha N)f(N)^2} \leq 2^{(k+1)^2\alpha\sigma^2 N^2} \qquad \text{[from (3.2)]} \tag{3.5}$$

ways to label the unsafe squares in Ψ.

For any subsquare $\psi \in \Psi$, let $[l](\psi)$ denote the set of labelings of the subsquare ψ induced by the $N \times N$ square labelings from the equivalence class $[l]$. The quantity $|[l]|$ is the number of valid labelings (of the $N \times N$ square) which are equivalent under $\sim$ to l. This number of valid labelings can be upper bounded by multiplying: (i) the number of labelings of the slightly smaller $s(N) \times s(N)$ square induced by the labelings in equivalence class $[l]$; and (ii) the number of (possibly non-valid) labelings of the set of $N^2 - s(N)^2$ positions in the $N \times N$ square that lie outside of the $s(N) \times s(N)$ square. Furthermore, for these two quantities in the product, (i) can be upper bounded by multiplying the numbers of labelings of each ψ in Ψ, induced by the labelings in $[l]$; and (ii) can be upper bounded by raising 2 to the number of positions in the $N \times N$ square that lie outside the $s(N) \times s(N)$ square. Thus,

$$\begin{aligned}
2^{(C_{\text{hex}}(d,k)-\frac{\epsilon}{3})N^2} &\leq |[l]| && \text{[from Lemma 3.2.7]} \\
&\leq \left(\prod_{\psi\in\Psi} |[l](\psi)|\right) \cdot 2^{N^2-s(N)^2} \\
&= \left(\prod_{\text{unsafe } \psi\in\Psi} |[l](\psi)|\right)\left(\prod_{\text{safe } \psi\in\Psi} |[l](\psi)|\right) \cdot 2^{N^2-s(N)^2} \\
&\leq 2^{(k+1)^2\alpha\sigma^2 N^2} \cdot \left(\prod_{\text{safe } \psi\in\Psi} |[l](\psi)|\right) \cdot 2^{N^2-s(N)^2} && \text{[from (3.5)]} \\
&\leq 2^{(k+1)^2\alpha\sigma^2 N^2} \cdot \left(\prod_{\text{safe } \psi\in\Psi} \max_{\text{safe } \psi\in\Psi} |[l](\psi)|\right) \cdot 2^{N^2-s(N)^2} \\
&\leq 2^{(k+1)^2\alpha\sigma^2 N^2} \cdot \left(\max_{\text{safe } \psi\in\Psi} |[l](\psi)|\right)^{|\Psi|} \cdot 2^{N^2-s(N)^2} \\
&\leq 2^{(k+1)^2\alpha\sigma^2 N^2} \cdot \left(\max_{\text{safe } \psi\in\Psi} |[l](\psi)|\right)^{N/\sigma^2} \cdot 2^{N^2-s(N)^2} && \text{[from (3.3)]}\,.
\end{aligned}$$

(Note that each "max" in the lines above is over a nonempty set by (3.4).) Solving for the max gives

$$\begin{aligned}\max_{\text{safe } \psi \in \Psi} |[l](\psi)| &\geq \left(2^{(C_{\text{hex}}(d,k)-\frac{\epsilon}{3})N^2-(k+1)^2\alpha\sigma^2N^2-N^2+s(N)^2}\right)^{\sigma^2/N} \\ &= 2^{(C_{\text{hex}}(d,k)-\frac{\epsilon}{3}-(k+1)^2\alpha\sigma^2-1+(s(N)^2/N^2))\sigma^2N}.\end{aligned}$$

In other words, since any safe ψ is an $f(N) \times f(N)$ square in which each strip of height $(k+1)$ contains labelings from a particular component of G_h, one gets

$$\begin{aligned}|L_{H,G_v}(f(N), f(N))| &\geq 2^{(C_{\text{hex}}(d,k)-\frac{\epsilon}{3}-(k+1)^2\alpha\sigma^2-1+(s(N)^2/N^2))\sigma^2N} \\ &\geq 2^{(C_{\text{hex}}(d,k)-\frac{\epsilon}{2}-(k+1)^2\alpha\sigma^2)\sigma^2N} \qquad (3.6)\end{aligned}$$

for N large enough, since $s(N)^2/N^2$ approaches one from below as $N \to \infty$. Therefore,

$$\begin{aligned}\lim_{N\to\infty} \frac{\log|L_{H,G_v}(N,N)|}{N^2} &= \lim_{N\to\infty} \frac{\log|L_{H,G_v}(f(N),f(N))|}{f(N)^2} \\ &\geq \lim_{N\to\infty} \frac{\log(2^{(C_{\text{hex}}(d,k)-\frac{\epsilon}{2}-(k+1)^2\alpha\sigma^2)\sigma^2N})}{\sigma^2N} && \text{[from (3.6)]} \\ &= C_{\text{hex}}(d,k) - \frac{\epsilon}{2} - (k+1)^2\alpha\sigma^2 \\ &\geq C_{\text{hex}}(d,k) - \epsilon && \text{[from (3.1)]}.\end{aligned}$$

Note that the first two limits above exist by subadditivity (e.g. [12]). Thus, we have

$$\begin{aligned}C_{\text{hex}}(d,k) &\leq \lim_{N\to\infty} \frac{\log|L_{H,G_v}(N,N)|}{N^2} \quad \text{(since } \epsilon \text{ was arbitrary)} \\ &\leq C_{\text{hex}}(d,k) \quad \text{(since } L_{H,G_v}(N,N) \subseteq L_{G_h,G_v}(N,N)).\end{aligned}$$

If we now start with H and G_v (instead of G_h and G_v) and repeat the argument in the vertical direction, then we will obtain

$$\lim_{N\to\infty} \frac{\log|L_{H,V}(N,N)|}{N^2} = \lim_{N\to\infty} \frac{\log|L_{H,G_v}(N,N)|}{N^2} = C_{\text{hex}}(d,k),$$

which completes the proof of the lemma. ■

3.2.6 Zero capacity theorem

The following theorem establishes that $C_{\text{hex}}(d,k) = 0$ whenever the Constant Position Algorithm outputs "success".

Theorem 3.2.9. *If the hexagonal (d,k) constraint has the constant position property, then the hexagonal (d,k) capacity is zero.*

Proof. Lemma 3.2.5 showed that there is a bijection between valid labelings of strips and certain walks through G_h and G_v, and so instead of counting valid labelings, it suffices to count corresponding walks.

Let s_i denote the $(k+1) \times N$ horizontal strip (in an $M \times N$ rectangle), whose bottom row is the ith row, counting from the bottom, of the $M \times N$ rectangle, i.e., $1 \leq i \leq M-k$.

Let W_a be a valid labeling of the bottommost $(k+1) \times N$ horizontal strip, s_1. Let W_b be a valid labeling of s_2, such that W_a and W_b agree on $s_1 \cap s_2$.

Suppose the walk through G_h corresponding to W_a lies in a component H_a, and the walk corresponding to W_b lies in a component H_b (these components must be cyclic since $N > k+1$). Since W_a and W_b agree on their common $k \times N$ horizontal strip (i.e., on rows 2 through $k+1$ from the bottom of the $M \times N$ rectangle), the labeling λ_a of any $(k+1) \times (k+1)$ square labeled by W_a, and the labeling λ_b of the $(k+1) \times (k+1)$ square shifted one row upward labeled by W_b, satisfy $(\lambda_a, \lambda_b) \in E_v$. Thus (H_a, H_b) is a pair of vertically semi-compatible components identified in Step 4.

Now, since the hexagonal (d,k) constraint has the constant position property by assumption, let $j \in \{1, \ldots, k+1\}$ be an index corresponding to (H_a, H_b) specified in the constant position property.

By the constant position property, for each particular labeling W_a of s_1, the labeling by W_b of column j of the top row of the leftmost $(k+1) \times (k+1)$ square in s_2 is completely determined by the labeling by W_a of the leftmost $(k+1) \times (k+1)$ square in s_1. (Recall that the walks corresponding to W_a and W_b do not transition between different components of G_h.) The same fact remains true if the two $(k+1) \times (k+1)$ squares slide together one column at a time from left to right. Thus, the sequence of values of the labeling by W_b in the top row of s_2, from horizontal positions j to $N-(k+1-j)$ in the $M \times N$ rectangle, are all completely determined by the labeling of s_1. These positions consist of at least the middle $N-2k$ positions in the top row of s_2, for any j. In sum, the labeling of s_1 immediately determines the labeling of k out of $k+1$ rows of s_2, and now we have shown that it also determines all but at most $2k$ of the positions in the top row of s_2. As a result, the number of different possible valid labelings of s_2, corresponding to walks from a given cyclic component of G_h, for a given labeling of s_1 corresponding to walks from a (possibly different) given cyclic component of G_h, is at most 2^{2k}. Varying the labelings W_b of s_2 over the α cyclic components of G_h increases this upper bound to at most $\alpha 2^{2k}$.

Continuing the argument from the previous paragraph inductively, moving one row upward in the $M \times N$ rectangle each time, shows that, for a given labeling of s_1, there are at most $(\alpha 2^{2k})^{M-k}$ ways to label the $M \times N$ rectangle excluding s_1.

Then, using the loose upper bound that there are at most $2^{(k+1)N}$ possible valid labelings of s_1 corresponding to a walks through any particular cyclic component of G_h, there are at most

$$2^{(k+1)N}(\alpha 2^{2k})^{M-k} \leq 2^{(k+1)N+(2k+\log\alpha)M} \tag{3.7}$$

labelings in $L_{H,V}(M,N)$. Therefore,

$$\begin{aligned}
0 &\leq C_{\text{hex}}(d,k) \\
&= \lim_{M,N\to\infty} \frac{\log|L_{H,V}(M,N)|}{MN} && \text{[from Lemma 3.2.8]} \\
&\leq \lim_{M,N\to\infty} \frac{\log\left(2^{(k+1)N+(2k+\log\alpha)M}\right)}{MN} && \text{[from (3.7)]} \\
&= \lim_{M,N\to\infty} \frac{(k+1)N+(2k+\log\alpha)M}{MN} \\
&= 0.
\end{aligned}$$

■

By successful execution of the Constant Position Algorithm presented in Section 3.2.4, we have obtained the following result.

Result 3.2.10. *The Constant Position Algorithm verified that the following hexagonal (d,k) constraints satisfy the constant position property:*

- $k = d+2$ *and* $2 \leq d \leq 9$
- $k = d+3$ *and* $3 \leq d \leq 9$
- $k = d+4$ *and* $d \in \{4,5,7,9\}$.

The following corollary follows from Result 3.2.10 and Theorem 3.2.9. We note that, while the proof of this corollary is rigorous, verification of the constant position property aspect of it appears to be virtually impossible to do by hand, and relies on computer-assisted verification of an enormous number of cases.

Corollary 3.2.11. *The hexagonal (d,k) capacity is zero whenever*

- $k = d+2$ *and* $2 \leq d \leq 9$

- $k = d + 3$ *and* $3 \leq d \leq 9$
- $k = d + 4$ *and* $d \in \{4, 5, 7, 9\}$.

We note that some of the zero capacities in Corollary 3.2.11 immediately imply others,[2] since $C_{\text{hex}}(d + 1, k) \leq C_{\text{hex}}(d, k)$ and $C_{\text{hex}}(d, k - 1) \leq C_{\text{hex}}(d, k)$,but we include them since they were previously unproven. The following conjecture states a converse to Theorem 3.2.9.

Conjecture 3.2.12. *If the hexagonal* (d, k) *capacity is zero, then the hexagonal* (d, k) *constraint has the constant position property.*

While we presently do not know if this conjecture is true in general, we have verified it computationally for a substantial number of cases for small d and k, namely for all $d \leq 9$.

Theorem 3.2.13. *Conjecture 3.2.12 is true for all* $d \leq 9$.

3.2.7 Algorithm implementation details

We describe below specific implementation details of each step of the Constant Position Algorithm, and indicate the greatest computational burdens and ways to reduce complexity.

Step 1: Creating the valid labelings of a $(k + 1) \times (k + 1)$ **square**

An iterative method is used that recursively creates the valid labelings of a $(k + 1) \times (k + 1)$ square from valid labelings of smaller rectangles. Pointers are used to represent adjacent subsquares in labelings, which leads to a massive saving in computational complexity in Step 2.

Let B_m be the collection of valid labelings of a $(k + 1) \times m$ rectangle. We first create $B_1, B_2, \ldots, B_{d+1}$ (in that order) and then use B_{d+1} to create B_{k+1}, thus avoiding explicit generation of $B_{d+2}, \ldots, B_k$. Generating these sets wastes memory, since the d constraint has no effect on the validity of labelings once $m > d + 1$, and the k constraint does not start having an effect until $m = k + 1$. In fact, the sizes of $B_{d+2}, \ldots, B_k$ can be quite large, but B_{k+1} is generally much smaller, since the k constraint plays a role. Thus, directly generating B_{k+1} from B_{d+1} saves memory at the expense of extra computation.

In order to create B_1, all valid labelings of a $(k + 1) \times 1$ rectangle (i.e., a column) are generated directly, and each is given a unique ID. To create B_2, it is determined which labelings of B_1 may be placed horizontally next to each other to create valid labelings of a $(k + 1) \times 2$ rectangle.

In order to create $B_3, B_4, \ldots$, a particular data structure built from pointers (which will be called IDs) is used to represent a labeling. Suppose $2 \leq m \leq d + 1$, and let λ be a labeling in B_m. Also, let $\lambda_a, \lambda_b \in B_{m-1}$ be

[2]Specifically, $C_{\text{hex}}(4, 8) = 0$ implies $C_{\text{hex}}(4, 7) = 0$ and $C_{\text{hex}}(5, 8) = 0$, $C_{\text{hex}}(5, 9) = 0$ implies $C_{\text{hex}}(5, 8) = 0$ and $C_{\text{hex}}(6, 9) = 0$, $C_{\text{hex}}(7, 11) = 0$ implies $C_{\text{hex}}(7, 10) = 0$ and $C_{\text{hex}}(8, 11) = 0$, and $C_{\text{hex}}(9, 13) = 0$ implies $C_{\text{hex}}(9, 12) = 0$.

the labelings of the left-most and right-most $(k+1) \times (m-1)$ subrectangles, respectively, of the $(k+1) \times m$ rectangle labeled by λ. The labeling λ is represented using a data structure that contains: (1) an ID (unique among labelings in B_m); (2) an array containing the IDs of the labelings of each column; (3) an array containing the IDs of λ_a and λ_b.

Creation of B_m, with $3 \leq m \leq d+1$, is now described. For every two labelings $\lambda, \lambda' \in B_{m-1}$ such that $\lambda_b = \lambda'_a$, a labeling λ^* of a $(k+1) \times m$ rectangle is induced, where the labeling of the left-most $(k+1) \times (m-1)$ rectangle is λ, and the labeling of the right-most column is the labeling of the right-most column of λ'. The new labeling λ^* is valid if and only if the restriction of λ^* to each row and diagonal does not violate the d constraint, since the restriction of λ^* to any column is valid by construction, and the rows and diagonals are not long enough for the labeling to violate the k constraint. If λ^* passes these validity checks, it is added to B_m.

Finally, generating B_{k+1} from B_{d+1} follows essentially the same procedure, except now consideration is made of all $(k-d+1)$-tuples of labelings in B_{d+1} that can be overlaid on each other successively (as in the previous paragraph) to create a labeling of a $(k+1) \times (k+1)$ rectangle. This labeling is valid if and only if the restriction to any row and diagonal does not violate the k constraint, since the restriction to any column is valid by construction, and the restriction to any row or diagonal already satisfies the d constraint, since any $d+1$ consecutive positions are labeled by some labeling in B_{d+1}. If the labeling passes these checks of validity, it is added to B_{k+1}.

Figure 3.9 shows the complexity reduction using this method compared to two other less efficient methods.

Step 2: Creating the label graphs G_h and G_v

A method is given to create the edge sets E_h and E_v. Each construction is described separately, as they are slightly different.

To determine the out-edges in E_h from a labeling (i.e., a vertex of G_h) λ of a $(k+1) \times (k+1)$ square, the data structure from Step 1 that represents λ is useful. Recall that Step 1 assigns an ID to each labeling of a $(k+1) \times (d+1)$ rectangle that appears in λ, and let λ_I denote the $(k-d+1)$-length array of the IDs of these sub-rectangles. Then to check if there is an edge in G_h from λ to a labeling λ' of a $(k+1) \times (k+1)$ square, it suffices to check if $\lambda_I(j) = \lambda'_I(j-1)$ for $j = 2, \ldots, k-d+1$. This reduces the problem of determining if $(\lambda, \lambda') \in E_h$ to comparing $(k-d)$ pairs of integers (instead of the more complex comparison of all $k(k+1)$ values of the bit positions in the overlapping rectangles).

The runtime of this step can be reduced by decreasing the number of pairs of valid square labelings that are examined. In practice, for the constraints of interest, each valid square labeling has relatively few out-edges in

E_h (typically, almost all have out-degree equal to one, and the rest generally have out-degree less than 5). As a pre-processing step, each labeling λ' is sorted by the element $\lambda'_I(1)$, which is the ID of the leftmost $(k+1) \times (d+1)$ sub-rectangle of λ'. Then for each square labeling λ, only square labelings λ' satisfying $\lambda'_I(1) = \lambda_I(2)$ are examined to find the out-edges from λ in E_h.

However, the same method cannot be used in determining the edges of G_v, since no record of any $(d+1) \times (k+1)$ sub-rectangles was made in Step 1. Nevertheless, a similar pre-processing step is performed by assigning an integer ID to each valid $(k+1) \times 1$ column, and then sorting the square labelings based on the ID of each square's leftmost column. Then a search is made for out-edges in E_v only between square labelings λ and λ' for which the labeling of the top k positions of the leftmost column of λ agree with the labeling of the bottom k positions of the leftmost column of λ'. This method indeed helps in practice, even though it does not prune the potential set of edges in E_v as much as the previous method pruned the potential set of edges in E_h.

Step 3: Finding the components of the label graphs G_h and G_v

We prune the label graphs G_h and G_v by iteratively removing the square labelings from Λ that are sources or sinks in either G_h or G_v. When such a labeling is removed, the corresponding vertices in both G_h and G_v are eliminated, since if a square labeling is a source or sink in either graph, then it cannot appear in a labeling of the infinite plane, and so it can be removed from both graphs (see Lemma 3.2.8).

Since removing sources or sinks from one graph may create more sources or sinks in the other graph, the implementation of the Constant Position Algorithm ping-pongs between removing sources and sinks from both G_h and G_v until the process halts.

We note that Tarjan's algorithm for finding the components of G_h (as described next) would also identify the sources and sinks in G_h, but our tests indicate that pruning sources and sinks drastically reduces the number of valid labelings of $(k+1) \times (k+1)$ squares (see Figure 3.7), and so it was performed here to reduce memory consumption.

The components of the pruned G_h are determined using Tarjan's algorithm [21]. The runtime of Tarjan's algorithm is linear in the number of vertices and edges, i.e., $O(|\Lambda| + |E_h|)$. The pruning step described previously is in part to prevent stack overflow, as the implementation of Tarjan's algorithm is recursive, and some (d, k) constraints allow on the order of billions of valid labelings of a $(k+1) \times (k+1)$ square, which could potentially result in as many levels of recursion.

Note that no explicit determination is made of the components of G_v, since such information is not needed in the rest of the algorithm.

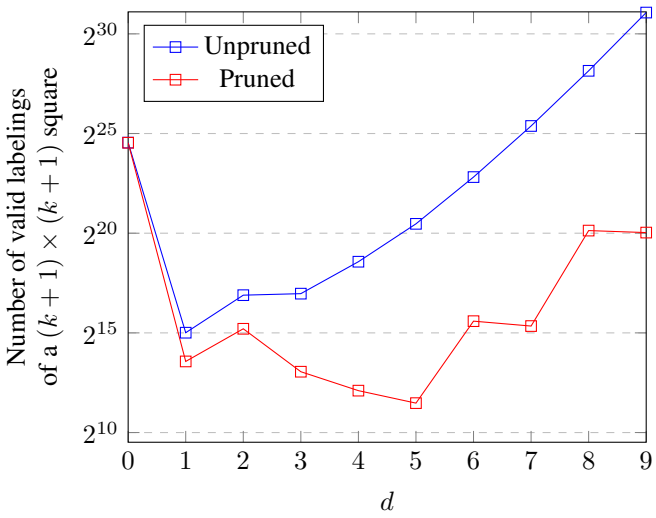


Figure 3.7. Plot showing the number of valid labelings of a $(k+1) \times (k+1)$ square versus d, and the number of pruned valid labelings of a $(k+1) \times (k+1)$ square versus d, where $k = d + 4$.

Step 4: Finding the pairs (H_a, H_b) of vertically semi-compatible components of G_h

To find pairs (H_a, H_b) that are vertically semi-compatible, we examine each component H_a of the pruned G_h, and every labeling λ_a in H_a, and check if λ_a has an out-edge in the pruned G_v to a labeling in H_b. Note that this condition does not guarantee that a walk through H_a and a walk through H_b could correspond to labelings of (vertically) successive $(k+1) \times N$ horizontal strips in an $M \times N$ rectangle. However, this weaker condition is much simpler to verify, and performs well despite creating more cases to check, since in practice the out-degree of most vertices is small.

Step 5: Showing the hexagonal (d, k) constraint has the constant position property

For the particular hexagonal (d, k) constraints considered in this paper, we observed that each component of the pruned G_h is vertically semi-compatible with a very small number of other components, often just one or two (see Table 3.2). This fact, combined with the small out-degrees of vertices in the pruned G_v, allows relatively quick verification of the constant position property.

3.2.8 Computational complexity of the algorithm

The Constant Position Algorithm was run for a number of previously open cases, and succeeded in showing that $C_{\text{hex}}(d, k) = 0$ in the following five new (d, k) constraints: $(6, 9)$, $(4, 8)$, $(5, 9)$, $(7, 11)$, and $(9, 13)$. These cases are listed in order of increasing computational complexity. Even though the $(6, 9)$ case follows immediately from the $(5, 9)$ case, since $C_{\text{hex}}(6, 9) \leq C_{\text{hex}}(5, 9) = 0$, we include it and other already-known cases in the information below as these cases provide interesting data about complexity.

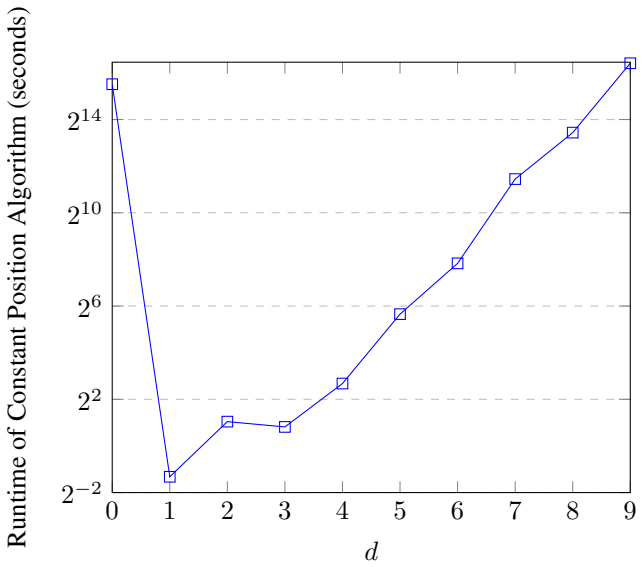


Figure 3.8. Runtime, in seconds, on a supercomputer of the Constant Position Algorithm for hexagonal (d, k) constraints, as a function of d, where $k = d + 4$. Note that one day is about $2^{16.4}$ seconds.

The run time of the algorithm is shown in Figure 3.8 for constraints of the form $(d, d + 4)$, for $0 \leq d \leq 9$ (the case $(6, 9)$ is not shown, but runs about as fast as the $(4, 8)$ case). Even though our results were limited in this plot to the cases $d = 4, 5, 7, 9$, the algorithm was run for all $d \leq 9$ in order to better understand the complexity.

There is roughly exponential growth for $1 \leq d \leq 9$, but there is a sharp decline from $d = 0$ to $d = 1$, with the runtime of the $d = 0$ case on the order of the runtime for the $d = 9$ case. There are two reasons for this phenomenon. First, as seen in Figure 3.7, the number of valid $(k + 1) \times (k + 1)$ squares in the $d = 0$ case, even after pruning, is many orders of magnitude larger than the number of valid $(k + 1) \times (k + 1)$ squares in the $d = 1$ case. Second, since there is essentially no d constraint in the $d = 0$ case, the graphs G_h and G_v in the $d = 0$ case are highly connected, which slows the Constant Position Algorithm in multiple places. The high connectivity of the graphs G_h and G_v in the $d = 0$ case is evidenced by the fact that even the $(0, 1)$ constraint has positive capacity [**?**, 10, 11].

The highest complexity case that could be run in reasonable time was when $(d, k) = (9, 13)$, in which case the runtime was about 2^{16} seconds, or about one full day. The next open case of interest would be when $(d, k) = (11, 15)$, which is projected to take at least about 8 days of runtime to complete. Since our supercomputer time allocation was limited to two full days at a time, the $d = 11$ case was not attempted. Furthermore, the amount of memory usage grew exponentially in d, which posed further difficulties.

Table 3.2 shows, for each hexagonal (d, k) constraint having zero capacity and with $d \leq 9$, the specific sizes of vertex and edge sets in G_h, the size of G_h after pruning, the number of (both total and non-cyclic) compo-

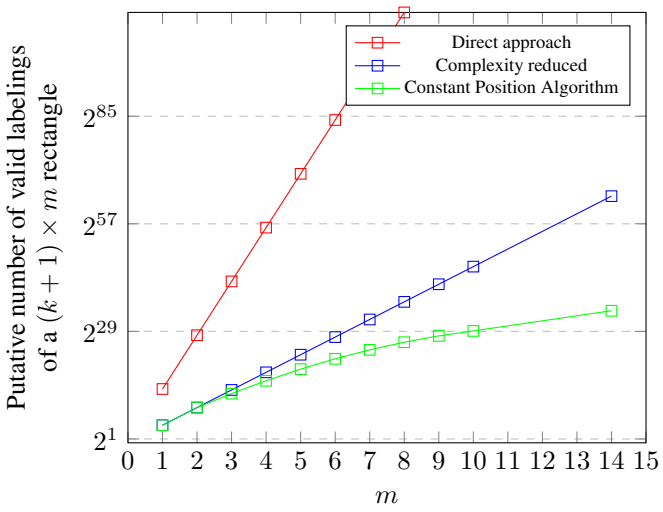


Figure 3.9. Plot showing the number of putative valid labelings of a $(k+1) \times m$ rectangle versus m for the case $(d, k) = (9, 13)$ based on three different methods: a direct method, a complexity reduced method, and our adaptive pruning method.

nents of G_h, the average number of vertices per component, and the number of pairs of vertically semi-compatible components.

One contribution of the Constant Position Algorithm is its dramatic reduction in computational complexity compared to more straightforward approaches. This complexity improvement enables the algorithm to discover zero hexagonal capacity constraints that would otherwise be computationally prohibitive. In particular, the algorithm adaptively prunes the set of putative valid labelings for each width m rectangle, and recursively builds the set of valid labelings of width $m+1$ from those obtained for width m. This reduction in the number of labelings to inspect makes the overall task more manageable.

A plot of the number of putative valid $(k+1) \times m$ labelings, as a function of the rectangle width m, is illustrated in Figure 3.9 for the case $(d, k) = (9, 13)$. The green curve corresponds to our complexity reduced version of the Constant Position Algorithm. The "direct approach" (shown in red) corresponds to a naive implementation without complexity reduction, i.e., checking all $2^{(k+1)m}$ possible labelings of a $(k+1) \times m$ rectangle for validity, one-by-one. The "complexity reduced" method (shown in blue) corresponds to first creating the valid labelings of a column of height $(k+1)$, and then checking which possible arrangements of these columns create valid labelings of a $(k+1) \times m$ rectangle. It can be seen that at the highest level of complexity when $m = 14$, the number of labelings that we must check for validity in our algorithm is about 2^{34}, which is considerably lower than the roughly 2^{64} required for the complexity-reduced blue curve.

A plot of the number of valid $(k+1) \times (k+1)$ squares versus d, where $k = d+4$, is shown in Figure 3.7. The blue curve shows the number of valid labelings of a $(k+1) \times (k+1)$ square that are found in Step 1 of the

Table 3.2. Computational complexity parameters for all hexagonal (d, k) constraints with $d \leq 9$ where $C_{\text{hex}}(d, k) = 0$. The quantities refer to the size of various parameters of the directed graph G_h, either before or after pruning. Our newly discovered cases for $k = d + 3$ and $k = d + 4$ are indicated by the asterisks on the far left. Every row in this table corresponds to a hexagonal (d, k) constraint that was confirmed via computer search to satisfy the constant position property.

(d, k)	Number of vertices $\lvert\Lambda\rvert$ before pruning	Number of edges $\lvert E_h\rvert$ before pruning	Number of vertices after pruning	Number of components α	Average number of vertices per component after pruning	Non-cyclic components	Number of vertically semi-compatible pairs (H_a, H_b)
$(1, 2)$	3	3	3	1	3.0	0	1
$(1, 3)$	20	20	3	1	3.0	0	1
$(2, 3)$	12	11	3	1	3.0	0	1
$(2, 4)$	135	170	65	1	65.0	0	1
$(3, 4)$	19	15	15	3	5.0	0	3
$(3, 5)$	186	150	15	3	5.0	0	3
* $(3, 6)$	3,539	4,082	202	19	10.6	0	19
$(4, 5)$	82	51	15	3	5.0	0	3
$(4, 6)$	817	678	202	19	10.6	0	19
* $(4, 7)$	13,400	15,020	234	19	12.3	0	19
* $(4, 8)$	388,397	572,552	4,391	252	17.4	0	264
$(5, 6)$	329	174	133	19	7.0	0	19
$(5, 7)$	3,158	2,040	133	19	7.0	0	19
* $(5, 8)$	53,701	49,201	2,384	245	9.7	0	249
* $(5, 9)$	1,449,120	1,883,744	2,846	235	12.1	0	237
$(6, 7)$	1,756	783	133	19	7.0	0	19
$(6, 8)$	17,281	10,480	2,220	241	9.2	0	241
* $(6, 9)$	283,200	243,603	2,248	241	9.3	0	241
$(7, 8)$	10,133	4,534	2,025	225	9.0	0	225
$(7, 9)$	102,614	56,923	2,025	225	9.0	0	225
* $(7, 10)$	1,696,233	1,324,066	39,946	3,663	10.9	0	3,663
* $(7, 11)$	43,511,098	47,066,939	41,478	4,431	9.4	736	4,478
$(8, 9)$	65,676	26,159	2,025	225	9.0	0	225
$(8, 10)$	689,199	354,239	39,946	3,663	10.9	0	3,663
* $(8, 11)$	11,667,348	8,594,525	39,974	3,663	10.9	0	3,663
$(9, 10)$	462,531	181,758	37,851	3,441	11	0	3,441
$(9, 11)$	5,142,892	2,548,549	37,851	3,441	11	0	3,441
* $(9, 12)$	88,149,741	60,893,380	1,068,296	82,697	13.0	0	82,697
* $(9, 13)$	2,244,615,058	2,132,131,265	1,069,432	83,601	12.8	892	83,652

algorithm, while the red curve shows the number of these squares retained after pruning all sources and sinks in the label graphs G_h and G_v in Step 2. As the plot indicates, for larger values of d, the vast majority of valid labelings of a $(k+1) \times (k+1)$ square do not appear in a component of G_h and a component of G_v, Eliminating these squares from consideration during Step 2 of the algorithm drastically improves the efficiency of the subsequent steps.

Computing resources

Due to the large amount of memory required to run the Constant Position Algorithm for larger values of d and $(k-d)$, the algorithm was implemented on a large-memory node of the Comet supercomputer at the San Diego Supercomputer Center. These nodes have a clock speed of 2.2 GHz. The OpenMP API was used for implementing the parallelism, primarily in Steps 1-3 of the algorithm. We used 64 CPUs in parallel and a total of 800 gigabytes of shared memory. The computationally most complex case we ran was $(d, k) = (9, 13)$, which took about one full day to run and had about 2 billion valid 14×14 square labelings.

The next unsolved case for which the hexagonal capacity is not known to be zero or positive is $(11, 15)$, which could not be performed with our current resources. Using statistical sampling, we estimate it would have about 80 billion valid 16×16 square labelings, which suggests that the time and space complexities would increase by about 40-fold compared to the $(9, 13)$ case. Such resources are presently not easily available.

3.3 Forbidden String Algorithm for proving zero hexagonal (d, k) capacity

In this section we present a second algorithm for automatically and rigorously proving that certain hexagonal (d, k) capacities are zero. In this entire section, we will frequently use the notation $A(i, j)$ to denote the labeling of a position in an array by a binary value, or else an "Unused" indicator. The position (i, j) uses ordinary Cartesian integer coordinates.[3]

A binary string is *forbidden* for a hexagonal (d, k) constraint if there exists a positive integer N such that for any rectangle with side lengths at least N, no valid labelings of the rectangle contain the string in any horizontal, vertical, or northeast diagonal. For some pairs (d, k) there exist one or more forbidden strings of the form 10^z1, where $d \leq z \leq k$, whereas for other pairs no such strings are forbidden.

If it is known that $C_{\text{hex}}(d, k-1) = 0$ and 10^k1 is a forbidden string for the hexagonal (d, k) constraint, then we deduce $C_{\text{hex}}(d, k) = C_{\text{hex}}(d, k-1) = 0$. Similarly, if $C_{\text{hex}}(d+1, k) = 0$ and 10^d1 is a forbidden string for the hexagonal (d, k) constraint, then $C_{\text{hex}}(d, k) = C_{\text{hex}}(d+1, k) = 0$. The basic idea behind our Forbidden String

[3] In contrast, in Section 3.4, we use matrix (row,column) coordinates for our arrays.

Algorithm is to establish, by a computer-generated proof, that either 10^d1 or 10^k1 is forbidden for the hexagonal (d, k) constraint, and then rely on a previously known fact that either $C_{\text{hex}}(d, k-1) = 0$ or $C_{\text{hex}}(d+1, k) = 0$.

3.3.1 Non-forbidden strings

Sometimes, however, it is provably impossible for certain strings 10^z1 to be forbidden. Here, we demonstrate a number of such cases, and thereby concentrate the use of the Forbidden String Algorithm on other cases.

In what follows, we will assume only integer valued coordinates of points in the plane. Suppose all of the points on the northwest line $y = -x + a$ and the northeast line $y = x + c$ are labeled 1 and all other points on the integer lattice are labeled 0. These two lines intersect if and only if a and c are either both even or both odd. As a result, any point on the line $y = -x + a$ lies on the same northeast line as one point on the line $y = -x + b$ if and only iff a and b have the same parity, or equivalently, if and only if $b - a$ is even. The horizontal (and vertical) number of points between the two northwest lines $y = -x + a$ and $y = -x + b$ is $b - a - 1$.

Suppose (u, v) lies on the line $y = -x + a$ and (s, t) lies on the line $y = -x + b$, and both points lie on the same northeast line $y = x + c$. Then $u = (a - c)/2$, $v = (a + c)/2$, $s = (b - c)/2$, and $t = (b + c)/2$, so $(s, t) = (u, v) + (1, 1)(b - a)/2$. Thus, the diagonal number of points between these two lines is $\frac{b-a}{2} - 1 = \frac{z-1}{2}$, where $z = b - a - 1$. In other words, if two northwest lines, separated horizontally by an odd number z of points, are labeled by 1s and every other point is labeled 0, then the string 10^z1 appears horizontally and vertically, and the string $10^{(z-1)/2}1$ appears diagonally.

For any sequence of nonnegative integers $a_1, \ldots, a_n$, define $s_j = j + a_1 + \cdots + a_j$ for all $j \geq 1$ and let $s_0 = 0$. Define a labeling by

$$L_{a_1,\ldots,a_n}(x, y) = \begin{cases} 1 & \text{if } x + y = s_j \mod s_n \text{ where } 0 \leq j \leq n-1 \\ 0 & \text{else.} \end{cases}$$

Then $L_{a_1,\ldots,a_n}$ consists of periodically repeating infinite northwest diagonals of 1s. Each infinite horizontal row is labeled by infinitely repeating the pattern $10^{a_1}10^{a_2}1 \ldots 10^{a_n}$ to the left and right. The 1 on the far left side of this pattern is at the origin, and the entire pattern shifts one position to the left each time one moves upward by one row. This lattice contains runs of a_1, a_2, ... ,a_n zeros horizontally and vertically between consecutive 1s.

If a_i is odd, then the string 0^{a_i} appears horizontally and vertically and the string $0^{(a_i-1)/2}$ appears diagonally, between the consecutive northwest diagonal lines passing through the points $(s_{i-1}, 0)$ and $(s_i, 0)$ in the labeling $L_{a_1,\ldots,a_n}$.

On the other hand, if a_i and a_{i+1} are both even, then the number of points horizontally between the north-

west diagonal lines passing through $(s_{i-1}, 0)$ and $(s_{i+1}, 0)$ is odd, so the string $0^{(a_i+a_{i+1})/2}$ appears diagonally between these two diagonal lines (i.e., it skips the diagonal line passing through the point $(s_i, 0)$). The string 0^{a_i} appears horizontally and vertically between the diagonal lines passing through $(s_{i-1}, 0)$ and $(s_i, 0)$, and the string $0^{a_{i+1}}$ appears horizontally and vertically between the diagonal lines passing through $(s_i, 0)$ and $(s_{i+1}, 0)$.

The following theorem eliminates certain possible strings as being forbidden.

Theorem 3.3.1. *For a given hexagonal (d, k) constraint, the string 10^z1 is not forbidden whenever $\hat{d} \leq z \leq \hat{k}$, where*

$$\hat{d} = \begin{cases} d & \text{if } d \text{ even} \\ d+1 & \text{if } d \text{ odd} \end{cases} \qquad \hat{k} = \begin{cases} k & \text{if } d \text{ even} \\ k-1 & \text{if } d \text{ odd.} \end{cases}$$

Proof. Take $a_1 = \hat{d}$, $a_2 = \hat{d} + 2$, $a_3 = \hat{d} + 4$, ... $a_n = \hat{k}$ in $L_{a_1,\ldots,a_n}$, and note that each a_i is even.

When $1 \leq i \leq n-1$, we have $a_{i+1} = a_i + 2$, so the string $0^{(a_i+a_{i+1})/2} = 0^{a_i+1}$ appears diagonally and the strings 0^{a_i} and $0^{a_{i+1}}$ appear horizontally and vertically. And when $i = n$, the string $0^{(a_1+a_n)/2}$ appears diagonally and the strings 0^{a_1} and 0^{a_n} appear horizontally and vertically.

In summary, the pattern 10^z1 appears along every horizontal row and vertical column whenever z is even and $\hat{d} \leq z \leq \hat{k}$. Also, since $(a_i + a_{i+1})/2 = ((\hat{d} + 2(i-1)) + (\hat{d} + 2(i-1) + 1)/2 = \hat{d} + 2i - 1$, the pattern 10^z1 appears along every northeast diagonal whenever z is odd and $\hat{d} \leq z \leq \hat{k}$.

Thus, the labeling satisfies the hexagonal (d, k) constraint and 10^z1 is not forbidden, since it appears in the valid labeling $L_{\hat{d},\hat{d}+2,\ldots,\hat{k}}$. ■

The Forbidden String Algorithm relies on deducing that $C_{\text{hex}}(d, k) = C_{\text{hex}}(d+1, k)$ by showing 10^d1 is forbidden, or $C_{\text{hex}}(d, k) = C_{\text{hex}}(d, k-1)$ by showing 10^k1 is forbidden. Theorem 3.3.1 shows that, in particular, if both d and k are even, then this approach will not work, since 10^d1 and 10^k1 are not forbidden strings of the hexagonal (d, k) constraint.

In some cases, certain strings of the form 10^z1 cannot be forbidden for a given hexagonal (d, k) constraint, but they do not fall within the scope of Theorem 3.3.1. In the theorem below, for each hexagonal (d, k) constraint given, we exhibit a single square labeling that can validly tile the plane, and we deduce therefore that any strings within it are not forbidden.

Theorem 3.3.2. *The string 10^z1 is not forbidden for the hexagonal (d, k) constraint in the following cases: (i) $d = 5$, $k = 8$, $z = d$; (ii) $d = 7$, $k = 11$, $z \in \{d, k\}$.*

Proof. The following square labelings in Figure 3.10 and Figure 3.11 satisfy the mentioned constraint, can tile the plane, and contain the strings asserted to not be forbidden. Figure 3.12 shows portions of the hexagonal lattice tiled by these labelings.

1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0	0	0	0	0

Figure 3.10. A tileable valid labeling of a 15×15 square for the hexagonal $(5, 8)$ constraint. The string $10^5 1$ appears in the top row (in red) and thus is not forbidden.

■

3.3.2 Algorithm description

Forbidden String Algorithm

Step 1: Set $A(i, j) =$ Unused, for all i, j.

Step 2: Assume $A(0, 0) = 1$ and $A(z + 1, 0) = 1$.

Step 3: Force 0s for the d constraint.

Step 4: If there does not exist 0^{k+1} horizontally, vertically, or diagonally, then go to Step 5.

Else repeatedly pop the stack until the top of the stack is an assumed 1.

If the top of the stack is the original assumption $A(0, 0) = 1$, then a proof is found, so exit.

Else convert the top of the stack to a forced 0 and repeat Step 4.

Step 5: Select an unused position (x, y) and assume $A(x, y) = 1$.

Step 6: Go to Step 3.

1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0

Figure 3.11. A tileable valid labeling of a 24×24 square for the hexagonal $(7, 11)$ constraint. The strings $10^7 1$ and $10^{11} 1$ appear (in red) in rows 1 and 2, respectively, and thus are not forbidden.

3.3.3 Algorithm details

The nonnegative integer z is an input parameter chosen to determine the initial assumed string $10^z 1$ in Step 2. In every case tested, we chose $z = d$ or $z = k$.

Each point in the array A has integer coordinates and is marked as either "Unused", or else labeled "0" or "1". The goal is to try to show that the original two assumptions in Step 2 lead to a violation of the hexagonal (d, k) constraint, thus implying that $10^z 1$ is a forbidden string.

A stack is used to keep track of the current state of the algorithm, namely which array positions (i, j) are labeled 0 or 1 and whether they achieved those values by being assumed or by being forced. Each "assume" or "force" instruction in the algorithm is also pushed onto the top of the stack.

In Step 3, the algorithm forces unused points to be 0 if they lie within d positions of a point labeled 1 horizontally, vertically, or diagonally. In other words,

- If $A(x + i, y) =$ Unused and $1 \le |i| \le d$, then force $A(x + i, y) = 0$.
- If $A(x, y + i) =$ Unused and $1 \le |i| \le d$, then force $A(x, y + i) = 0$.
- If $A(x + i, y + i) =$ Unused and $1 \le |i| \le d$, then force $A(x + i, y + i) = 0$.

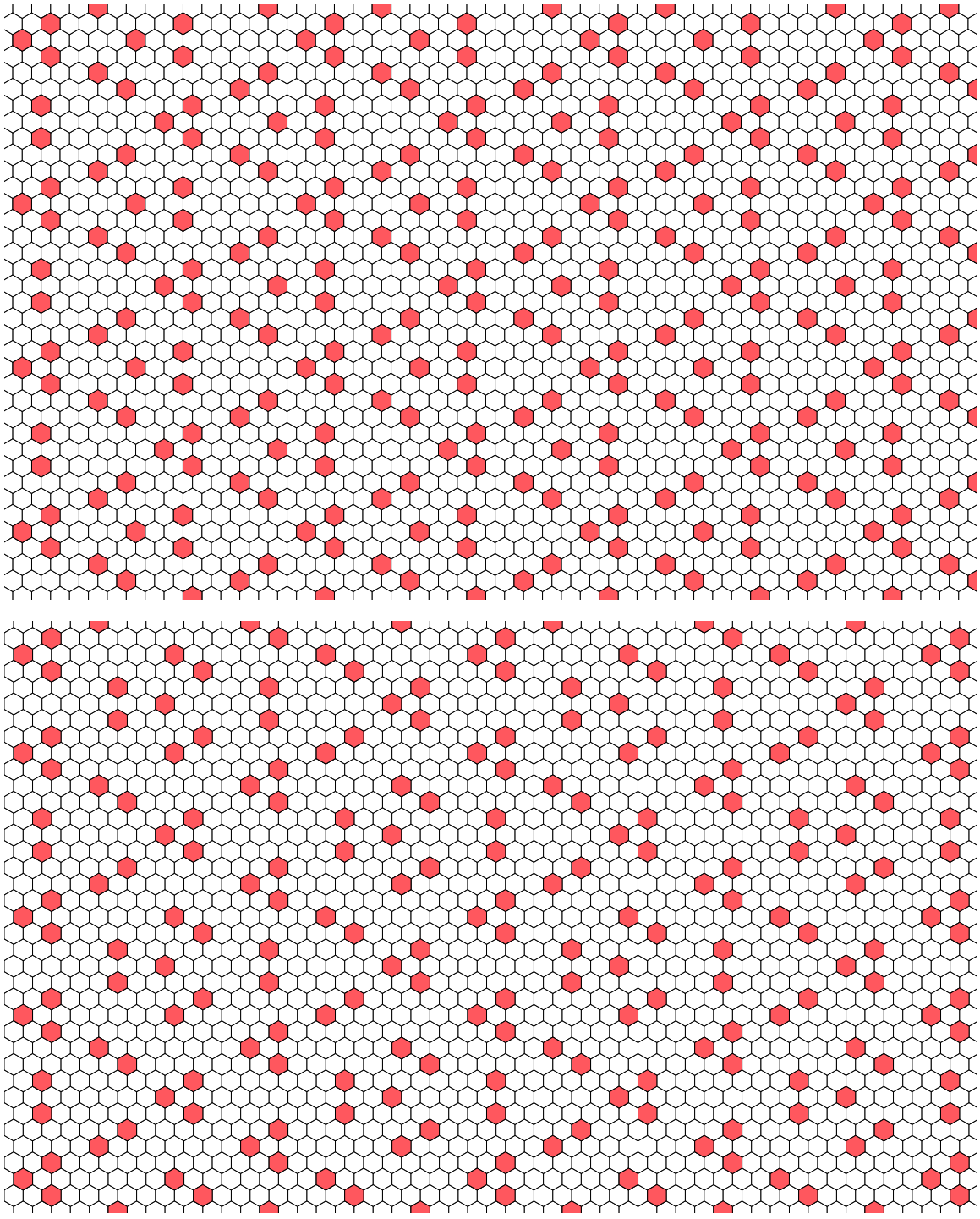


Figure 3.12. The upper labeling satisfies the hexagonal $(5, 8)$ constraint, and the lower labeling satisfies the hexagonal $(7, 11)$ constraint. Red hexagons indicate 1s, and white hexagons indicate 0s. The upper labeling shows $10^5 1$ is not a forbidden string under the hexagonal $(5, 8)$ constraint, and the lower labeling shows $10^7 1$ and $10^{11} 1$ are not forbidden strings under the hexagonal $(7, 11)$ constraint.

Since the algorithm always forces 0s after a 1 is added, the array A is guaranteed to obey the d constraint. If $k < 2d$, then additional 0s can be forced when a 1 is inserted, in order to prevent a violation of the k constraint. Specifically, in addition to points at distances $1, \ldots, d$ in all 3 directions, 0s can be forced at distances $k+2, \ldots, 2d+1$ in all 3 directions, since otherwise, if any of these points were labeled 1, a string of 0^{k+1} would result.

Step 4 checks for violations of the k constraint, and, if found, then removes previously forced bits until the top of the stack is an assumed 1, at which point the assumed 1 is converted to a forced 0. We then repeat Step 4 until the k constraint is enforced. To reduce search complexity, each stack push records the smallest bounding rectangle containing all 0s and 1s in the array up to that point. Only that bounding rectangle, rather than the entire array A, needs to be searched for strings 0^{k+1}.

In Step 5, the algorithm finds some unused position and labels it as an assumed 1. This can be a deterministic process or randomized. We chose to search the array loosely in an order defined by a spiral space filling curve starting at the origin and eventually covering the entire array, but with an additional randomized component.

In practice an $N \times N$ char array is used, with the coordinates indexed from 0 to $N-1$, and N chosen sufficiently large (e.g., $N = 100$ generally suffices). The location $(N/2, N/2)$ is chosen as the origin. If the algorithm ever attempts to assume or force a labeling of a position (x, y) that lies outside the range of the array A, then the algorithm halts and declares a failure to find a proof of a forbidden string. This can occur when there exists a valid labeling of all locations in the array, under the original assumptions.

A recursive implementation of the Forbidden String Algorithm is given in Appendix 3.A, which may be useful for alternate implementations or a formal correctness proof.

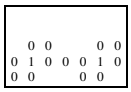
3.3.4 Example for $d = 1$ and $k = 3$

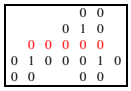
We illustrate in Figure 3.13 the Forbidden String Algorithm with the hexagonal $(1, 3)$ constraint, by showing that 10^31 is a forbidden string, and thus $C_{\text{hex}}(1, 3) = C_{\text{hex}}(1, 2) = 0$. The automated proof is shown on the left-hand side and the labeled bits at various stages on the right-hand side. Each time an "unused" (i.e., not labled) bit is labeled either 0 or 1, that bit labeling is pushed onto a stack and is represented by an extra indentation of the proof line in the figure. Conversely, when a bit is popped off the stack it is marked as "unused" and a reduction of indentation occurs.

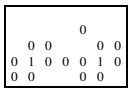
For example, the original two assumed 1s are pushed on the stack at lines 1 and 8 of the proof, at locations $(0, 0)$ and $(4, 0)$, respectively. The forced 0s that resulted from these initial 1s are pushed after each assumption, namely in lines 2–7 and lines 9–14 of the proof. The square diagram between lines 11-14 shows the labeled bits based on the original two assumed 1s.

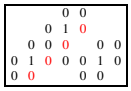
An additional 1 is assumed in line 15 position $(3, 2)$ and its forced 0s follow it in the proof. This assumed

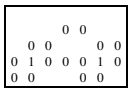
```
 1:  A( 0, 0) = 1 Assumed
 2:    A( 0, 1) = 0 Forced
 3:      A( 0, -1) = 0 Forced
 4:        A(-1, 0) = 0 Forced
 5:          A(-1, -1) = 0 Forced
 6:            A( 1, 1) = 0 Forced
 7:              A( 1, 0) = 0 Forced
 8:                A( 4, 0) = 1 Assumed
 9:                  A( 4, -1) = 0 Forced
10:                    A( 4, 1) = 0 Forced
11:                      A( 3, 0) = 0 Forced
12:                        A( 3, -1) = 0 Forced
13:                          A( 5, 0) = 0 Forced
14:                            A( 5, 1) = 0 Forced
15:                              A( 3, 2) = 1 Assumed
16:                                A( 3, 1) = 0 Forced
17:                                  A( 2, 1) = 0 Forced
18:                                    A( 2, 2) = 0 Forced
19:                                      A( 3, 3) = 0 Forced
20:                                        A( 4, 2) = 0 Forced
21:                                          A( 4, 3) = 0 Forced
22:                                          k-violation in row 1
23:                                        A( 4, 3) = Unused
24:                                      A( 4, 2) = Unused
25:                                    A( 3, 3) = Unused
26:                                  A( 2, 2) = Unused
27:                                A( 2, 1) = Unused
28:                              A( 3, 1) = Unused
29:                              A( 3, 2) = 0 Flipped bit
30:                                A( 2, 2) = 1 Assumed
31:                                  A( 2, 1) = 0 Forced
32:                                    A( 1, 1) = 0 Forced
33:                                      A( 1, 2) = 0 Forced
34:                                        A( 2, 3) = 0 Forced
35:                                          A( 3, 3) = 0 Forced
36:                                          k-violation in diagonal 1
37:                                        A( 3, 3) = Unused
38:                                      A( 2, 3) = Unused
39:                                    A( 1, 2) = Unused
40:                                  A( 1, 1) = Unused
41:                                A( 2, 1) = Unused
42:                                A( 2, 2) = 0 Flipped bit
43:                                  A( 3, 1) = 1 Forced
44:                                    A( 2, 1) = 0 Forced
45:                                      A( 4, 2) = 0 Forced
46:                                      k-violation in diagonal 1
47:                                    A( 4, 2) = Unused
48:                                  A( 2, 1) = Unused
49:                                A( 3, 1) = Unused
50:                              A( 2, 2) = Unused
51:                            A( 3, 2) = Unused
52:                          A( 5, 1) = Unused
53:                        A( 5, 0) = Unused
54:                      A( 3, -1) = Unused
55:                    A( 3, 0) = Unused
56:                  A( 4, 1) = Unused
57:                A( 4, -1) = Unused
58:                A( 4, 0) = 0 Flipped bit
59:                Contradiction
QED
```











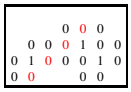


Figure 3.13. Proof automatically generated by the Forbidden String Algorithm. The proof shows that $10^3 1$ is a forbidden string for the hexagonal $(1,3)$ constraint, and thus $C_{\text{hex}}(1,3) = C_{\text{hex}}(1,2) = 0$. The level of indentation indicates the number of assigned binary labels in the plane. Snapshots of the assumed and forced bit values are shown at various points in the proof.

1, however, leads to a horizontal string 0^5 in row 1, as shown in line 22 of the proof. This string violates the fact that $k = 3$, and the violation is illustrated in the square diagram ending at line 22 with the string 0^5 shown in red.

Forced bits are then continually popped off the top of the stack from line 23 through line 28 in the proof, until an assumed 1 is reached. Since this assumed 1 (at location $(3, 2)$) had just led to a contradiction, it is flipped to a forced 0 in line 29, and the resulting bit labelings are shown in the square diagram ending at line 29.

A new 1 is assumed in line 30 at location $(2, 2)$, and its forced 0s follow it in lines 31–35. As a result, a violation of the k constraint occurs due to the string 0^4 occuring diagonally, and this violation is illustrated in the square diagram ending at line 36. Thus the stack pops off forced bits in lines 37–41 until the assumed 1 at $(2, 2)$ is reached. That assumed 1 is converted to a forced 0 in line 42, as illustrated in the diagram ending at line 42.

At this point a 1 is forced at $(3, 1)$ to avoid 0^4 from occuring vertically in column 3. But this 1 forces 0s at $(2, 1)$ and $(4, 2)$, as shown in lines 43–45, thus causing the string 0^4 to occur diagonally, violating the k constraint. This violation is listed in line 46 and shown in the diagram ending at line 46.

Finally, the stack is popped again until the original assumed 1 at $(4, 0)$ is reached, in lines 47–57, and this assumed 1 is converted to a 0 in line 58, as shown in the diagram ending at line 58. This line, however, finishes the proof, since one concludes that the original two assumed 1s lead to a contradiction.

3.3.5 Algorithmic zero capacity results

In searching for forbidden strings of the form 10^d1 or 10^k1, we need not consider cases when d is even or when k is even (by Theorem 3.3.1). The following theorem is thus limited to strings 10^d1 with odd d, or strings 10^k1 with odd k. Each of the strings in the theorem was automatically proven to be forbidden by the Forbidden String Algorithm.

Theorem 3.3.3. *The following are forbidden strings for the indicated hexagonal* (d, k) *constraints:*

d	k	*forbidden string*
3	6	10^d1
4	7	10^k1
6	9	10^k1
7	10	10^d1
8	11	10^k1
9	12	10^d1
10	13	10^k1
11	14	10^d1

Corollary 3.3.4. *The hexagonal (d, k) capacity is zero when $k = d + 3$ and $d \in \{3, 4, 6, 7, 8, 9, 10, 11\}$.*

Proof. It was stated in [13] and proven in our Part I that $C_{\text{hex}}(d, d + 2) = 0$ for all $d > 0$. Thus, Theorem 3.3.3 implies that $C_{\text{hex}}(d, d+3) = C_{\text{hex}}(d+1, d+3) = 0$ when $d \in \{3, 7, 9, 11\}$ and $C_{\text{hex}}(d, d+3) = C_{\text{hex}}(d, d+2) = 0$ when $d \in \{4, 6, 8, 10\}$. ■

We note that in Corollary 3.3.4, the proofs of $C_{\text{hex}}(3, 6) = 0$ and $C_{\text{hex}}(4, 7) = 0$ each relied on the fact that $C_{\text{hex}}(4, 6) = 0$ (which was proven in our Part I), since $10^3 1$ and $10^7 1$ were forbidden strings, respectively. This gives an alternate derivation of $C_{\text{hex}}(3, 5) = C_{\text{hex}}(5, 7) = 0$, from that given in Part I.

The Constant Position Algorithm found each of the first six cases of zero capacity that were found by the Forbidden String Algorithm in Corollary 3.3.4, but it also found more cases that failed for the Forbidden String Algorithm. Thus, the Constant Position Algorithm gives an improvement over the Forbidden String Algorithm in many cases. On the other hand, the Forbidden String Algorithm is simpler to describe, tends to run much faster, and verifies the interesting property of forbidden strings. Also, the Constant Position Algorithm was unable to prove $C_{\text{hex}}(11, 14) = 0$ with the available supercomputer resources, due to an overflow of memory, whereas the Forbidden String Algorithm was able to prove it.

The Forbidden String Algorithm was implemented in C++ on an Apple MacBook Air laptop computer and took at most a couple of seconds to run, up to the third largest case (i.e., $d = 9$, $k = 12$), and took about 30 minutes for the case $(d, k) = (10, 13)$. The largest case (i.e., $d = 11$, $k = 14$) required more memory and took 42 minutes to run on a supercomputer, pushing assumptions on its stack almost ten billion times before finding the proof. Table 3.3 shows some statistics for the Forbidden String Algorithm.

Table 3.3. Complexity statistics for the Forbidden String Algorithm.

d	k	stack pushes	assumed 1s	maximum stack depth
3	6	1,039	111	8
4	7	33,329	1,943	22
6	9	739,886	28,509	16
7	10	1,258,418	39,333	14
8	11	45,060,612	1,194,333	19
9	12	97,422,226	2,229,709	16
10	13	3,856,454,149	78,232,745	22
11	14	9,868,475,943	177,564,097	18

The "stack pushes" column shows the total number of times a push was made on the stack. This quantity reflects the number of times a particular position in the array A was set to either 0 or 1. The "assumed 1s" column shows the total number of times a position in the array A was set to 1 by assumption (i.e., the 1 was not forced).

The "maximum stack depth" column shows the largest number of assumed 1s that were ever on the stack at a single time.

3.4 Rectangle Tiling Algorithm for proving positive hexagonal (d, k) capacity

The following algorithm attempts to automatically discover distinct labelings of $M \times N$ rectangular arrays A and B which are valid no matter how they (jointly) tile the plane, if such rectangle labelings exist. For any i and j, let $A(i, j)$ and $B(i, j)$ denote the value of the binary labeling of position (i, j) in these two arrays, respectively.[4]

The main idea is to randomly label some of the unlabled positions in the rectangles with a 1s, and then fill in all the resulting forced 0s, always checking that the d and k constraints are met. If a constraint is violated, then such assumed labeled positions can be reversed with backtracking.

3.4.1 Algorithm description

Rectangle Tiling Algorithm

Step 1: Set $A(i, j) = B(i, j) =$ Unused, for all i, j.

Step 2: Choose $A(1, 1) = 1$ and $B(1, 1) = 0$.

Step 3: Force 0s for the d constraint in A and B for all tiling configurations.

Step 4: If there does not exist 0^{k+1} horizontally, vertically, or diagonally, for any tiling configurations, then go to Step 5.

Else repeatedly pop the stack until the top of the stack is a chosen 1.

If the top of the stack is the original choice $A(1, 1) = 1$, then failure, so exit.

Else convert the top of the stack to a forced 0 and repeat Step 4.

Step 5: If there are no unused positions in A and B, then the algorithm succeeded, so exit.

Step 6: Select an unused position (x, y) in A or B and set it to 1.

Step 7: Go to Step 3.

[4] Matrix notation is used here, where i is the row (increasing downward from 1 to M) and j is the column (increasing rightward from 1 to N).

3.4.2 Algorithm details

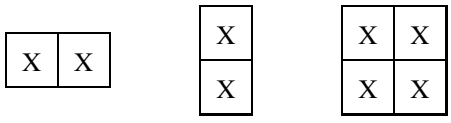
The two arrays A and B are implemented as two-dimensional char arrays in C++, each of size $M \times N$, where M and N are fixed positive integers. Step 1 initializes all MN positions in each of A and B. Step 2 chooses one position where A and B are forced to differ, in order to ensure the two rectangles are different. The upper-left corner of the arrays is one possible choice, but other locations are also feasible, and possibly advantageous. Step 3 enforces the d constraint by forcing nearby 0s. Step 4 enforces the k constraint. If it is violated, then chosen 1s are continually removed from the stack (and converted to forced 0s), until there is no longer a violation. Step 6 tries to find a new position to label 1. Various strategies can be used for this choice, or just a simple random selection.

While the Rectangle Tiling Algorithm shares some features with the Forbidden String Algorithm, they have fundamentally different objectives. The former tries to find two different valid rectangles, whereas the latter tries to find a contradiction to the assumption of a string of the form 10^z1 existing in some valid labeling.

There are nuances to how the two algorithms enforce constraints as well. In the Forbidden String Algorithm, enforcing the d constraint is rather straightforward, but in the Rectangle Tiling Algorithm, many different possible configurations of the rectangles A and B must be considered, since any chosen 1 in one of the two arrays can cause forced 0s in itself or the other array in multiple ways, depending on how they tile the plane. Similarly, checking for violations of the k constraint in the Forbidden String Algorithm is straightforward, although somewhat laborious, whereas in the Rectangle Tiling Algorithm, many different tiling configurations must be examined.

The Rectangle Tiling Algorithm is not guaranteed to terminate. This can happen if there do not exist two rectangular labelings that tile the plane according to the desired constraint, or possibly due to failure to find such rectangular labelings even if they exist.

If the Rectangle Tiling Algorithm does terminate, one can check the two rectangles produced to verify that they do not violate the constraints for arbitrary tilings of the plane. For example, if $M = N > k$, then it suffices to check the following configurations for violations of constraints:



where the first configuration is checked for horizontal violations, the second configuration for vertical violations, and the third configuration for diagonal violations.

In each configuration, all possible assignments of the squares A and B to the locations denoted by "X" are constructed (i.e., 4 arrangements for each of the first two configurations and $2^4 = 16$ for the third configuration).

The two algorithms also present different space complexity concerns. While both algorithms use a stack to perform backtracking so that bad choices of labelings can be corrected, the stack can grow without bound in the Forbidden String Algorithm (or at least up to the area of the bounding array used, which can be very large), whereas in the Rectangle Tiling Algorithm, the stack can never be larger than $2MN$, and if it fills to that value, then the algorithm has successfully completed its task.

A recursive implementation of the Rectangle Tiling Algorithm is given in Appendix 3.B, which may be useful for alternate implementations or a formal correctness proof.

3.4.3 Enforcing the d and k constraints

In Step 3 of the Rectangle Tiling Algorithm, the d constraint is enforced by labeling positions in the arrays A and B zero if they are within distance d of a chosen 1, for any configuration of A and B in a tiling. In Step 4 of the Rectangle Tiling Algorithm, the k constraint is enforced by hunting for violating strings of the form 0^{k+1} in arbitrary configurations of A and B in a tiling. Enforcing the k constraint is considerably more time-consuming that the d constraint, but both processes share the common feature of examining arbitrary tilings by labelings of A and B.

In order to efficiently be able to enforce these two constraints, we use the graphical representation of arbitrary tilings described above. Each of the three graphs G_{hori}, G_{vert}, G_{diag} is used to check all directed paths of length d from the position of the chosen 1, in the horizontal, vertical, and diagonal directions. For any nodes on such paths that are marked as "Unused", a 0 is forced for the corresponding position and array. Then the whole process is repeated for the reverse-direction-edge versions of the three graphs.

For example, if A and B are 5×5 squares and $(d, k) = (1, 5)$, Figure 3.14 shows in the far left all the forced 0s in both A and B due to the assumptions that $A(1, 1) = 1$ and $B(1, 1) = 0$.

Let us denote certain data structures by $A_{i,j}$ and $B_{i,j}$, corresponding to each position (i, j) of $M \times N$ arrays A and B, respectively. These data structures contain the binary value labeling the position, or else an indication that it is "unused". We define the following three directed graphs to enable a reduced complexity search for potential violations of the k constraint when the arrays A and B are partially labeled:

$$G_{\text{hori}} = (V, E_{\text{hori}})$$

$$G_{\text{vert}} = (V, E_{\text{vert}})$$

$$G_{\text{diag}} = (V, E_{\text{diag}}).$$

The vertex set V and directed edge sets E_{hori}, E_{vert}, E_{diag} are given by:

$$
\begin{aligned}
V &= \bigcup_{i=1}^{M} \bigcup_{j=1}^{N} \{A_{i,j}, B_{i,j}\} \\
E_{\text{hori}} &= \bigcup_{i=1}^{M} \left(\bigcup_{j=1}^{N-1} \{(A_{i,j}, A_{i,j+1}), (B_{i,j}, B_{i,j+1})\} \right. \\
&\qquad \left. \cup \{(A_{i,N}, A_{i,1}), (A_{i,N}, B_{i,1}), (B_{i,N}, B_{i,1}), (B_{i,N}, A_{i,1})\} \right) \\
E_{\text{vert}} &= \bigcup_{j=1}^{N} \left(\bigcup_{i=1}^{M-1} \{(A_{i,j}, A_{i+1,j}), (B_{i,j}, B_{i+1,j})\} \right. \\
&\qquad \left. \cup \{(A_{M,j}, A_{1,j}), (A_{M,j}, B_{1,j}), (B_{M,j}, B_{1,j}), (B_{M,j}, A_{1,j})\} \right) \\
E_{\text{diag}} &= \bigcup_{i=2}^{M} \bigcup_{j=1}^{N-1} \{(A_{i,j}, A_{i-1,j+1}), (B_{i,j}, B_{i-1,j+1})\} \\
&\quad \cup \bigcup_{i=2}^{M} \{(A_{i,N}, A_{i-1,1}), (A_{i,N}, B_{i-1,1}), (B_{i,N}, B_{i-1,1}), (B_{i,N}, A_{i-1,1})\} \\
&\quad \cup \bigcup_{j=1}^{N-1} \{(A_{1,j}, A_{M,j+1}), (A_{1,j}, B_{M,j+1}), (B_{1,j}, B_{M,j+1}), (B_{1,j}, A_{M,j+1})\} \\
&\quad \cup \{(A_{1,N}, A_{M,1}), (A_{1,N}, B_{M,1}), (B_{1,N}, A_{M,1}), (B_{1,N}, B_{M,1})\}.
\end{aligned}
$$

For the horizontal graph G_{hori}, all of the nodes corresponding to positions of A and B, except for positions in the far right column, have exactly one out-edge, namely to the next position to the right in the same array. Each of the positions in the far right column has two out-edges, pointing to the positions in A and B of the same row, but in the far left column.

For the vertical graph G_{vert}, all of the nodes corresponding to positions of A and B, except for positions in the bottom row, have exactly one out-edge, namely to the next position down in the same array. Each of the positions in the bottom row has two out-edges, pointing to the positions in A and B of the same column, but in the top row.

For the diagonal graph G_{diag}, all of the nodes corresponding to positions of A and B, except for positions in the top row and right column, have exactly one out-edge, namely to the next position to the right and up in the same array. The top row and right column positions have two out-edges each, directed to the similarly diagonally-adjacent positions in A and B.

These three graphs facilitate searching for the string 0^{k+1} horizontally, vertically, or northeast diagonally.

Any horizontal (respectively, vertical or diagonal) run of 0s of length z corresponds to a path[5] of length z through the graph G_{hori} (respectively, G_{vert} or G_{diag}).

One way to search for any possible occurrence of 0^{k+1} is to assume such a run starts at a particular location (x, y) in either A or B, and then search for a path of length $k+1$, labeled by 0s, in either G_{hori}, G_{vert}, or G_{diag}. This, however, repeats work unnecessarily, since adjacent starting positions may share substantial portions of runs.

A more efficient technique, which we use, is to first find, starting at each position (x, y) of A, the longest path in G_{hori} labeled entirely by 0s and the longest path in the reversed-edge-direction version of G_{hori} labeled entirely by 0s. This will yield the longest horizontal run of 0s in A that passes through the position (x, y). This process is then repeated for each position of B. With such a technique, duplicated work is avoided and the longest runs of 0s passing through each position of A and B are determined. If any of these runs have length greater than k, then the process can be terminated immediately and a horizontal violation of the k constraint can be declared. If no such violation is discovered, this process is repeated for the graph G_{vert}, and then again for G_{diag}. If no run of k zeros is found in any of the three graphs, then the current labeling is declared to not (yet) violate the k constraint.

3.4.4 Example for $d = 1$ and $k = 5$

We illustrate in Figure 3.14 the Rectangle Tiling Algorithm with the hexagonal $(1, 5)$ constraint, by constructing two 5×5 distinct labelings that satisfy the constraint no matter how they tile the plane. Thus $C_{\text{hex}}(1, 5) > 1/25$. The stack is shown at 6 different stages, after choosing 1s and after popping due to violations of the k constraint. Below each stack is the current labeling of two squares A and B, with unlabeled portions left blank.

In the first stack snapshot shown in Figure 3.14, the original two chosen values are shown (a 1 and a 0), and the resulting forced 0s from the chosen 1. The stack grows upward, and each line stores the coordinates of the labeled point, the value of the label, which of the two arrays (A or B) it lies in, and whether the value was chosen or forced. It can be seen that in the fifth snapshot of the stack, a violation of the k constraint is discovered (in red) as the string 0^5 on the main diagonal of array B. The reason this is a violation is that no row, column, or main northeast diagonal can be all 0s, for otherwise, a tiling in the plane of two such squares in a diagonal configuration will result in a diagonal string 0^{10}. Also, note that in the fourth stack snapshot, it is possible to discover that $B(2, 4) = 1$ is forced, in order to prevent the k-violation described in the fifth stack snapshot. However, not all such forced values are found in practice, and implementing such procedures leads to a tradeoff between the time spent hunting for such forced values and the benefit of finding them sooner, rather than later.

[5]We assume the vertices in any path are distinct.

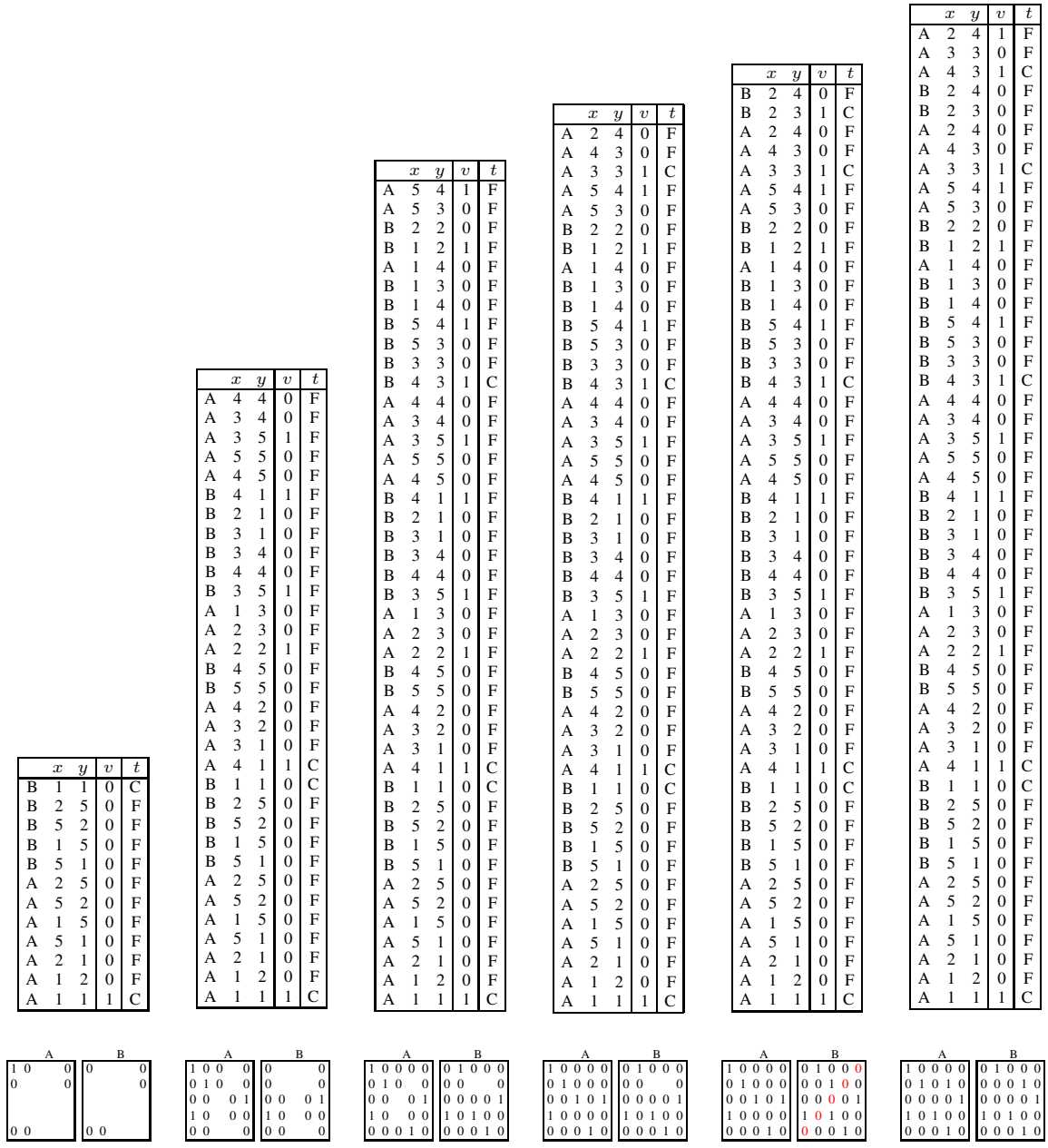


Figure 3.14. Snapshots of the stack during the Rectangle Tiling Algorithm. The stack grows upward, and each line in the stack shows which square (A or B) is modified, the location (x, y) of modification, the binary value v, and the type of change t ("C" for Chosen, or "F" for Forced). The coordinates are given in matrix notation, namely (row, column), where the upper-left corner is position $(1, 1)$.

3.4.5 Algorithmic positive capacity results

We include here a proof of five positive hexagonal (d, k) capacities. The first one, $C_{\text{hex}}(0, 1) > 0$, is rather trivial but is included for completeness since the only other known proof, due to Baxter and Joyce in more general form as discussed earlier, is extremely complicated. The other four cases have been stated previously but never proven in the literature.

In each case we demonstrate positive capacity by exhibiting a pair of distinct square labelings which can arbitrarily tile the plane without violating the corresponding constraint. Thus each square labeling carries with it one bit of information content.

To find these tileable square labelings, we wrote a backtracking algorithm that starts with two blank squares and attempts to fill in 1s in randomly chosen unoccupied positions, along with any 0s that are then forced due to the d constraint. If a contradiction results from trying to place a 0 where a 1 already exists, or vice versa, then the previous action is popped off of a stack, and new moves are attempted. Successive pushing and popping of the stack eventually led to the results shown. Computer run times were typically on the order of several minutes.

We note that it is an open question whether pairs of labelings exist that tile the plane validly whenever a capacity is positive. Alternatively, it is conceivable that only aperiodic tilings of the plane can demonstrate positive capacity. In fact, Durand, Gamard, Grandjean [8] demonstrated the existence of a certain Wang tile set that can only tile the plane aperiodically, and yet achieves a positive capacity, although their results do not apply directly to the hexagonal (d, k) constraint situation.

Theorem 3.4.1. *If* $(d, k) \in \{(0, 1), (1, 4), (2, 5), (3, 7), (4, 9)\}$*, then the hexagonal* (d, k) *capacity is positive.*

Proof. For each constraint, a square is given that can take on two distinct labelings by choosing the value of the indicated x to be either 0 or 1, where $\overline{x} = 1 - x$ (see Figures 3.15 – 3.19). One can verify by inspection that the resulting two labelings for each constraint can be arbitrarily assigned to squares in a tiling of the plane by the squares, without violating the relevant hexagonal (d, k) constraint. Thus the area occupied by one such square can have any one of at least two possible labelings, so the capacity is lower bounded by the reciprocal of the area of the square, which in particular is positive. ■

1	x
1	1

Figure 3.15. Two distinct labelings of a 2×2 square for the hexagonal $(0, 1)$ constraint where $x \in \{0, 1\}$. Thus, $C_{\text{hex}}(0, 1) \geq 1/4$.

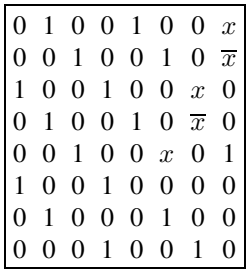


Figure 3.16. Two distinct labelings of an 8×8 square for the hexagonal $(1, 4)$ constraint where $x \in \{0, 1\}$. Thus, $C_{\text{hex}}(1, 4) \geq 1/64$.

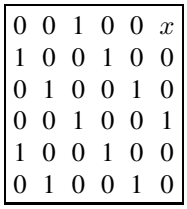


Figure 3.17. Two distinct labelings of a 6×6 square for the hexagonal $(2, 5)$ constraint where $x \in \{0, 1\}$. Thus, $C_{\text{hex}}(2, 5) \geq 1/36$.

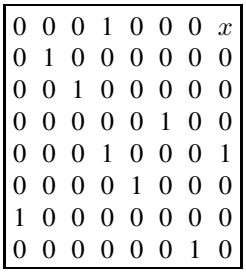


Figure 3.18. Two distinct labelings of an 8×8 square for the hexagonal $(3, 7)$ constraint where $x \in \{0, 1\}$. Thus, $C_{\text{hex}}(3, 7) \geq 1/64$.

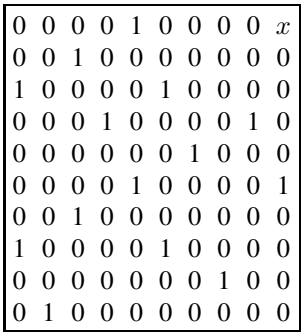


Figure 3.19. Two distinct labelings of a 10×10 squares for the hexagonal $(4, 9)$ constraint where $x \in \{0, 1\}$. Thus, $C_{\text{hex}}(4, 9) \geq 1/100$.

3.A Recursive implementation of the Forbidden String Algorithm

```
int A[N][N];
int Origin = N/2;

for(i,j=1 to N) A[i][j] = Unused;

Stack<Cell> TheStack;

struct Cell {
  int X, Y, value;
  string type;
}

bool Assign( X, Y ) {
  A[X][Y] = 1;

  Cell c;
  c.X = X; c.Y = Y;
  c.type = "assumed";
  c.value = 1;
  TheStack.push( c );

  ForceZeros();

  if( No_k_Violation() )
      if( NoUnusedLocations() ) return false; // Ran out of memory.

  else {
      do{ A[TheStack.top.X][TheStack.top.Y] = Unused;
          TheStack.pop() }
      while( !(TheStack.top.type = "assumed" && TheStack.top.value == 1 ));

      if(TheStack.top.X == Origin && TheStack.top.Y == Origin )
          return true; // Found proof that the string 1 0^z 1 is forbidden.
      else {
          A[TheStack.top.X][TheStack.top.Y] = 0;
          TheStack.top.value = 0;
          TheStack.top.type = "forced";
      }
  }

  (X,Y) = NewUnusedLocation();
  return Assign( X, Y ); // Try assuming another unused location is 1.
}

A( Origin + z + 1, Origin ) = 1;

if( Assign( Origin, Origin ) ) then print "Success";
else print "Failure";
```

3.B Recursive implementation of the Rectangle Tiling Algorithm

```
int A[N][M], B[N][M];
for(i=1 to N, j=1 to M) A[i][j] = B[i][j] = Unused;

Stack<Cell> TheStack;

struct Cell {
  int X, Y, value;
  string type;
  int** Rect;
}

bool Tile( WhichRectangle, X, Y ) {
  WhichRectangle[X][Y] = 1;

  Cell c;
  c.X = X; c.Y = Y;
  c.Rect = WhichRectangle;
  c.type = "chosen";
  c.value = 1;
  TheStack.push( c );

  ForceZeros();

  if( No_k_Violation() )
      if( RectanglesFull() ) return true; // Found good rectangles.

  else {
      do{ TheStack.top.Rect[TheStack.top.X][TheStack.top.Y] = Unused;
          TheStack.pop() }
      while( !(TheStack.top.type = "chosen" && TheStack.top.value == 1 ));

      if(TheStack.top.X == 1 && TheStack.top.Y == 1 && TheStack.top.Rect == A )
          return false; // Original assumptions led to contradiction.
      else {
          TheStack.top.Rect[TheStack.top.X][TheStack.top.Y] = 0;
          TheStack.top.value = 0;
          TheStack.top.type = "forced";
      }
  }

  Rect = NewUnfilledRectangle;
  (X,Y) = NewUnusedLocation( Rect );
  return Tile( Rect, X, Y ); // Try filling in another 1 in one rectangle.
}

B(1,1) = 0;

if( Tile( A, 1, 1) ) then print "Success";
else print "Failure";
```

Chapter 3 is a reprint of the material as it appears in: S. Congero and K. Zeger, "Hexagonal run-length zero capacity region—Part II: Automated proofs", *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 153-177, January 2022.

References

[1] R. J. Baxter, "Hard hexagons: exact solution" *Journal of Physics A* vol. 13, pp. 1023 – 1030, 1980.

[2] R. J. Baxter, *Exactly Solved Models in Statistical Mechanics*, Academic Press, 1982.

[3] R. J. Baxter, "Planar lattice gases with nearest-neighbour exclusion," *Annals Combinatorics 3*, vol. 191, pp. 191 – 203, 1999.

[4] R.J. Baxter and S.K. Tsang, "Entropy of hard hexagons," *J. Phys. A (Math. Gen)* vol. 13, pp. 1023 – 1030, 1980.

[5] N. J. Calkin and H. S. Wilf, "The number of independent sets in a grid graph," *SIAM Journal of Discrete Math*, vol. 11, pp. 54 – 60, February 1998.

[6] K. Censor and T. Etzion, "The positive capacity region of two-dimensional run-length-constrained channels," *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 5128 – 5140, November 2006.

[7] S. Congero and K. Zeger, "Hexagonal run-length zero capacity region—Part I: Analytical proofs," *IEEE Transactions on Information Theory* (submitted September 21, 2020, revised August 22, 2021).

[8] B. Durand, G. Gamard, and A. Grandjean, "Aperiodic tilings and entropy," *International Conference on Developments in Language Theory,* pp. 166 – 177, Springer, 2014.

[9] K. S. Immink, *Codes for Mass Data Storage Systems*, second edition, Shannon Foundation Publishers, Eindhoven, The Netherlands, 2004.

[10] G. S. Joyce, "Exact results for the activity and isothermal compressibility of the hard-hexagon model," *Journal of Physics A: Mathematical and General*, vol. 21 (20): pp. L983 – L988, 1988.

[11] G. S. Joyce, "On the hard hexagon model and the theory of modular functions," *Philosophical Transactions of the Royal Society of London A*, vol. 325, pp. 643 – 702, 1988.

[12] A. Kato and K. Zeger, "On the capacity of two-dimensional run length constrained channels," *IEEE Transactions on Information Theory*, vol. 45, no. 4, pp. 1527 – 1540, July 1999.

[13] Zs. Kukorelly and K. Zeger, "The capacity of some hexagonal (d, k) constraints", *IEEE International Symposium on Information Theory (ISIT)*, Washington, D.C., p. 64, June 2001.

[14] Zs. Kukorelly and K. Zeger, "Automated theorem proving for hexagonal run length constrained capacity computation", *IEEE International Symposium on Information Theory (ISIT)*, Seattle, Washington, July 2006.

[15] B.D. Metcalf and C.P. Yang, "Degeneracy of anti-ferromagnetic Ising lattices at critical magnetic field and zero temperature," *Physical Review B* vol. 18, pp. 2304 – 2307, 1978.

[16] Zs. Nagy and K. Zeger, "Capacity bounds for the three-dimensional run length limited channel," *IEEE Transactions on Information Theory*, vol. 46, pp. 1030 – 1033, May 2000.

[17] L. Onsager, "Crystal statistics. I. A two-dimensional model with an order-disorder transition," *Physical Review*, vol. 65, nos. 3 and 4, pp. 117 – 149, 1944.

[18] R. Pavlov, "Approximating the hard square entropy constant with probabilistic methods," *Annals of Probability*, vol. 40, no. 6, pp. 2362 – 2399, 2012.

[19] M. Schwartz and A. Vardy, "New bounds on the capacity of multidimensional run-length constraints," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4373 – 4382, July 2011.

[20] A. Sharov and R. M. Roth, "Two-dimensional constrained coding based on tiling," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1800 – 1807, April 2010.

[21] R.E. Tarjan, “Depth first search and linear graph algorithms,” *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146 – 160, 1972.

[22] G.H. Wannier, “Antiferromagnetism. The triangular Ising net,” *Physical Review*, vol. 79, no. 2, pp. 357 – 364, 1950.

[23] W. Weeks and R. E. Blahut, “The capacity and coding gain of certain checkerboard codes”, *IEEE Transactions on Information Theory*, vol. 44, pp. 1193 – 1203, May 1998.

Chapter 4

The 3/4 Conjecture for Fix-Free Codes with At Most Three Distinct Codeword Lengths

Abstract

The $3/4$ Conjecture was posed 25 years ago by Ahlswede, Balkenhol, and Khachatrian, and states that if a multiset of positive integers has Kraft sum at most $3/4$, then there exists a code that is both a prefix code and a suffix code with these integers as codeword lengths. We prove that the $3/4$ Conjecture is true whenever the given multiset of positive integers contains at most three distinct values.

4.1 Background on fix-free codes

One of the most intriguing unsolved questions in information theory is the so-called "3/4 Conjecture" for fix-free codes. The conjecture was posed 25 years ago by Ahlswede, Balkenhol, and Khachatrian, and states that if a multiset of positive integers has Kraft sum at most $3/4$, then there exists a code that is both a prefix code and a suffix code with these integers as codeword lengths. This conjecture is analogous to the well-known fact that if a multiset of positive integers has Kraft sum at most 1, then there exists a prefix code with these integers as codeword lengths.

In this paper, we prove that the $3/4$ Conjecture is true whenever the given multiset of positive integers contains at most three distinct values.

Our proof technique is partially constructive and partially existential, the latter approach relying on a random coding argument, similar in spirit to that used in the classical channel coding theorem of Shannon [60].

For any two binary words u and v, let uv denote their concatenation. Let ϵ denote the empty word, such that $\epsilon u = u\epsilon = u$ for any binary word u. Denote the binary alphabet by $A = \{0, 1\}$. Let $A^0 = \{\epsilon\}$, and for each $n \geq 1$ let A^n denote the set of all n-bit binary words. Also, let $A^* = \cup_{n=0}^{\infty} A^n$ be the set of all finite-length binary words. For any sets $S, T \subseteq A^*$, denote their direct product by $ST = \{uv : u \in S, v \in T\}$. Note that $\varnothing T = T\varnothing = \varnothing$ vacuously. For any binary word $u \in A^*$, let $|u|$ denote its length.

A *code* is a finite subset of A^*, and a code's elements are called *codewords*. A word $u \in A^*$ is a *prefix* (respectively, *suffix*) of a word $v \in A^*$ if there exists $x \in A^*$ such that $v = ux$ (respectively, $v = xu$).

A *prefix code* (respectively, *suffix code*) is a code for which no codeword is a prefix (respectively, suffix) of any other codeword. A *fix-free code*[1] is a code that is both a prefix code and a suffix code.

If C is a code, then CA^* (respectively, A^*C) is the set of all words having a prefix (respectively, suffix) in C.

A *pattern* is a code described by a string in $\{0, 1, A\}^*$. The code consists of all possible words obtained by assigning either 0 or 1 to each occurrence of A in the pattern's string. For example, $110A1A^20$ is a pattern that contains $|11A0A^20| = 2^3 = 8$ strings, each of length 7, namely

$$\{1100000, 1100010, 1100100, 1100110, 1110000, 1110010, 1110100, 1110110\}.$$

Also note that the length-one patterns 0 and 1 are the sets $\{0\}$ and $\{1\}$, respectively.

The *multiplicity* of an integer in a multiset is the number of occurrences of that integer in the multiset. If

[1] Fix-free codes have also been called *"biprefix codes"* (e.g., [7, 49–52, 54]), *"bifix codes"* (e.g., [6, 8–12]), *"affix codes"* (e.g., [21, 53]), *"reversible variable length codes"* (e.g., [5, 27, 30, 34, 45, 61, 63–73, 82]), and *"never-self-synchronizing codes"* (e.g., [23]).

a multiset of positive integers has distinct integers $\lambda_1, \lambda_2, \ldots$ with corresponding multiplicities $\mu_1, \mu_2, \ldots$, then the *Kraft sum* of the multiset is the quantity

$$\sum_{n \geq 1} \mu_n 2^{-\lambda_n}$$

and the *Kraft sum* of a code C is the quantity

$$\mathcal{K}(C) = \sum_{u \in C} 2^{-|u|}.$$

Note that the Kraft sum of any pattern $U \in \{0, 1, A\}^p$ is $\mathcal{K}(U) = |U|/2^p$, where $|U|$ equals 2 raised to the number of As in U.

As an example, the multiset $\{2, 3, 3, 4, 4, 4, 4\}$ has distinct lengths 2, 3, and 4, with corresponding multiplicities $\mu_1 = 1$, $\mu_2 = 2$, $\mu_3 = 4$, and its Kraft sum is $1 \cdot 2^{-2} + 2 \cdot 2^{-3} + 4 \cdot 2^{-4} = 1$.

Variable length codes have been successfully used for transmission and storage of information for at least 75 years. In particular, binary prefix codes have been the most commonly used variable length codes, and are widely embedded in many practical communication systems, such as speech, image, and video coding standards.

Prefix codes have been extensively studied and are well understood both in theory and practice. The existence of prefix codes with a given set of codeword lengths was characterized by Kraft [43] in 1949, and an optimal construction algorithm was given by Huffman [29] in 1952 that finds prefix codes with minimum average length with respect to a source distribution.

A fix-free code is a special type of prefix code, namely one that is also a suffix code. Fix-free codes have been studied for primarily four reasons: (1) theoretical and algebraic properties; (2) data compression; (3) error correction; (4) sufficient conditions for existence using Kraft-type inequalities.

Theoretical analyses of fix-free codes were originated in 1956 by Schützenberger [57] and in 1959 by Gilbert and Moore [23]. Various other algebraic properties were given from the 1960s to 1980s by Berstel and Perrin [7], Césari [16, 17], Leonard [47], Perrin [49–52], Reutenauer [54], and Schützenberger [58, 59], and more recently by Berstel, Berthe, DeFelice, Dolce, Leroy, Perrin, Reutenauer, and Rindone [8–12] and Gillman and Rivest [24].

A special case of a fix-free code is a *palindromic* (or "symmetric") code which is defined as a prefix code, all of whose codewords are palindromes. Constructions of such codes were considered in [1,3,26,55,61,63,64,74].

In 1995, Takishima, Wada, and Murakami [61] studied fix-free codes for providing error correction capability by decoding both in the forward and reverse directions. Numerous other studies applying such codes to error correction appeared later (e.g., [5, 15, 22, 25, 27, 30, 34, 35, 45, 46, 48, 62–73, 82]). In fact, the practical application

of fix-free codes was adopted into international standards for video compression, including ISO MPEG-4 [31] in 1998 and ITU-T H.263+ [32] in 2000.

In terms of data compression, prefix codes achieving a minimum possible average length with respect to a given source distribution are well known from Huffman's algorithm [29]. For fix-free codes, the situation is a bit more complicated. Some studies of this include [1, 33, 36, 37, 39, 41, 42, 55, 74, 77, 80, 81].

In addition to the practical use of fix-free codes for error correction of variable length lossless codes, the foundational theory of fix-free codes has been a topic of great interest.

In order for any variable length code to be useful, it is generally required that it be uniquely decodable (UD), which means that there is only one way to correctly parse a concatenation of variable length codewords. Prefix codes are always UD, and it is known that for every UD code, there exists a prefix code with the same codeword lengths [18]. So there is no loss of generality in restricting one's attention from general UD variable length codes to prefix codes.

On the other hand, for the purpose of lossless data compression, one would like the average codeword length to be as short as possible, in order to reduce transmission and storage costs. This assumes each codeword is assigned to represent a particular outcome of a discrete source random variable. The desire to have codes be UD and short on average are opposing needs. That is, if a code is too short, it cannot also be UD.

The Kraft inequality makes this idea quantitatively precise. Specifically, the Kraft inequality gives an upper bound of 1 on the Kraft sum of a multiset of positive integers corresponding to the codeword lengths of a prefix code. In other words, as long as this upper bound is not violated, a prefix code exists having those positive integers as its codeword lengths. In fact, the converse to the Kraft theorem is also true, namely that the Kraft sum of the codeword lengths of any prefix code can be at most 1.

For fix-free codes, a similar trade-off exists between having short codeword lengths and being both a prefix and a suffix code. An analogous question to the prefix code case asks for the lowest possible upper bound on the Kraft sum of a multiset of positive integers that would ensure the existence of a fix-free code having those positive integers as its codeword lengths. No improved converse can exist however, since fix-free codes can indeed have Kraft sum equal to 1, such as a code consisting of all codewords of a given length.

In 1996, Ahlswede, Balkenhol, and Khachatrian [4] showed the weaker result that if the Kraft sum is at most $1/2$, then a fix-free code is guaranteed to exist with the corresponding codeword lengths. They also showed the existence of a fix-free code if the Kraft sum of the multiset of codeword lengths is at most $3/4$ and each integer in the multiset is at least twice any smaller integer in the multiset. More generally, they showed that any upper bound on the Kraft sum that ensures the existence of a fix-free code cannot be larger than $3/4$. Perhaps most interestingly, the authors of [4] conjectured that $3/4$ itself is in fact such an upper bound on the Kraft sum. This is

now commonly referred to as the "$3/4$ Conjecture", and is stated next.

Conjecture 4.1.1 (Ahlswede, Balkenhol and Khachatrian [4])**.** *If a multiset of positive integers has Kraft sum at most $3/4$, then there exists a fix-free code whose codeword lengths are the elements of the multiset.*

Since the $3/4$ Conjecture was made, numerous attempts to prove it have failed. However, many interesting special cases of the conjecture have been proven, which we review next.

In 1999, Harada and Kobayashi [28] proved that Conjecture 4.1.1 holds if the multiset contains at most two distinct positive integers. Initially, they attempted to use a randomized algorithm in their proof, but demonstrated that it is not guaranteed to find the desired fix-free code. To achieve their result, they used a deterministic algorithm. They were unable to extend their methods beyond multisets containing at most two distinct positive integers and, in fact, stated the following:

> *"However, finding a fix-free code for $l_1, \ldots, l_n$ which consists of three or more different lengths seems not to be always easy."*

In 2012, Savari, Yazdi, Abedini, and Khatri [55, p. 5121] proved, among other things, one special case of Conjecture 4.1.1 where the given multiset has three distinct values. In particular, their result is limited to the case where the smallest such value is 2 and appears exactly once, and the remaining two values have further specific restrictions. The authors also stated the following that acknowledges the Harada-Kobayashi result for multisets with two distinct values and confirms the difficulty of proving Conjecture 4.1.1 for multisets containing three distinct values:

> *"The progress on the $3/4$ conjecture has been slow even over binary code alphabets. One of the early results [by Harada-Kobayashi] is that the $3/4$ conjecture holds for length sequences $(l_1, \ldots, l_n)$ for which $l_i \in \{\lambda_1, \lambda_2\}$ for all i . The case where $l_i \in \{\lambda_1, \lambda_2, \lambda_3\}$ is only partly understood."*

It is precisely the proof of Conjecture 4.1.1 for at most three distinct integers that we achieve in the present paper (in our Theorem 4.2.1).

In 2001, Ye and Yeung [75] proved that Conjecture 4.1.1 holds when the multiset values do not exceed 7. They also proved the weaker result that a fix-free code exists when the multiset contains the integer 1 and the Kraft sum is at most $5/8$.

Also in 2001, Yekhanin [76] gave a proof sketch that Conjecture 4.1.1 holds in two different cases: (1) when the multiset values do not exceed 8; or (2) when the Kraft sum of the submultiset of is and $(i+1)$s is at least $1/2$, where i is the smallest integer in the multiset. A special case of this second result is stated as the following theorem, which we use as one component of our main result, Theorem 4.2.1. Theorem 4.1.2 is proved in more detail in Yekhanin's unpublished notes in [78].

Theorem 4.1.2. *Conjecture 4.1.1 holds when the Kraft sum of the multiset of smallest length words is at least* $1/2$.

In 2004, Yekhanin [77], also proved Conjecture 4.1.1 holds when the Kraft sum is at most $5/8$.

In 2005, Kukorelly and Zeger [44] proved that Conjecture 4.1.1 holds in two different cases: (1) when the minimum integer i in the multiset is at least 2, and no integer in the multiset, except possibly the largest one, occurs more than 2^{i-2} times; or (2) when every integer in the multiset, except possibly the largest one, occurs at most twice.

In 2007, Schnettler [56] (see also [19, 20, 40]) gave a survey of sufficient conditions for the existence of fix-free codes and generalized to nonbinary alphabets the result described above in [44]. He also expanded the proof sketch given in [76] to a more general version of Theorem 4.1.2, and proved several specialized cases of Conjecture 4.1.1.

In 2008, Khosravifard and Gulliver [38] further studied and improved the algorithm used by Harada and Kobayashi [28] to establish Conjecture 4.1.1 for two-level integer multisets. They experimentally showed that their algorithm tends to almost always find fix-free codes, when they exist, for multisets containing at most 30 integers, with two or more distinct values.

In 2013, Aghajan and Khosravifard [2] calculated the fraction of cases covered by Yekhanin's result (2) in [76].

In 2015, Bodewig [13, 14] proved several special cases of Conjecture 4.1.1 for infinite multisets.

Today, there still remains an infinite number of unsolved cases of Conjecture 4.1.1.

4.2 Summary of the main result

Our main result covers an infinite number of new cases not previously known in the literature, and is summarized in Theorem 4.2.1.

Theorem 4.2.1. *Conjecture 4.1.1 is true whenever the multiset contains at most three distinct integers.*

Proof. By Lemma 4.5.1, it suffices to prove the result when the Kraft sum is exactly $3/4$. If the multiset contains only one distinct integer λ_1, then any subset of A^{λ_1} of size $\mu_1 = 3 \cdot 2^{\lambda_1 - 2}$ will give the desired fix-free code. If the multiset contains exactly two distinct integers, then the result is known by [28] (see also our Theorem 4.3.1).

Suppose the multiset contains exactly three distinct integers, which, in increasing order, are λ_1, λ_2, λ_3, with nonzero multiplicities μ_1, μ_2, μ_3, respectively, and such that $\mu_1 2^{-\lambda_1} + \mu_2 2^{-\lambda_2} + \mu_3 2^{-\lambda_3} = 3/4$. Theorem 4.1.2 implies that Conjecture 4.1.1 holds when $\mu_1 2^{-\lambda_1} \geq 1/2$, so it suffices to assume $\mu_1 2^{-\lambda_1} \leq 1/2$, in which case the proof follows from our following three results:

- Theorem 4.6.2, i.e., when $\mu_1 2^{-\lambda_1} \leq \frac{1}{2}$ and $\mu_2 2^{-\lambda_2} \leq \frac{1}{4}$
- Theorem 4.7.2, i.e., when $\mu_1 2^{-\lambda_1} \leq \frac{1}{2}$ and $\frac{1}{4} \leq \mu_2 2^{-\lambda_2} \leq \frac{1}{2}\left(1 - \mu_1 2^{-\lambda_1}\right)$
- Theorem 4.8.1, i.e., when $\mu_1 2^{-\lambda_1} \leq \frac{1}{2}$ and $\frac{1}{2}\left(1 - \mu_1 2^{-\lambda_1}\right) \leq \mu_2 2^{-\lambda_2}$

These theorems are stated and proven in Sections 4.6, 4.7, 4.8, respectively. If $\lambda_1 = 1$, then $\mu_1 = 1$, so Theorem 4.1.2 applies. Thus, for each of Theorems 4.6.2, 4.7.2, and 4.8.1, it suffices to assume $\lambda_1 \geq 2$. Throughout the proofs of these three theorems, we will use the following transformed quantities:

$$\begin{aligned} n &= \lambda_1 - 1 \\ l &= \lambda_2 - \lambda_1 + 1 \\ k &= \lambda_3 - \lambda_1 + 1. \end{aligned} \tag{4.1}$$

■

The main idea used in proving Theorems 4.6.2, 4.7.2, and 4.8.1 is to build sets of codewords of the three desired lengths so that none of the shorter words is a prefix or suffix of any longer word.

The codewords of the shortest length λ_1 will be elements of the set $U_1 A^n$, where $U_1 = 0$. The set of codewords of the middle length λ_2 will be a union of at most three sets of the form $U_2 A^{l-2} U_3 A^n$, where U_2 and U_3 are two fixed bits. Since $\lambda_1 = n + 1$, the conditions $U_1 \neq U_2$ and $U_2 = U_3$ ensure any word from $U_2 A^{l-2} U_3 A^n$ will not have a length-λ_1 prefix or suffix from $U_1 A^n$. Additionally, even if $U_1 = U_2$ or $U_1 = U_3$, we will still be able to choose codewords of length λ_2 as long as these words avoid having prefixes or suffixes among the codewords from $U_1 A^n$.

Once the codewords of lengths λ_1 and λ_2 are constructed with the correct multiplicities μ_1 and μ_2, and with no offending prefixes or suffixes, we then carefully construct enough codewords of length λ_3 to make the total Kraft sum equal $3/4$, while avoiding prefixes and suffixes from codewords of lengths λ_1 and λ_2.

During this construction, the locations in words of length λ_3 of the fixed bits U_1, U_2, and U_3 (which are fixed in words of lengths λ_1 and λ_2) play an important role in our ability to avoid prefixes and suffixes of words of lengths λ_1 and λ_2. There are three possible "overlap cases" that are separately considered, depending on how much overlap there is between the prefix and suffix of length λ_2 in a codeword of length λ_3. The three cases are illustrated in Figure 4.1, and correspond to whether the value $\lambda_2 - \lambda_1$ is less than, equal to, or greater than, $\lambda_3 - \lambda_2$, An equivalent condition, using the terminology from (4.1), is whether the value $2l - k$ is less than, equal to, or greater than 1.

In Overlap Case 1, the fixed bits U_2 and U_3 of the length-λ_2 prefixes and suffixes do not coincide, and also the length-λ_2 prefixes do not overlap the length-λ_2 suffixes in their first l positions. In this case, length-λ_3 codewords are drawn from sets of the form $Z_1A^{l-2}Z_2A^{k-2l}Z_3A^{l-2}Z_4A^n$, for some subset of the 16 possible assignments of (Z_1, Z_2, Z_3, Z_4). In Overlap Case 2, the fixed bit U_3 in length-λ_2 prefixes of length-λ_3 codewords coincides with the fixed bit U_2 in length-λ_2 suffixes of such length-λ_3 codewords. In this case, length-λ_3 codewords are drawn from sets of the form $Z_1A^{l-2}Z_2A^{l-2}Z_3A^n$, for some subset of the 8 possible assignments of (Z_1, Z_2, Z_3). In Overlap Case 1, the fixed bits U_2 and U_3 of the length-λ_2 prefixes and suffixes do not coincide, but the length-λ_2 prefixes do overlap the length-λ_2 suffixes in their first l positions. It turns out that this latter property is a source of complication throughout the construction. In this case, length-λ_3 codewords are drawn from sets of the form $Z_1A^{k-l-1}Z_3A^{2l-k-2}Z_2A^{k-l-1}Z_4A^n$, for some subset of the 16 possible assignments of (Z_1, Z_2, Z_3, Z_4).

In all three cases, the codewords of length λ_3 are randomly selected from carefully designed patterns to avoid words of lengths λ_1 and λ_2 as prefixes and suffixes. The proof of our main result demonstrates that, on average, the random choices of codewords of lengths λ_1 and λ_2 leave enough remaining potential codewords of length λ_3 to satisfy the Kraft sum being $3/4$ without violating the prefix or suffix conditions. This bound on the average Kraft sum implies that there exists at least one particular choice of words of the desired lengths that forms a fix-free code and satisfies the requirements.

4.3 The $3/4$ Conjecture with two distinct lengths

In order to illustrate aspects of our random coding technique in a relatively simple example, we next prove Conjecture 4.1.1 when the multiset of positive integers is restricted to having only two distinct values. This result was originally given by Harada and Kobayashi [28] using a different, and considerably longer, proof.

Theorem 4.3.1. *Suppose a multiset of positive integers consists of μ_1 copies of λ_1 and μ_2 copies of λ_2, such that $\lambda_1 < \lambda_2$ and $\mu_1 2^{-\lambda_1} + \mu_2 2^{-\lambda_2} \leq 3/4$. Then there exists a fix-free code with μ_1 codewords of length λ_1 and μ_2 codewords of length λ_2.*

Proof. Let F be a randomly chosen set of μ_1 distinct words of length λ_1. Each word of length $\lambda_2 - \lambda_1$ is the prefix of a unique word of length λ_2 whose length-λ_1 prefix equals its length-λ_1 suffix. So the number of such words is $2^{\lambda_2-\lambda_1}$, and their Kraft sum is $2^{\lambda_2-\lambda_1} \cdot 2^{-\lambda_2} = 2^{-\lambda_1}$. The probability that any such word does not have its common length-λ_1 prefix/suffix in F is $\frac{2^{\lambda_1}-\mu_1}{2^{\lambda_1}}$. On the other hand, any word of length λ_2 whose length-λ_1 prefix and suffix differ does not have a prefix or suffix in F with probability $\frac{2^{\lambda_1}-\mu_1}{2^{\lambda_1}} \cdot \frac{2^{\lambda_1}-\mu_1-1}{2^{\lambda_1}-1}$ (using Lemma 4.5.6), and the Kraft sum of the set of such words is $(2^{\lambda_2} - 2^{\lambda_2-\lambda_1}) \cdot 2^{-\lambda_2} = 1 - 2^{-\lambda_1}$. Therefore, the expected Kraft

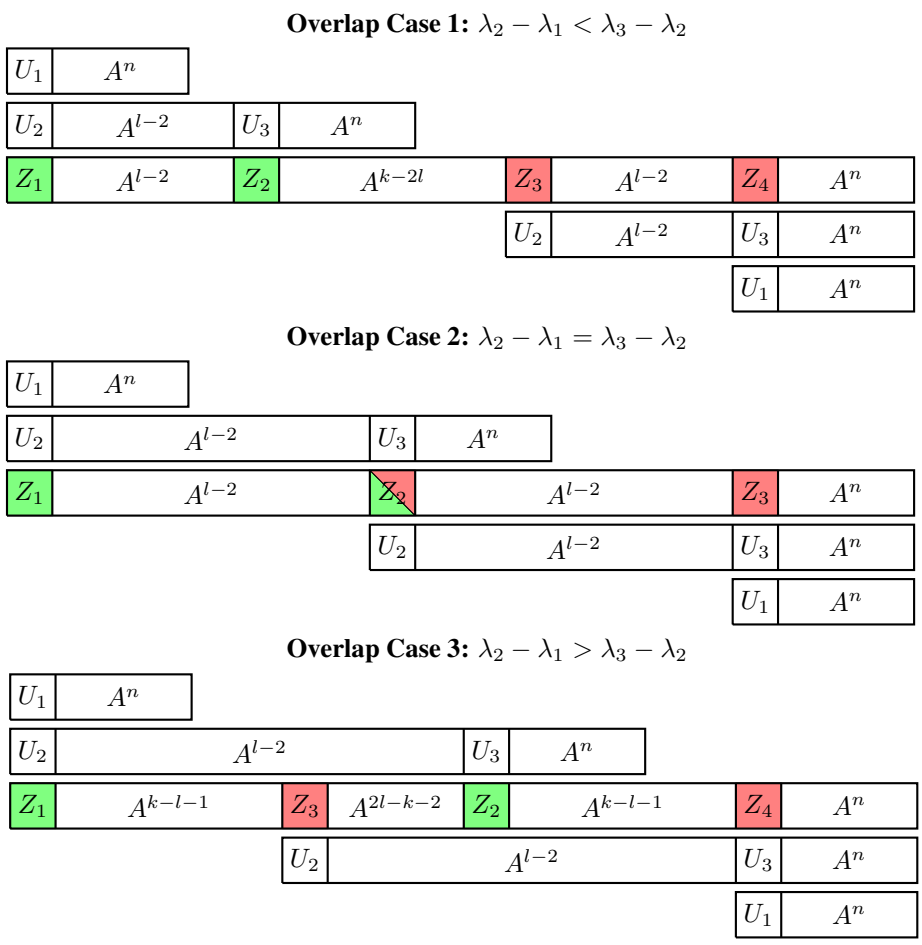


Figure 4.1. Three cases of code word overlap. The three word lengths illustrated are λ_1, λ_2, and λ_3. The bit positions U_1, U_2, U_3 correspond to certain fixed bits in patterns of length λ_1 and λ_2, and the bit positions Z_1, Z_2, Z_3, Z_4 represent fixed bits in patterns of length λ_3. These fixed bit positions are used to avoid prefixes and suffixes in order to create a fix-free code.

sum of the set of length-λ_2 words that have neither a prefix nor suffix in F is

$$2^{-\lambda_1} \cdot \frac{2^{\lambda_1} - \mu_1}{2^{\lambda_1}} + (1 - 2^{-\lambda_1}) \cdot \frac{(2^{\lambda_1} - \mu_1)(2^{\lambda_1} - \mu_1 - 1)}{2^{\lambda_1}(2^{\lambda_1} - 1)}$$
$$= \frac{3}{4} - \mu_1 2^{-\lambda_1} + \left(\frac{1}{2} - \mu_1 2^{-\lambda_1}\right)^2 \geq \frac{3}{4} - \mu_1 2^{-\lambda_1} \geq \mu_2 2^{-\lambda_2}.$$

Thus, there exists at least one particular choice of F such that there are at least μ_2 words of length λ_2 that have neither a prefix nor suffix in F, i.e., there are then enough available words of length λ_2 to create the claimed fix-free code. ■

4.4 Overview of the proof of the $3/4$ Conjecture with three distinct lengths

We give here a preview and high-level description of the proof of the $3/4$ Conjecture with three distinct lengths (i.e., the proof of Theorem 4.2.1). In Sections 4.6–4.8, detailed proofs are given, and some useful lemmas are given in Section 4.5 and proven in the appendix.

The random coding method illustrated in the proof of Theorem 4.3.1 for two distinct lengths plays an important role in the case of three distinct lengths, but significant complications arise when trying to avoid prefixes and suffixes in the words of longest length. To avoid such prefixes and suffixes in our constructions, we assign fixed values to certain bit locations in the chosen words of all three lengths, which in turn does make the analysis based on random coding more difficult. Also, the method in the proof of Theorem 4.3.1 of counting each length-λ_2 word whose length-λ_1 prefix is also a suffix does not work when there are bits with fixed values, as in the proofs with three distinct lengths, so we instead develop a more widely applicable result that is proven in our Lemma 4.5.3. As a result, the proofs provided in subsequent sections are substantially longer and more complex than that of Theorem 4.3.1.

Explicitly constructing the needed numbers of strings of lengths λ_1, λ_2, and λ_3 appears to be a difficult task, so we chose an alternative approach that randomly chooses such strings according to certain rules that maintain the prefix/suffix conditions. The construction process chooses the correct number of strings of lengths λ_1 and λ_2 and then we show that on average there remains enough strings of length λ_3 to complete the process.

The proof of Theorem 4.2.1 is broken into three main cases, depending on the values of the Kraft sum components $\mu_1 2^{-\lambda_1}$ and $\mu_2 2^{-\lambda_2}$. The three cases are:

(1) $\mu_1 2^{-\lambda_1} \leq \frac{1}{2}$ and $\mu_2 2^{-\lambda_2} \leq \frac{1}{4}$

(2) $\mu_1 2^{-\lambda_1} \leq \frac{1}{2}$ and $\frac{1}{4} \leq \mu_2 2^{-\lambda_2} \leq \frac{1}{2}\left(1 - \mu_1 2^{-\lambda_1}\right)$

(3) $\mu_1 2^{-\lambda_1} \leq \frac{1}{2}$ and $\frac{1}{2}\left(1 - \mu_1 2^{-\lambda_1}\right) \leq \mu_2 2^{-\lambda_2}$.

The third main case is broken into the following four subcases:

(a) $\lambda_2 \geq 2\lambda_1$ (i.e., $n \leq l - 2$)

(b) $\lambda_2 < 2\lambda_1$ (i.e., $n > l - 2$) and $\frac{1}{4} \leq \mu_1 2^{-\lambda_1} \leq \frac{1}{2}$

(c) $\lambda_2 < 2\lambda_1$ (i.e., $n > l - 2$) and $\mu_1 2^{-\lambda_1} < \frac{1}{4}$ and $\frac{1}{4} \leq \mu_2 2^{-\lambda_2} \leq \frac{1}{2}$

(d) $\lambda_2 < 2\lambda_1$ (i.e., $n > l - 2$) and $\mu_1 2^{-\lambda_1} < \frac{1}{4}$ and $\frac{1}{2} < \mu_2 2^{-\lambda_2}$.

Each of the main cases (1) and (2) and the subcases (3a)–(3d) are further divided into the three overlap cases illustrated in Figure 4.1, namely:

- $\lambda_2 - \lambda_1 < \lambda_3 - \lambda_2$ (i.e., $2l - k < 1$)
- $\lambda_2 - \lambda_1 = \lambda_3 - \lambda_2$ (i.e., $2l - k = 1$)
- $\lambda_2 - \lambda_1 > \lambda_3 - \lambda_2$ (i.e., $2l - k > 1$).

Within each overlap case of each main case or subcase, specific definitions are given of the three sets F_1, F_2, and F_3. These are the sets containing codewords of lengths λ_1, λ_2, and λ_3, respectively. Our construction chooses these three sets using various randomizations, and we show that in each case, on average, there are enough codewords to correctly populate the sets without violating the prefix or suffix conditions. Once this step is accomplished, we then conclude that there must be at least one (non-random) code with the correct sizes of F_1, F_2, and F_3, and without violating the prefix or suffix conditions.

- Constructing F_1:

  In all cases and subcases we define $F_1 = 0A^n - C$, where C is a set of size $2^n - \mu_1$ chosen randomly from certain subsets of $0A^n$. In other words, we construct F_1 by starting with all binary strings of length $n+1$ that start with 0, and then we delete in a random way enough of those strings to leave exactly μ_1 remaining.

  The motivation behind this definition of F_1 is that when we construct larger codewords of lengths λ_2 and λ_3, they can avoid having length-λ_1 codewords as prefixes by having a 1 in their leftmost position or having a word in C as a prefix, and they can avoid having length-λ_1 codewords as suffixes by having a 1 in position $n+1$ from the right or having a word in C as a suffix.

  In the main cases (1) and (2) and the subcase (3a), we choose C uniformly at random from among the 2^{n-1} length-λ_1 strings of $0A^n$. For these cases, the construction of F_1 is equivalent to simply choosing μ_1 elements at random without replacement from $0A^n$.

  In subcase (3b), we choose C uniformly at random from among the 2^{n-1} length-λ_1 strings of $0A^{l-2}1A^{n-l+1}$. In other words, in this case F_1 is constructed by randomly deleting enough strings from $0A^n$ containing a 1 in the lth position to leave exactly μ_1 strings remaining.

  In subcases (3c) and (3d), since $\mu_1 2^{-\lambda_1} < \frac{1}{4}$ the value of μ_1 is smaller than in case (3b), so the random set C must be made larger than in (3b). So we choose C to have all 2^{n-1} strings in $0A^{l-2}1A^{n-l+1}$ together with $2^{n-1} - \mu_1$ strings chosen uniformly at random from $0A^{l-2}0A^{n-l+1}$. In other words, in these cases F_1 is constructed by deleting all strings from $0A^n$ that contain a 1 in the lth position and also randomly deleting enough strings from $0A^n$ that contain a 0 in the lth position to leave exactly μ_1 strings remaining.

- Constructing F_2:

  The construction of F_2 requires that F_2 has μ_2 strings, each of length λ_2, and that none of these strings contains a prefix or suffix in F_1.

  Notice that each word in $1A^{l-2}1A^n$ avoids both prefixes and suffixes from F_1. There are $2^{n+l-2} = \frac{1}{4}2^{\lambda_2}$ such words available for F_2, and each is of length $n + l = \lambda_2$. For main case (1), this number of codewords is sufficient since $\mu_2 \leq \frac{1}{4}2^{\lambda_2}$, but for main cases (2) and (3) more codewords are needed of length λ_2 since $\mu_2 > \frac{1}{4}2^{\lambda_2}$ in those cases. In these two cases, one way to increase the number of codewords of length λ_2 is to include in F_2 some words from $0A^{l-2}1A^n$ or $1A^{l-2}0A^n$, and then require such words to have a prefix or suffix, respectively, from C, in order to avoid prefixes or suffixes from F_1.

  When constructing F_2, we make use of a new set D which is chosen uniformly at random from one of the four sets: $1A^{l-2}1A^n$, $0A^{l-2}1A^n$, $1A^{l-2}C$, or CA^{l-1}. In all cases except (3d), the words in set D are avoided when constructing F_2, which allows words of length λ_3 to have prefixes or suffixes from D in the construction of F_3. In contrast, in case (3d) we add words from D when constructing F_2, and then avoid such words in constructing F_3.

  Table 4.1 shows for each main case, subcase, and overlap case which of the four sets D is chosen from, how many words D contains, and the exact definition of F_2. The precise usage of these quantities will become apparent in the detailed proofs given in Sections 4.6–4.8. The involvement of D for constructing F_2 in each case allows sufficient codewords of length λ_2 while avoiding prefixes and suffixes from F_1.

- Constructing F_3:

  Our general strategy for constructing the set F_3 is to form a union of subsets of A^{k+n}, where each subset obeys certain constraints (according to which overlap case is being considered) that prevent prefixes or suffixes from F_1 or F_2. Each subset in such a union is an intersection of two specially constructed sets $Y_{p,q}$ and $W_{r,s}$ over various binary values of p, q, r, s, specified by an index set $\mathcal{I}$. The intersection of $Y_{p,q}$ and $W_{r,s}$ produces a pattern that falls into one of three specific forms, as seen in the third column of Table 4.2. The patterns are designed based on the locations of the bits Z_1, Z_2, Z_3, Z_4 in Figure 4.1. By controlling the values of these four bits we prevent the strings in F_3 from having prefixes and suffixes in F_1 and F_2. The sets $Y_{p,q}$ regulate prefixes and the sets $W_{r,s}$ regulate suffixes.

  Table 4.2 summarizes the construction of F_3 and the pattern used for each overlap case:

  These constructions of the random set F_3 are used for all cases, except for main case (3d), where a slightly different construction is used.

Table 4.1. The sets F_2 and D for all overlap cases and subcases used in the proof of Theorem 4.3.1

Case : Overlap	F_2	$\lvert D\rvert$	D is from
(1)	$1A^{l-2}1A^n - D$	$2^{n+l-2} - \mu_2$	$1A^{l-2}1A^n$
(2): 1,3	$(1A^{l-2}1A^n - D) \cup 1A^{l-2}C$	$\frac{1}{2}(1-\mu_1 2^{-\lambda_1})2^{\lambda_2} - \mu_2$	$1A^{l-2}1A^n$
(2): 2	$1A^{l-2}1A^n \cup (1A^{l-2}C - D)$	$\frac{1}{2}(1-\mu_1 2^{-\lambda_1})2^{\lambda_2} - \mu_2$	$1A^{l-2}C$
(3a): 1	$(1A^{l-2}1A^n - D) \cup 1A^{l-2}C \cup CA^{l-2-n}1A^n$	$(\frac{3}{4}-\mu_1 2^{-\lambda_1})2^{\lambda_2} - \mu_2$	$1A^{l-2}1A^n$
(3a): 2,3	$1A^{l-2}1A^n \cup (1A^{l-2}C - D) \cup CA^{l-2-n}1A^n$	$(\frac{3}{4}-\mu_1 2^{-\lambda_1})2^{\lambda_2} - \mu_2$	$1A^{l-2}C$
(3b): 1,3	$(1A^{l-2}1A^n - D) \cup CA^{l-1}$	$(\frac{3}{4}-\mu_1 2^{-\lambda_1})2^{\lambda_2} - \mu_2$	$1A^{l-2}1A^n$
(3b): 2	$1A^{l-2}1A^n \cup (CA^{l-1} - D)$	$(\frac{3}{4}-\mu_1 2^{-\lambda_1})2^{\lambda_2} - \mu_2$	CA^{l-1}
(3c): 1,3	$(1A^{l-2}1A^n - D) \cup 0A^{l-2}1A^n$	$2^{\lambda_2-1} - \mu_2$	$1A^{l-2}1A^n$
(3c): 2	$1A^{l-2}1A^n \cup (0A^{l-2}1A^n - D)$	$2^{\lambda_2-1} - \mu_2$	$0A^{l-2}1A^n$
(3d)	$1A^{l-2}1A^n \cup 0A^{l-2}1A^n \cup D$	$\mu_2 - 2^{\lambda_2-1}$	$1A^{l-2}C$

Table 4.2. The set F_3 and corresponding pattern for all overlap cases used in the proof of Theorem 4.3.1

Overlap Case	F_3	Pattern
(1)	$\bigcup_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} (Y_{Z_1,Z_2} \cap W_{Z_3,Z_4})$	$Z_1A^{l-2}Z_2A^{k-2l}Z_3A^{l-2}Z_4A^n$
(2)	$\bigcup_{(Z_1,Z_2,Z_3)\in\mathcal{I}} (Y_{Z_1,Z_2} \cap W_{Z_2,Z_3})$	$Z_1A^{l-2}Z_2A^{l-2}Z_3A^n$
(3)	$\bigcup_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} (Y_{Z_1,Z_3} \cap W_{Z_2,Z_4})$	$Z_1A^{k-l-1}Z_2A^{2l-k-2}Z_3A^{k-l-1}Z_4A^n$

The proofs in Sections 4.6–4.8 calculate the average size of F_3 and show that it is at least μ_3. This ensures that there is at least one (deterministic) instance of the random sets C and D that guarantees a (deteministic) instance of the set F_3 with at least μ_3 elements, and this instance of F_3 can be pruned back to have exactly μ_3 elements.

4.5 Lemmas about Kraft sums

This section gives many technical lemmas used to prove the main theorems in the sections to follow. Most of the proofs of the lemmas in this section can be found in the appendix.

For any set $S \subseteq A^*$, the *indicator function* $1_S : A^* \to \{0, 1\}$ is defined by

$$1_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{else.} \end{cases}$$

Lemma 4.5.1. *If a multiset of positive integers has Kraft sum less than $3/4$, then the multiplicity of its largest value can be increased to make the Kraft sum equal to $3/4$.*

Some basic facts about Kraft sums will be used throughout this paper. As some examples: (i) if S and T are disjoint codes, then $\mathcal{K}(S \cup T) = \mathcal{K}(S) + \mathcal{K}(T)$; (ii) if S and T are codes and at least one of them is fixed length, then $\mathcal{K}(ST) = \mathcal{K}(S)\,\mathcal{K}(T)$; (iii) $\mathcal{K}(A^n) = 1$ for all $n \geq 1$; and (iv) $\mathcal{K}(0) = \mathcal{K}(1) = 1/2$. Two consequences of these facts are given in the following lemma.

Lemma 4.5.2.

(i) If $p_1, \ldots, p_n$ are nonnegative integers and $u_1, \ldots, u_n \in A^*$, then

$$\mathcal{K}(u_1 A^{p_1} u_2 A^{p_2} \ldots u_n A^{p_n}) = 2^{-(|u_1| + \cdots + |u_n|)}.$$

(ii) If S is a code and T is a fixed-length random code, then $E[\mathcal{K}(ST)] = \mathcal{K}(S)\,E[\mathcal{K}(T)]$.

Let $m \geq l \geq 1$ be integers, and let $U \in \{0, 1, A\}^m$. Define $R_l(U) \subseteq A^m$ to be the set of words $w \in U$ such that the l-bit prefix of w equals the l-bit suffix of w.

The following lemma is used in many of the proofs of our other lemmas.

Lemma 4.5.3. *Let $U = U_1U_2 \cdots U_m \in \{0,1,A\}^m$ and let $l \leq m$ be a positive integer. Then the number of words in U whose length-l prefix and suffix are the same is*

$$|R_l(U)| = \prod_{p=1}^{m-l} \left| \bigcap_{\substack{1 \leq i \leq m \\ i \equiv p \bmod (m-l)}} U_i \right|.$$

Three examples illustrating the usage of Lemma 4.5.3 are given next.

- Let $U = 0A^20A^31$ and $l = 5$. So $m-l = 3$. Then $U_1 \cap U_4 \cap U_7 = 0 \cap 0 \cap A = 0$, $U_2 \cap U_5 \cap U_8 = A \cap A \cap 1 = 1$, and $U_3 \cap U_6 = A \cap A = A$. So $|R_l(U)| = |0| \cdot |1| \cdot |A| = 1 \cdot 1 \cdot 2 = 2$. The two words in $R_l(U)$ are 01001001 and 01101101.

- Let $U = 0A^20A^21A$ and $l = 5$. So $m - l = 3$. Then $U_1 \cap U_4 \cap U_7 = 0 \cap 0 \cap 1 = \varnothing$, $U_2 \cap U_5 \cap U_8 = A \cap A \cap A = A$, and $U_3 \cap U_6 = A \cap A = A$. So $|R_l(U)| = |\varnothing| \cdot |A| \cdot |A| = 0 \cdot 2 \cdot 2 = 0$.

- Let $U = 1A^21A1A^21$ and $l = 6$. So $m - l = 3$. Then $U_1 \cap U_4 \cap U_7 = 1 \cap 1 \cap A = 1$, $U_2 \cap U_5 \cap U_8 = A \cap A \cap A = A$, and $U_3 \cap U_6 \cap U_9 = A \cap 1 \cap 1 = 1$. So $|R_l(U)| = |1| \cdot |A| \cdot |1| = 1 \cdot 2 \cdot 1 = 2$. The two words in $R_l(U)$ are 101101101 and 111111111.

A *fixed point* in a pattern $X \in \{0,1,A\}^*$ is a position in the pattern's string whose value is not equal to A.

We will say that a randomly generated set of words of a given length is *of a fixed size* if the set is chosen according to some probability distribution on all sets of the same cardinality that contain words of the given length.

Lemma 4.5.4. *Let m be a positive integer. For each $i = 1, 2$ let $X_i \in \{0,1,A\}^{m_i}$, with $m_i \leq m$, and let Y_i be a set of a fixed size drawn uniformly at random and without replacement from X_i, where the words of Y_1 and Y_2 are drawn independentally of each other. Let $W_1 = A^aY_1A^b \cap U_1$ and $W_2 = A^cY_2A^d \cap U_2$ for some patterns $U_1 \subseteq A^aX_1A^b$ and $U_2 \subseteq A^cX_2A^d$, where $a + b = m - m_1$ and $c + d = m - m_2$. Let p denote the number of positions where U_1 and U_2 both have a fixed point, and assume that the values of U_1 and U_2 agree at each such position. Then*

$$E[\mathcal{K}(W_1 \cap W_2)] = 2^p \cdot \prod_{i=1}^{2} \mathcal{K}(U_i) \frac{\mathcal{K}(Y_i)}{\mathcal{K}(X_i)}.$$

Note that in the above lemma, the cardinality of each random set Y_i is fixed and its elements are all of length m_i, so the Kraft sum of Y_i is deterministic.

Corollary 4.5.5. *Let Y be a set of a fixed size chosen uniformly at random and without replacement from a pattern $X \in \{0,1,A\}^{n+1}$. Let $U \in \{0,1,A\}^{n+k}$. If $U \subseteq A^a X A^b$, then*

$$\mathcal{K}\left(A^a Y A^b \cap U\right) = \mathcal{K}(U) \cdot \frac{\mathcal{K}(Y)}{\mathcal{K}(X)}.$$

Lemma 4.5.6. *Let X be a set of size at least 2 and let C be a set of a fixed size chosen uniformly at random from X. For any particular element of X, the probability that the element lies in C is $|C|/|X|$. For any two particular distinct elements of X, the probability that both lie in C is $\frac{|C|(|C|-1)}{|X|(|X|-1)}$.*

Lemma 4.5.7. *Let $n \geq 1$ and $p \geq 0$ be integers, let $b \in A$, and let C be a set of a fixed size chosen uniformly at random from bA^n. Then for any $U \in \{0,1,A\}^p$,*

$$E[\mathcal{K}\left(CA^{p+1} \cap bUbA^n \cap A^{p+1}C\right)] = \mathcal{K}(U)\,\mathcal{K}(C)^2.$$

The next lemma calculates the expected Kraft sum of the set all $(k+n)$-bit words that have both a prefix and suffix in a randomly chosen set of words of the form $1A^{l-2}1A^n$, where $2 \leq l < k$.

Lemma 4.5.8. *Let $n, l, k \geq 0$ be integers, with $2 \leq l < k$, and let D be a subset of a fixed size chosen uniformly at random from $1A^{l-2}1A^n$. Then*

$$E[\mathcal{K}\left(DA^{k-l} \cap A^{k-l}D\right)] = \begin{cases} \mathcal{K}(D)^2 & \text{if } 2l-k<1 \\ 2\mathcal{K}(D)^2 & \text{if } 2l-k=1 \\ \mathcal{K}(D)^2 & \text{if } 2l-k>1 \text{ and } (k-l) \nmid (2l-k-1) \\ \mathcal{K}(D)^2 + \frac{\mathcal{K}(D)(\frac{1}{4}-\mathcal{K}(D))}{2^{n+l-2}-1} & \text{if } 2l-k>1 \text{ and } (k-l) \mid (2l-k-1). \end{cases}$$

Corollary 4.5.9. *Let $n, l, k \geq 1$ be integers, with $2 \leq l < k$ and $n > l-2$. Let C_0 be a subset of a fixed size*

chosen uniformly at random from $0A^{l-2}0A^{n-(l-1)}$. *Then*

$$E[\mathcal{K}\left(C_0A^{k-l} \cap A^{k-l}C_0\right)]$$
$$= \begin{cases} \mathcal{K}\left(C_0\right)^2 & \text{if } 2l-k<1 \\ 2\mathcal{K}\left(C_0\right)^2 & \text{if } 2l-k=1 \\ \mathcal{K}\left(C_0\right)^2 & \text{if } 2l-k>1 \text{ and } (k-l) \nmid (2l-k-1) \\ \mathcal{K}\left(C_0\right)^2 + \frac{\mathcal{K}(C_0)(\frac{1}{4}-\mathcal{K}(C_0))}{2^{n-1}-1} & \text{if } 2l-k>1 \text{ and } (k-l) \mid (2l-k-1). \end{cases}$$

Proof. This corollary follows from Lemma 4.5.8 by changing 1s to 0s. ■

Lemma 4.5.10. *Let* $C \subseteq A^{n+1}$ *be a random set of a fixed size. Let* $g(C) \subseteq A^{n+k}$ *be a set that is some function of* C. *If* D *is a set of a fixed size chosen uniformly at random from* $1A^{l-2}C$, *then*

$$E[\mathcal{K}\left(DA^{k-l} \cap g(C)\right)] = \frac{\mathcal{K}\left(D\right)}{\mathcal{K}\left(C\right)/2} \cdot E[\mathcal{K}\left(1A^{l-2}CA^{k-l} \cap g(C)\right)]$$
$$E[\mathcal{K}\left(g(C) \cap A^{k-l}D\right)] = \frac{\mathcal{K}\left(D\right)}{\mathcal{K}\left(C\right)/2} \cdot E[\mathcal{K}\left(g(C) \cap A^{k-l}1A^{l-2}C\right)],$$

and if D *is a set of a fixed size chosen uniformly at random from* CA^{l-1}, *then*

$$E[\mathcal{K}\left(DA^{k-l} \cap g(C)\right)] = \frac{\mathcal{K}\left(D\right)}{\mathcal{K}\left(C\right)} \cdot E[\mathcal{K}\left(CA^{k-1} \cap g(C)\right)],$$

where $\mathcal{K}(D)/\mathcal{K}(C) = 0$ *whenever* $\mathcal{K}(C) = \mathcal{K}(D) = 0$.

In Theorem 4.8.1(c) and Theorem 4.8.1(d) in Section 4.8, the set C is not chosen uniformly at random from a fixed pattern, but instead $C = C_1 \cup C_0$, where $C_1 = 0A^{l-2}1A^{n-(l-1)}$ and C_0 is chosen uniformly at random from $0A^{l-2}0A^{n-(l-1)}$. The following lemma calculates $E[\mathcal{K}\left(DA^{k-l} \cap A^{k-1}C\right)]$ in this situation.

Lemma 4.5.11. *Let* $n, l, k \geq 1$ *be integers, with* $2 \leq l < k$ *and* $n > l-2$. *Let* $C = C_1 \cup C_0$ *be a set of a fixed size where* $C_1 = 0A^{l-2}1A^{n-(l-1)}$ *and* C_0 *is chosen uniformly at random from* $0A^{l-2}0A^{n-(l-1)}$. *Let* D *be a set*

of a fixed size chosen uniformly at random from $1A^{l-2}C$*. Then*

$$E[\mathcal{K}\left(DA^{k-l} \cap A^{k-1}C\right)]$$
$$= \begin{cases} \mathcal{K}(C)\,\mathcal{K}(D) & \text{if } 2l-k<1 \\ 2(\mathcal{K}(C)-\frac{1}{4})\mathcal{K}(D) & \text{if } 2l-k=1 \\ \mathcal{K}(C)\,\mathcal{K}(D) & \text{if } 2l-k>1 \text{ and } (k-l)\nmid(2l-k-1) \\ \mathcal{K}(C)\,\mathcal{K}(D)+\frac{\mathcal{K}(D)(\mathcal{K}(C)-\frac{1}{4})(\frac{1}{2}-\mathcal{K}(C))}{\mathcal{K}(C)(2^{n-1}-1)} & \text{if } 2l-k>1 \text{ and } (k-l)\mid(2l-k-1). \end{cases}$$

Lemma 4.5.12. *Let* $n \geq 1$ *and* $a, b \geq 0$ *be integers and let* C *be a subset of a fixed size chosen uniformly at random from* $0A^n$*. Then*

$$E[\mathcal{K}\left(1A^aCA^{a+b+2} \cap 1A^a0A^b0A^a1A^n \cap A^{a+b+2}CA^{a+1}\right)]$$
$$= \begin{cases} \frac{\mathcal{K}(C)^2}{4} - \frac{\mathcal{K}(C)(\frac{1}{2}-\mathcal{K}(C))}{4(2^n-1)} & \text{if } n>a \text{ and } (b+1)\mid(a+1) \\ \frac{\mathcal{K}(C)^2}{4} & \text{else.} \end{cases}$$

Lemma 4.5.13. *Let* $n \geq 1$ *and* $a, b \geq 0$ *be integers and let* C *be a subset of a fixed size chosen uniformly at random from* $0A^n$*. Then*

$$E[\mathcal{K}\left(CA^{2a+b+3} \cap 0A^a0A^b0A^a1A^n \cap 0A^aCA^{a+b+2}\right)]$$
$$= \begin{cases} \frac{\mathcal{K}(C)^2}{4} & \text{if } n \leq b \\ \frac{\mathcal{K}(C)^2}{4} & \text{if } b<n\leq a+b+1 \text{ and } (a+1)\nmid(b+1) \\ \frac{\mathcal{K}(C)^2}{4} + \frac{\mathcal{K}(C)(\frac{1}{2}-\mathcal{K}(C))}{4(2^n-1)} & \text{if } b<n\leq a+b+1 \text{ and } (a+1)\mid(b+1) \\ \frac{\mathcal{K}(C)^2}{4} - \frac{\mathcal{K}(C)(\frac{1}{2}-\mathcal{K}(C))}{4(2^n-1)} & \text{if } n>a+b+1. \end{cases}$$

Lemma 4.5.14. *Let* $n \geq 1$ *and* $a, b \geq 0$ *be integers and let* $l = a+b+3$*. Let* C *be a subset of a fixed size of at least* 1 *chosen uniformly at random from* $0A^n$*, and let* D *be a set of a fixed size chosen uniformly at random from*

$1A^{l-2}C$. *Then*

$$E[\mathcal{K}\left(DA^{a+1} \cap 1A^a1A^b0A^a0A^n \cap A^{a+1}D\right)]$$
$$= \begin{cases} \mathcal{K}\left(D\right)^2 & \text{if } (a+1) \nmid (b+1) \\ \mathcal{K}\left(D\right)^2 - \frac{\mathcal{K}(D)}{|C|2^{l-2}-1} \cdot (\frac{\mathcal{K}(C)}{2} - \mathcal{K}\left(D\right)) & \text{if } (a+1) \mid (b+1). \end{cases}$$

Lemma 4.5.15. *Let $n \geq 1$ and $l \geq 3$ and*

$$f(x,y) = \left(\frac{1}{4} - y\right)^2 - \frac{\frac{1}{2} - x}{2(2^n - 1)}\left(\frac{x}{2} - y\right) - \frac{y}{x2^{n+l-1} - 1}\left(\frac{x}{2} - y\right) - \frac{1}{4(2^n - 1)}x\left(\frac{1}{2} - x\right).$$

Then $f(x,y) \geq 0$ for all $x \in [\frac{1}{2^{n+1}}, \frac{1}{2} - \frac{1}{2^{n+1}}] \cup \{\frac{1}{2}\}$ and $y \in [0, \frac{x}{2} - \frac{1}{2^{n+l}}] \cup \{\frac{x}{2}\}$.

4.6 Main result, part 1: $\mu_1 2^{-\lambda_1} \leq \frac{1}{2}$ and $\mu_2 2^{-\lambda_2} \leq \frac{1}{4}$

We are given three positive integers in increasing order, λ_1, λ_2, λ_3, and corresponding nonzero multiplicities μ_1, μ_2, μ_3, such that the Kraft sum $\mu_1 2^{-\lambda_1} + \mu_2 2^{-\lambda_2} + \mu_3 2^{-\lambda_3}$ equals $3/4$. We are also given that the Kraft sums of the words of lengths λ_1 and λ_2 are upper bounded by $1/2$ and $1/4$, respectively. Our objective is to demonstrate that a fix-free code exists with the corresponding multiset of integers as codeword lengths.

We first construct length-λ_1 codewords by randomly removing a subset C of $0A^n$, whose size is chosen to leave exactly μ_1 codewords remaining. Then, length-λ_2 codewords are chosen from the words in $1A^{l-2}1A^n$, since none of them can have a prefix or suffix from the length-λ_1 words already chosen. Specifically, these words are chosen by randomly removing a subset D of $1A^{l-2}1A^n$, whose size is picked to leave exactly μ_2 words remaining after removal. The largest possible Kraft sum of the length-λ_2 words that can be achieved in this manner occurs when no words are removed, i.e., when $|D| = 0$. In this case, the expected Kraft sum of the length-λ_2 words is $\mathcal{K}\left(1A^{l-2}1A^n\right) = 1/4$, by Lemma 4.5.2, which explains the upper bound on $\mu_2 2^{-\lambda_2}$ used in Theorem 4.6.2.

Finally, length-λ_3 words are constructed to avoid prefixes and suffixes in the randomly constructed sets of words of lengths λ_1 and λ_2.

It appears to be a somewhat difficult task to describe which codewords of lengths λ_1 and λ_2 to use in order to ensure the availability of the needed length-λ_3 codewords, while preserving the fix-free condition and the $3/4$ Kraft sum upper bound.

We use a probabilistic approach and remove the correct number of codewords of lengths λ_1 and λ_2 by random selection. In other words, we remove $2^{\lambda_1 - 1} - \mu_1$ of the original length-λ_1 codewords, uniformly at random

from among the 2^{λ_1-1} original length-λ_1 codewords, and then we remove $2^{\lambda_2-2}-\mu_2$ of the original length-λ_2 codewords uniformly at random from among the 2^{λ_2-2} original length-λ_2 codewords. We prove that, on average, there are at least μ_3 codewords of length λ_3 that do not have any prefix or suffix from the resulting μ_1 codewords of length λ_1 and μ_2 codewords of length λ_2. So, there must exist at least one actual collection of μ_1 codewords of length λ_1 and μ_2 codewords of length λ_2 that result in at least μ_3 codewords of length λ_3 that have neither prefix nor suffix in the collection. This existential technique is somewhat analogous to that used in Shannon's proof of the channel coding theorem [60].

Throughout the proof of the main results, we shall use certain repeated terminology. The quantities λ_1, λ_2, and λ_3 will represent, in increasing order, the three distinct codeword lengths for the desired fix-free code. Throughout, we will use the quantities n, l, and k as defined in (4.1).

The proof of Theorem 4.6.2 is broken into three "overlap cases", namely when: (1) $2l-k<1$; (2) $2l-k=1$; and (3) $2l-k>1$. These cases correspond, respectively, to when a length-λ_2 prefix and a length-λ_2 suffix of a length-λ_3 word: (1) overlap in at most n positions; (2) overlap in exactly $n+1$ positions; and (3) overlap in at least $n+2$ positions. The same three cases are also used to prove Theorems 4.7.2 and 4.8.1. These three cases are illustrated in Figure 4.1.

The proof of Theorem 4.6.2 uses the following lemma, whose proof can be found in the appendix.

Lemma 4.6.1. *Let $n,l,k\geq 1$ be integers such that $2\leq l<k$. Let C be a set of a fixed size chosen uniformly at random from $0A^n$. Let D be a set of a fixed size chosen uniformly at random from $1A^{l-2}1A^n$. For any $b_1,b_2\in A$, if $2l-k\neq 1$, then*

(i) $\mathcal{K}\left(1A^{l-2}0A^{n+k-l}\cap A^{k-l}0A^{l-2}1A^n\right)=1/16$

(ii) $E[\mathcal{K}\left(CA^{k-1}\cap 0A^{l-2}b_1A^{n+k-l}\cap A^{k-l}0A^{l-2}1A^n\right)]=\mathcal{K}(C)/8$

(iii) $E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l}\cap A^{k-l}b_1A^{l-2}C\right)]=\mathcal{K}(C)/8$

(iv) $E[\mathcal{K}\left(CA^{k-1}\cap 0A^{l-2}b_1A^{n+k-l}\cap A^{k-l}b_2A^{l-2}C\right)]=\mathcal{K}(C)^2/4$

(v) $E[\mathcal{K}\left(DA^{k-l}\cap A^{k-l}0A^{l-2}1A^n\right)]=\mathcal{K}(D)/4$

(vi) $E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l}\cap A^{k-l}D\right)]=\mathcal{K}(D)/4$

(vii) $E[\mathcal{K}\left(CA^{k-1}\cap 0A^{l-2}b_1A^{n+k-l}\cap A^{k-l}D\right)]=\mathcal{K}(C)\mathcal{K}(D)/2$

(viii) $E[\mathcal{K}\left(DA^{k-l}\cap A^{k-l}b_1A^{l-2}C\right)]=\mathcal{K}(C)\mathcal{K}(D)/2$

(ix) $$E[\mathcal{K}\left(DA^{k-l}\cap A^{k-l}D\right)]=\begin{cases}\mathcal{K}(D)^2 & \text{if } 2l-k<1\\ \mathcal{K}(D)^2 & \text{if } 2l-k>1 \text{ and } (k-l)\nmid(2l-k-1)\\ \mathcal{K}(D)^2+\frac{\mathcal{K}(D)(\frac{1}{4}-\mathcal{K}(D))}{2^{n+l-2}-1} & \text{if } 2l-k>1 \text{ and } (k-l)\mid(2l-k-1)\end{cases}$$

and if $2l - k = 1$*, then*

(x) $\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-l}0A^{l-2}1A^n\right) = 1/8$

(xi) $E[\mathcal{K}\left(CA^{k-1} \cap 0A^{l-2}0A^{n+k-l} \cap A^{k-l}0A^{l-2}1A^n\right)] = \mathcal{K}(C)/4$

(xii) $E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-l}0A^{l-2}C\right)] = \mathcal{K}(C)/4$

(xiii) $E[\mathcal{K}\left(CA^{k-1} \cap 0A^{l-2}b_1A^{n+k-l} \cap A^{k-l}b_1A^{l-2}C\right)] = \mathcal{K}(C)^2/2$

(xiv) $E[\mathcal{K}\left(CA^{k-1} \cap 0A^{l-2}1A^{n+k-l} \cap A^{k-l}D\right)] = \mathcal{K}(C)\,\mathcal{K}(D)$

(xv) $E[\mathcal{K}\left(DA^{k-l} \cap A^{k-l}1A^{l-2}C\right)] = \mathcal{K}(C)\,\mathcal{K}(D)$

(xvi) $E[\mathcal{K}\left(DA^{k-l} \cap A^{k-l}D\right)] = 2\mathcal{K}(D)^2$.

Theorem 4.6.2. *Suppose a multiset of positive integers consists of* μ_1 *copies of* λ_1*,* μ_2 *copies of* λ_2*, and* μ_3 *copies of* λ_3*, such that* $2 \leq \lambda_1 < \lambda_2 < \lambda_3$*. Then there exists a fix-free code with* μ_1 *codewords of length* λ_1*,* μ_2 *codewords of length* λ_2*, and* μ_3 *codewords of length* λ_3*, whenever the following conditions hold:*

$$\begin{aligned} \mu_1 2^{-\lambda_1} &\leq \frac{1}{2} \\ \mu_2 2^{-\lambda_2} &\leq \frac{1}{4} \\ \mu_1 2^{-\lambda_1} + \mu_2 2^{-\lambda_2} + \mu_3 2^{-\lambda_3} &= \frac{3}{4}. \end{aligned}$$

Proof. Let C be a set of size $2^n - \mu_1$ chosen uniformly at random from the 2^n length-λ_1 elements of $0A^n$, and let D be a set of size $2^{n+l-2} - \mu_2$ chosen uniformly at random from the 2^{n+l-2} length-λ_2 elements of $1A^{l-2}1A^n$. Define the following (random) sets:

$$\begin{aligned} F_1 &= 0A^n - C \\ F_2 &= 1A^{l-2}1A^n - D. \end{aligned}$$

Then F_1 contains μ_1 words, each of length λ_1, and F_2 contains μ_2 words, each of length λ_2. By Lemma 4.5.2, we have $\mathcal{K}(F_1) = \mathcal{K}(0A^n) - \mathcal{K}(C) = \frac{1}{2} - \mathcal{K}(C) \leq \frac{1}{2}$ and $\mathcal{K}(F_2) = \mathcal{K}\left(1A^{l-2}1A^n\right) - \mathcal{K}(D) = \frac{1}{4} - \mathcal{K}(D) \leq \frac{1}{4}$.

In each of three cases, we will construct a third random set of words, F_3. The random set

$$F = F_1 \cup F_2 \cup F_3$$

on average forms the desired fix-free code. The union of non-random instances of F_1, F_2, and F_3 will then yield the asserted fix-free code. Let

$$Y_{i,j} = \begin{cases} CA^{k-1} \cap 0A^{l-2}jA^{n+k-l} & \text{if } i{=}0 \\ 1A^{l-2}0A^{n+k-l} & \text{if } i{=}1,\ j{=}0 \\ DA^{k-l} & \text{if } i{=}j{=}1 \end{cases} \qquad W_{i,j} = \begin{cases} A^{k-l}iA^{l-2}C & \text{if } j{=}0 \\ A^{k-l}0A^{l-2}1A^{n} & \text{if } i{=}0,\ j{=}1 \\ A^{k-l}D & \text{if } i{=}j{=}1. \end{cases}$$

- Overlap Case 1: $2l - k < 1$.

  In this case, the set F_3 is built as a union of 16 disjoint subsets of A^{k+n}. The basic building block of each such subset is a pattern of the form $Z_1A^{l-2}Z_2A^{k-2l}Z_3A^{l-2}Z_4A^n$, where Z_1, Z_2, Z_3, Z_4 are fixed bits that ensure the 16 subsets are disjoint and are chosen to avoid prefixes or suffixes from F_1 or F_2. When these four bits do not prevent such prefixes or suffixes, the sets $Y_{i,j}$ and $W_{i,j}$ are constructed to remove offending prefixes or suffixes. These constructions can require certain subsets to have prefixes or suffixes in C and/or D. The terms in each intersection below satisfy $Y_{Z_1,Z_2} \cap W_{Z_3,Z_4} \subseteq Z_1A^{l-2}Z_2A^{k-2l}Z_3A^{l-2}Z_4A^n$.

  Let $\mathcal{I} = A^4$ and define the set F_3, containing words of length λ_3, by:

$$F_3 = \bigcup_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} (Y_{Z_1,Z_2} \cap W_{Z_3,Z_4})$$

  Each of the 16 sets in the union comprising F_3 consists of words of length λ_3, and these sets, except when $(Z_1, Z_2, Z_3, Z_4) = (1, 0, 0, 1)$, are random, since they involve the random sets C or D. Thus, the Kraft sums of all but one term in the union are random variables.

  When $Z_1 = 0$ (respectively, $Z_4 = 0$), the words in the sets of the union are designed to contain prefixes (respectively, suffixes) in C in order to avoid prefixes (respectively, suffixes) in F_1, and when $Z_1 = Z_2 = 1$ (respectively, $Z_3 = Z_4 = 1$), the words in the sets of the union are designed to contain prefixes (respectively, suffixes) in D in order to avoid prefixes (respectively, suffixes) in F_2.

  It is easy to verify that none of the words of F_2 have prefixes or suffixes in F_1, none of the words of F_3 have prefixes or suffixes in F_1 or F_2, and that every two of the sets in the union forming F_3 are disjoint.

Next, we lower bound the expected Kraft sum of F_3:

$$\begin{aligned}
E[\mathcal{K}(F_3)] &= \sum_{(Z_1,Z_2,Z_3,Z_4)\in A^4} E[\mathcal{K}(Y_{Z_1,Z_2} \cap W_{Z_3,Z_4})] \\
&= \frac{\mathcal{K}(C)^2}{4} + \frac{\mathcal{K}(C)}{8} + \frac{\mathcal{K}(C)^2}{4} + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} + \frac{\mathcal{K}(C)^2}{4} + \frac{\mathcal{K}(C)}{8} \\
&\quad + \frac{\mathcal{K}(C)^2}{4} + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} + \frac{\mathcal{K}(C)}{8} + \frac{1}{16} + \frac{\mathcal{K}(C)}{8} + \frac{\mathcal{K}(D)}{4} \\
&\quad + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} + \frac{\mathcal{K}(D)}{4} + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} + \mathcal{K}(D)^2 \qquad (4.2) \\
&= \left(\mathcal{K}(C) + \mathcal{K}(D) - \frac{1}{4}\right)^2 + \mathcal{K}(C) + \mathcal{K}(D) \\
&\geq \mathcal{K}(C) + \mathcal{K}(D) \\
&= \frac{1}{2} - \mu_1 2^{-\lambda_1} + \frac{1}{4} - \mu_2 2^{-\lambda_2} \qquad (4.3) \\
&= \frac{3}{4} - \mu_1 2^{-\lambda_1} - \mu_2 2^{-\lambda_2} \qquad (4.4)
\end{aligned}$$

where (4.2) follows from Lemma 4.6.1.

From (4.4), we can lower bound the expected size of the random set F_3 by

$$E[|F_3|] \geq 2^{\lambda_3}\left(\frac{3}{4} - \mu_1 2^{-\lambda_1} - \mu_2 2^{-\lambda_2}\right) = \mu_3.$$

There must exist at least one instance of the randomly constructed set F_3 that satisfies the same lower bound satisfied by the average size of F_3. Such an instance of the random set F_3 corresponds to some particular choices of the random sets C and D. Let $\widehat{F}_1$ and $\widehat{F}_2$ denote the resulting (non-random) instances of the random sets F_1 and F_2, respectively. Let $\widehat{F}_3$ denote the resulting (non-random) instance of F_3, but only after throwing away enough codewords to make the size of $\widehat{F}_3$ exactly equal to the lower bound on $E[|F_3|]$. That is,

$$|\widehat{F}_3| = \mu_3.$$

The code $\widehat{F}_1 \cup \widehat{F}_2 \cup \widehat{F}_3$ is fix-free, has Kraft sum equal to $3/4$, and has μ_1, μ_2, μ_3 codewords of sizes λ_1, λ_2, λ_3, respectively.

- Overlap Case 2: $2l - k = 1$.

  In this case, the set F_3 is built in a similar manner as in Overlap Case 1, although here it will be a union of only 8 disjoint subsets of A^{k+n}, using patterns of the form $Z_1 A^{l-2} Z_2 A^{l-2} Z_3 A^n$. The terms in each intersection below satisfy $Y_{Z_1,Z_2} \cap W_{Z_2,Z_3} \subseteq Z_1 A^{l-2} Z_2 A^{l-2} Z_3 A^n$.

  Let $\mathcal{I} = A^3$ and define the set F_3 containing words of length λ_3, and lower bound its expected Kraft sum as follows:

$$\begin{aligned}
F_3 &= \bigcup_{(Z_1,Z_2,Z_3)\in\mathcal{I}} (Y_{Z_1,Z_2} \cap W_{Z_2,Z_3}) \\
E[\mathcal{K}(F_3)] &= \sum_{(Z_1,Z_2,Z_3)\in\mathcal{I}} E[\mathcal{K}(Y_{Z_1,Z_2} \cap W_{Z_2,Z_3})] \\
&= \frac{\mathcal{K}(C)^2}{2} + \frac{\mathcal{K}(C)}{4} + \frac{\mathcal{K}(C)^2}{2} + \mathcal{K}(C)\,\mathcal{K}(D) \\
&\quad + \frac{\mathcal{K}(C)}{4} + \frac{1}{8} + \mathcal{K}(C)\,\mathcal{K}(D) + 2\mathcal{K}(D)^2 \qquad (4.5) \\
&= \left(\mathcal{K}(C) + \mathcal{K}(D) - \frac{1}{4}\right)^2 + \left(\mathcal{K}(D) - \frac{1}{4}\right)^2 + \mathcal{K}(C) + \mathcal{K}(D) \\
&\geq \mathcal{K}(C) + \mathcal{K}(D)
\end{aligned}$$

  where (4.5) follows from Lemma 4.6.1. Overlap Case 2 is then finished by applying the same reasoning as used from (4.3) to the end of Overlap Case 1.

- Overlap Case 3: $2l - k > 1$.

  This case is nearly identical to Overlap Case 1, but uses the following definition of F_3:

$$F_3 = \bigcup_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} (Y_{Z_1,Z_3} \cap W_{Z_2,Z_4})$$

  where $Y_{Z_1,Z_3} \cap W_{Z_2,Z_4} \subseteq Z_1 A^{k-l-1} Z_2 A^{2l-k-2} Z_3 A^{k-l-1} Z_4 A^n$. The only other difference is that the equal sign in (4.2) changes to $\geq$, since in Lemma 4.6.1(ix) when $2l - k > 1$ we have $E[\mathcal{K}\left(DA^{k-l} \cap A^{k-l}D\right)] \geq \mathcal{K}(D)^2$.

■

4.7 Main result, part 2: $\mu_1 2^{-\lambda_1} \leq \frac{1}{2}$ and $\frac{1}{4} \leq \mu_2 2^{-\lambda_2} \leq \frac{1}{2}\left(1-\mu_1 2^{-\lambda_1}\right)$

In Theorem 4.6.2, sets of words of lengths λ_1 and λ_2 were initially constructed, and then some words were independently and uniformly removed from each set in order to bring their sizes down to μ_1 and μ_2, respectively. Then a set of of length-λ_3 words was constructed that avoided prefixes and suffixes from the random sets of lengths λ_1 and λ_2. The constraint in Theorem 4.6.2 that $\mu_2 2^{-\lambda_2} \leq \frac{1}{4}$ allowed us to construct the set F_2 of length-λ_2 words entirely based on words from $1A^{l-2}1A^n$ (whose Kraft sum is $1/4$).

The construction for Theorem 4.7.2 is slightly different however, since this theorem requires $\mu_2 2^{-\lambda_2} \geq \frac{1}{4}$, but there are not enough words in $1A^{l-2}1A^n$ to get a Kraft sum larger than $1/4$ for the length-λ_2 words.

To solve this issue, in Theorem 4.7.2 we start with a larger set of length-λ_2 words, namely $1A^{l-2}1A^n \cup 1A^{l-2}C$, where C is a random set of words removed from $0A^n$ to leave exactly μ_1 of such words of length $n+1 = \lambda_1$ remaining (just like in Theorem 4.6.2). Then, to construct a set of μ_2 length-λ_2 codewords, we remove a randomly selected subset D from $1A^{l-2}1A^n$ in Overlap Cases 1 and 3, and from $1A^{l-2}C$ in Overlap Case 2, where the cardinality of D ensures that there will be a total of μ_2 codewords of length λ_2 left after removal. In this manner, no length-λ_1 codewords can be prefixes or suffixes of any length-λ_2 codewords, since any word in the set $1A^{l-2}1A^n$ has a fixed bit of 1 where a length-λ_1 prefix or suffix from $0A^n$ would have a 0, and any word in the set $1A^{l-2}C$ has a fixed bit of 1 where a length-λ_1 prefix would have a 0, and has a length-λ_1 suffix from C, which, by construction, cannot be one of the μ_1 words of length λ_1 chosen for the fix-free code.

The largest Kraft sum of the length-λ_2 words that we can get with this technique is when we do not remove any codewords of length λ_2, i.e., when $|D| = 0$, in which case the expected Kraft sum of the length-λ_2 words, in all three overlap cases, is

$$\begin{aligned}
\mathcal{K}\left(1A^{l-2}1A^n \cup 1A^{l-2}C\right) &= \mathcal{K}\left(1A^{l-2}1A^n\right) + \mathcal{K}\left(1A^{l-2}C\right) \\
&= \mathcal{K}\left(1A^{l-2}1A^n\right) + \mathcal{K}\left(1A^{l-2}\right)\mathcal{K}\left(C\right) && (4.6)\\
&= \frac{1}{4} + \frac{1}{2}(2^n - \mu_1)2^{-(n+1)} && (4.7)\\
&= \frac{1}{2}\left(1 - \mu_1 2^{-\lambda_1}\right) && (4.8)
\end{aligned}$$

where (4.6) follows from Lemma 4.5.2 and Corollary 4.5.5; (4.7) follows from Lemma 4.5.2 and the fact that $\mathcal{K}\left(C\right)$ equals the constant $(2^n - \mu_1)2^{-(n+1)}$; and (4.8) explains the upper bound on $\mu_2 2^{-\lambda_2}$ imposed in Theorem 4.7.2.

The proof of Theorem 4.7.2 uses the following lemma, whose proof can be found in the appendix.

Lemma 4.7.1. *Let $n, l, k \geq 1$ be integers such that $2 \leq l < k$. Let C be a set of a fixed size chosen uniformly at random from $0A^n$. Let D_1 be a set of a fixed size chosen uniformly at random from $1A^{l-2}1A^n$, and let D_2 be a set*

of a fixed size chosen uniformly at random from $1A^{l-2}C$. *For any* $b_1, b_2 \in A$, *if* $2l-k \neq 1$, *then*

(i) $E[\mathcal{K}\left(1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}0A^{l-2}1A^n\right)] = \left(\frac{1}{2} - \mathcal{K}(C)\right)/8$

(ii) $E[\mathcal{K}\left(1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}0A^{l-2}C\right)] = \mathcal{K}(C)\left(\frac{1}{2} - \mathcal{K}(C)\right)/4$

(iii) $E[\mathcal{K}\left(1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}D_1\right)] = \mathcal{K}(D_1)\left(\frac{1}{2} - \mathcal{K}(C)\right)/2$

and if $2l-k=1$, *then*

(iv) $E[\mathcal{K}\left(1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}0A^{l-2}1A^n\right)] = \left(\frac{1}{2} - \mathcal{K}(C)\right)/4$

(v) $E[\mathcal{K}\left(CA^{k-1} \cap 0A^{l-2}1A^{n+k-l} \cap A^{k-l}D_2\right)] = \mathcal{K}(C)\,\mathcal{K}(D_2)$

(vi) $E[\mathcal{K}\left(1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}0A^{l-2}C\right)] = \mathcal{K}(C)\left(\frac{1}{2} - \mathcal{K}(C)\right)/2$

(vii) $E[\mathcal{K}\left(D_2A^{k-l} \cap A^{k-l}0A^{l-2}1A^n\right)] = \mathcal{K}(D_2)/2$

(viii) $E[\mathcal{K}\left(D_2A^{k-l} \cap A^{k-l}0A^{l-2}C\right)] = \mathcal{K}(C)\,\mathcal{K}(D_2)$

Theorem 4.7.2. *Suppose a multiset of positive integers consists of* μ_1 *copies of* λ_1, μ_2 *copies of* λ_2, *and* μ_3 *copies of* λ_3, *such that* $2 \leq \lambda_1 < \lambda_2 < \lambda_3$, *Then there exists a fix-free code with* μ_1 *codewords of length* λ_1, μ_2 *codewords of length* λ_2, *and* μ_3 *codewords of length* λ_3, *whenever the following conditions hold:*

$$\begin{aligned} \mu_1 2^{-\lambda_1} &\leq \frac{1}{2} \\ \frac{1}{4} \leq \mu_2 2^{-\lambda_2} &\leq \frac{1}{2}\left(1 - \mu_1 2^{-\lambda_1}\right) \\ \mu_1 2^{-\lambda_1} + \mu_2 2^{-\lambda_2} + \mu_3 2^{-\lambda_3} &= \frac{3}{4}. \end{aligned}$$

Proof. As in Part 1, let C be a set of size $2^n - \mu_1$ chosen uniformly at random from among the 2^n length-λ_1 elements of $0A^n$ and define the following (random) set:

$$F_1 = 0A^n - C.$$

- For Overlap Cases 1 and 3 below:

  Let D be a set of size $\frac{1}{2}\left(1 - \mu_1 2^{-\lambda_1}\right)2^{\lambda_2} - \mu_2$ chosen uniformly at random from among the 2^{n+l-2} length-λ_2 elements of $1A^{l-2}1A^n$, and let

$$F_2 = (1A^{l-2}1A^n - D) \cup (1A^{l-2}C).$$

- For Overlap Case 2 below:

  Let D be a set of size $\frac{1}{2}\left(1-\mu_1 2^{-\lambda_1}\right)2^{\lambda_2}-\mu_2$ chosen uniformly at random from among the $2^{l-2}\cdot|C|$ length-λ_2 elements of $1A^{l-2}C$, and let

$$F_2 = 1A^{l-2}1A^n \cup (1A^{l-2}C - D).$$

Then $\mathcal{K}(D) = \frac{1}{2}\left(1-\mu_1 2^{-\lambda_1}\right) - \mu_2 2^{-\lambda_2}$, so $0 \le \mathcal{K}(D) \le \frac{1}{2}\left(1-\mu_1 2^{-\lambda_1}\right) - \frac{1}{4} = \frac{1}{4} - \mu_1 2^{-\lambda_1-1} \le \frac{1}{4}$. This means there are enough words from which to choose D, since $\mathcal{K}\left(1A^{l-2}1A^n\right) = \frac{1}{4}$, and $\mathcal{K}\left(1A^{l-2}C\right) = \frac{1}{2}\mathcal{K}(C) = \frac{1}{2}(\frac{1}{2} - \mu_1 2^{-\lambda_1}) = \frac{1}{4} - \mu_1 2^{-\lambda_1-1}$, by Lemma 4.5.2.

Note that the words in F_2 all start with 1, and have a 0 in position l only if they have a suffix in C. These conditions guarantee that no word in F_1 is a prefix or suffix of a word in F_2. In all 3 cases,

$$\begin{aligned}
|F_1| &= |0A^n| - |C| = 2^n - (2^n - \mu_1) = \mu_1 \\
|F_2| &= |1A^{l-2}1A^n| + |1A^{l-2}C| - |D| \\
&= 2^{n+l-2} + 2^{l-2}(2^n - \mu_1) - \frac{1}{2}\left(1-\mu_1 2^{-\lambda_1}\right)2^{\lambda_2} + \mu_2 \\
&= \mu_2
\end{aligned}$$

so the set F_1 contains μ_1 words, each of length λ_1, and F_2 contains μ_2 words, each of length λ_2. The set F_1 can be viewed as being chosen uniformly at random among all subsets of $0A^n$ of size μ_1.

In each case, we will also construct a third random set F_3, consisting of μ_3 words of length λ_3. The random set

$$F = F_1 \cup F_2 \cup F_3$$

on average forms the desired fix-free code. The union of non-random instances of F_1, F_2, and F_3 will then yield the asserted fix-free code.

- Overlap Case 1: $2l - k < 1$.

Let

$$Y_{i,j} = \begin{cases} CA^{k-1} \cap 0A^{l-2}jA^{n+k-l} & \text{if } i{=}0 \\ 1A^{l-2}(0A^n{-}C)A^{k-l} & \text{if } i{=}1,\ j{=}0 \\ DA^{k-l} & \text{if } i{=}j{=}1 \end{cases} \quad W_{i,j} = \begin{cases} A^{k-l}0A^{l-2}C & \text{if } i{=}j{=}0 \\ A^{k-l}0A^{l-2}1A^n & \text{if } i{=}0,\ j{=}1 \\ A^{k-l}D & \text{if } i{=}j{=}1. \end{cases} \tag{4.9}$$

Let $\mathcal{I} = A^4 - A^2 10$ and define the set F_3 by:

$$F_3 = \bigcup_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} (Y_{Z_1,Z_2} \cap W_{Z_3,Z_4})$$

where $Y_{Z_1,Z_2} \cap W_{Z_3,Z_4} \subseteq Z_1A^{l-2}Z_2A^{k-2l}Z_3A^{l-2}Z_4A^n$.

In contrast to Overlap Cases 1 and 3 of Part 1, here F_3 is comprised of only 12 of the 16 possible sets obtained from the pattern $Z_1A^{l-2}Z_2A^{k-2l}Z_3A^{l-2}Z_4A^n$, namely by excluding $(Z_3, Z_4) = (1, 0)$ from the union. One can verify that no words in F_1 or F_2 can be either prefixes or suffixes of any words in F_3.

The expected Kraft sum of F_3 is then lower bounded as follows:

$$\begin{aligned} E[\mathcal{K}(F_3)] &= \sum_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} E[\mathcal{K}(Y_{Z_1,Z_2} \cap W_{Z_3,Z_4})] \\ &= \frac{\mathcal{K}(C)^2}{4} + \frac{\mathcal{K}(C)}{8} + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} + \frac{\mathcal{K}(C)^2}{4} \\ &\quad + \frac{\mathcal{K}(C)}{8} + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} + \frac{\mathcal{K}(C)(\frac{1}{2} - \mathcal{K}(C))}{4} + \frac{\frac{1}{2} - \mathcal{K}(C)}{8} \\ &\quad + \frac{\mathcal{K}(D)(\frac{1}{2} - \mathcal{K}(C))}{2} + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} + \frac{\mathcal{K}(D)}{4} + \mathcal{K}(D)^2 \qquad (4.10) \\ &= \left(\frac{\mathcal{K}(C)}{2} + \mathcal{K}(D) - \frac{1}{4}\right)^2 + \frac{\mathcal{K}(C)}{2} + \mathcal{K}(D) \\ &\geq \frac{\mathcal{K}(C)}{2} + \mathcal{K}(D) \qquad (4.11) \\ &= \frac{1}{4} - \frac{1}{2}\mu_1 2^{-\lambda_1} + \frac{1}{2} - \frac{1}{2}\mu_1 2^{-\lambda_1} - \mu_2 2^{-\lambda_2} \qquad (4.12) \\ &= \frac{3}{4} - \mu_1 2^{-\lambda_1} - \mu_2 2^{-\lambda_2} \end{aligned}$$

where (4.10) follows from Lemma 4.7.1 when $(Z_1, Z_2) = (1, 0)$ and otherwise follows from Lemma 4.6.1; and (4.12) follows from the quantities $|C|$ and $|D|$ defined at the beginning of the proof of this theorem.

The current case is then finished by applying the same reasoning used following (4.4) to the end of Part 1, Overlap Case 1.

- Overlap Case 2: $2l - k = 1$.

Let

$$Y_{i,j} = \begin{cases} CA^{k-1} \cap 0A^{l-2}jA^{n+k-l} & \text{if } i{=}0 \\ (D \cup 1A^{l-2}(0A^n{-}C))A^{k-l} & \text{if } i{=}1,\ j{=}0 \end{cases} \qquad W_{i,j} = \begin{cases} A^{k-l}0A^{l-2}C & \text{if } i{=}j{=}0 \\ A^{k-l}0A^{l-2}1A^n & \text{if } i{=}0,\ j{=}1 \\ A^{k-l}D & \text{if } i{=}1,\ j{=}0. \end{cases}$$

Let $\mathcal{I} = A^3 - (11A \cup A11)$ and define the set F_3 by:

$$F_3 = \bigcup_{(Z_1,Z_2,Z_3)\in\mathcal{I}} (Y_{Z_1,Z_2} \cap W_{Z_2,Z_3})$$

where $Y_{Z_1,Z_2} \cap W_{Z_2,Z_3} \subseteq Z_1A^{l-2}Z_2A^{l-2}Z_3A^n$. In this case, F_3 is comprised of only 5 of the 8 possible sets obtained from the pattern $Z_1A^{l-2}Z_2A^{l-2}Z_3A^n$, namely by excluding (Z_1, Z_2, Z_3) from being $(1, 1, 0)$, $(0, 1, 1)$, or $(1, 1, 1)$ in the union.

The expected Kraft sum of F_3 can be lower bounded as follows:

$$\begin{aligned} E[\mathcal{K}(F_3)] &= \sum_{(Z_1,Z_2,Z_3)\in\mathcal{I}} E[\mathcal{K}(Y_{Z_1,Z_2} \cap W_{Z_2,Z_3})] \\ &= \frac{\mathcal{K}(C)^2}{2} + \frac{\mathcal{K}(C)}{4} + \mathcal{K}(C)\mathcal{K}(D) \\ &\quad + \left(\frac{\mathcal{K}(C)(\frac{1}{2} - \mathcal{K}(C))}{2} + \mathcal{K}(C)\mathcal{K}(D)\right) \\ &\quad + \left(\frac{\frac{1}{2} - \mathcal{K}(C)}{4} + \frac{\mathcal{K}(D)}{2}\right) \qquad (4.13) \\ &= \frac{(1 - 2\mathcal{K}(C))(1 - 4\mathcal{K}(D))}{8} + \mathcal{K}(C)\mathcal{K}(D) + \frac{\mathcal{K}(C)}{2} + \mathcal{K}(D) \\ &\geq \frac{\mathcal{K}(C)}{2} + \mathcal{K}(D) \qquad (4.14) \end{aligned}$$

where (4.13) follows from Lemma 4.6.1 when $Z_1 = Z_2 = 0$ and otherwise follows from Lemma 4.7.1; and (4.14) follows since $\mathcal{K}(D) \leq 1/4$ and $\mathcal{K}(C) \leq 1/2$.

Overlap Case 2 is then finished by applying the same reasoning as used from (4.11) to the end of Overlap

Case 1.

- Overlap Case 3: $2l - k > 1$.

We use the same sets $Y_{i,j}$ and $W_{i,j}$ as defined in (4.9) for Overlap Case 1. Let $\mathcal{I} = A^4 - A1A0$ and define the set F_3 by:

$$F_3 = \bigcup_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} (Y_{Z_1,Z_3} \cap W_{Z_2,Z_4})$$

where $Y_{Z_1,Z_3} \cap W_{Z_2,Z_4} \subseteq Z_1A^{k-l-1}Z_2A^{2l-k-2}Z_3A^{k-l-1}Z_4A^n$. Here F_3 is comprised of 12 of the 16 possible sets obtained by excluding $(Z_2, Z_4) = (1, 0)$.

The expected Kraft sum of F_3 is then lower bounded as follows:

$$\begin{aligned} E[\mathcal{K}(F_3)] &= \sum_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} E[\mathcal{K}(Y_{Z_1,Z_3} \cap W_{Z_2,Z_4})] \\ &\geq \left(\frac{\mathcal{K}(C)}{2} + \mathcal{K}(D) - \frac{1}{4}\right)^2 + \frac{\mathcal{K}(C)}{2} + \mathcal{K}(D) \end{aligned} \tag{4.15}$$

where the $\geq$ in (4.15) follows from Lemma 4.6.1(ix), and the remainder of the proof is the same as for Overlap Case 1 starting at (4.11).

■

4.8 Main result, part 3: $\mu_1 2^{-\lambda_1} \leq \frac{1}{2}$ and $\frac{1}{2}\left(1 - \mu_1 2^{-\lambda_1}\right) \leq \mu_2 2^{-\lambda_2}$

The methods for constructing codes in this section are similar in spirit to the methods from the past two sections, but contain some significant changes as well. As before, sets of words of lengths λ_1 and λ_2 are initially constructed, but in this section sometimes we will then remove words at random from these sets, and sometimes we will add words at random to these sets. Specifically, in the proofs of Theorem 4.8.1, parts (a), (b), and (c), some words are removed, and in part (d) some words are added, but in all cases the resulting words of lengths λ_1 and λ_2 have cardinalities μ_1 and μ_2, respectively. Then, just as in previous sections, we will show that there are enough words of length λ_3 available to produce the desired fix-free code.

There are additional complications in this section that result in more cases to consider than in the previous sections. Before, we removed words of length λ_2 from $1A^{l-2}1A^n$ or $1A^{l-2}C$ only, but in this section we will need to remove words of length λ_2 from $CA^{l-1} \cap 0A^{l-2}1A^n$ as well. However, if $n > l - 2$ and C happens to be

a subset of $0A^{l-2}0A^{n-l+1}$, then $CA^{l-1} \cap 0A^{l-2}1A^n = \varnothing$, leaving us no length-$\lambda_2$ words to remove from this set. To remedy this, we split the proof into separate lemmas, where we first consider the case when $n \leq l-2$ (in which we proceed in a similar fashion as the previous sections), and then consider when $n > l-2$. This latter case requires us to take more care in choosing C, and so we break this case into three separate lemmas.

Additionally, it turns out that as n grows, other complications can arise depending on the values of the lengths λ_2 and λ_3. As can be seen in Lemma 4.8.2, particularly in cases (ix)–(xii), the expected Kraft sums of certain sets may depend on specific divisibility conditions involving the codeword lengths. These conditions are a result of the ways in which randomly chosen codewords may overlap each other as factors in codewords of a larger length. Fortunately, these complications are present in Overlap Case 3 of Theorem 4.8.1(a) only, and we use Lemma 4.5.15 to prove our desired result even in this case.

Theorem 4.8.1. *Suppose a multiset of positive integers consists of μ_1 copies of λ_1, μ_2 copies of λ_2, and μ_3 copies of λ_3, such that $2 \leq \lambda_1 < \lambda_2 < \lambda_3$. Then there exists a fix-free code with μ_1 codewords of length λ_1, μ_2 codewords of length λ_2, and μ_3 codewords of length λ_3, whenever the following conditions hold:*

$$\begin{aligned} \mu_1 2^{-\lambda_1} &\leq \frac{1}{2} \\ \frac{1}{2}\left(1 - \mu_1 2^{-\lambda_1}\right) &\leq \mu_2 2^{-\lambda_2} \\ \mu_1 2^{-\lambda_1} + \mu_2 2^{-\lambda_2} + \mu_3 2^{-\lambda_3} &= \frac{3}{4}. \end{aligned}$$

Theorem 4.8.1 follows immediately from the following four cases, which depend on the values of λ_1, λ_2, μ_1, and μ_2:

(a) $\lambda_2 \geq 2\lambda_1$

(b) $\lambda_2 < 2\lambda_1$ and $\frac{1}{4} \leq \mu_1 2^{-\lambda_1} \leq \frac{1}{2}$

(c) $\lambda_2 < 2\lambda_1$ and $\mu_1 2^{-\lambda_1} < \frac{1}{4}$ and $\frac{1}{4} \leq \mu_2 2^{-\lambda_2} \leq \frac{1}{2}$

(d) $\lambda_2 < 2\lambda_1$ and $\mu_1 2^{-\lambda_1} < \frac{1}{4}$ and $\frac{1}{2} < \mu_2 2^{-\lambda_2}$.

4.8.1 Proof of Theorem 4.8.1(a)

The proof of Theorem 4.8.1(a) uses the following lemma, whose proof can be found in the appendix.

Lemma 4.8.2. *Let $n, l, k \geq 1$ be integers such that $2 \leq l < k$ and $n \leq l-2$. Let C be a set of a fixed size chosen uniformly at random from $0A^n$. Let D_1 be a set of a fixed size chosen uniformly at random from $1A^{l-2}1A^n$, and let D_2 be a set of a fixed size chosen uniformly at random from $1A^{l-2}C$. For any $b \in A$, if $2l - k < 1$, then*

(i) $E[\mathcal{K}\left(CA^{l-2-n}0A^{n+k-l} \cap A^{k-l}(0A^n - C)A^{l-2-n}1A^n\right)] = \mathcal{K}(C)\,(\frac{1}{2} - \mathcal{K}(C))/4$

(ii) $E[\mathcal{K}\left(1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}(0A^n - C)A^{l-2-n}1A^n\right)] = (\frac{1}{2} - \mathcal{K}(C))^2/4$

(iii) $E[\mathcal{K}\left(D_1A^{k-l} \cap A^{k-l}(0A^n - C)A^{l-2-n}1A^n\right)] = \mathcal{K}(D_1)\,(\frac{1}{2} - \mathcal{K}(C))/2.$

If $2l - k = 1$, *then*

(iv) $E[\mathcal{K}\left(CA^{l-2-n}0A^{n+k-l} \cap A^{k-l}(0A^n - C)A^{l-2-n}1A^n\right)] = \mathcal{K}(C)\,(\frac{1}{2} - \mathcal{K}(C))/2$

(v) $E[\mathcal{K}\left(1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}(0A^n - C)A^{l-2-n}1A^n\right)] = (\frac{1}{2} - \mathcal{K}(C))/4.$

If $2l - k > 1$, *then*

(vi) $E[\mathcal{K}\left(CA^{l-2-n}0A^{n+k-l} \cap A^{k-l}D_2\right)] = \mathcal{K}(C)\,\mathcal{K}(D_2)\,/2$

(vii) $E[\mathcal{K}\left(1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}D_2\right)] = (\frac{1}{2} - \mathcal{K}(C))\mathcal{K}(D_2)\,/2$

(viii) $E[\mathcal{K}\left(D_2A^{k-l} \cap A^{k-l}0A^{l-2}C\right)] = \mathcal{K}(C)\,\mathcal{K}(D_2)\,/2$

(ix)

$$E[\mathcal{K}\left(CA^{l-2-n}0A^{n+k-l} \cap A^{k-l}(0A^n - C)A^{l-2-n}1A^n\right)]$$
$$= \frac{\mathcal{K}(C)\,(\frac{1}{2} - \mathcal{K}(C))}{4} - \begin{cases} \frac{\mathcal{K}(C)(\frac{1}{2} - \mathcal{K}(C))}{4(2^n - 1)} & \textit{if } n > 2l - k - 2 \textit{ and } (k-l) \mid (2l-k-1) \\ 0 & \textit{otherwise} \end{cases}$$

(x)

$$E[\mathcal{K}\left(1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}(0A^n - C)A^{l-2-n}1A^n\right)]$$
$$= \frac{(\frac{1}{2} - \mathcal{K}(C))^2}{4} - \begin{cases} \frac{\mathcal{K}(C)(\frac{1}{2} - \mathcal{K}(C))}{4(2^n - 1)} & \textit{if } n > k - l - 1 \textit{ and } (2l-k-1) \mid (k-l) \\ 0 & \textit{otherwise} \end{cases}$$

(xi)

$$E[\mathcal{K}\left(D_2A^{k-l} \cap A^{k-l}(0A^n - C)A^{l-2-n}1A^n\right)]$$
$$= \frac{(\frac{1}{2} - \mathcal{K}(C))\mathcal{K}(D_2)}{2} + \begin{cases} \frac{\mathcal{K}(D_2)(\frac{1}{2} - \mathcal{K}(C))}{2(2^n - 1)} & \textit{if } n > k - l - 1 \textit{ and } (2l-k-1) \mid (k-l) \\ 0 & \textit{otherwise} \end{cases}$$

(xii)

$$E[\mathcal{K}\left(D_2 A^{k-l} \cap A^{k-l} D_2\right)]$$
$$= \mathcal{K}(D_2)^2 - \begin{cases} \frac{\mathcal{K}(D_2)}{|C| \cdot 2^{l-2}-1}\left(\frac{\mathcal{K}(C)}{2} - \mathcal{K}(D_2)\right) & \text{if } (k-l) \mid (2l-k-1) \\ 0 & \text{otherwise.} \end{cases}$$

.

Proof of Theorem 4.8.1(a). Here we assume $\lambda_2 \geq 2\lambda_1$ (or equivalently $n \leq l-2$ by (4.1)).

Let C be a set of size $2^n - \mu_1$ chosen uniformly at random from among the 2^n length-λ_1 elements of $0A^n$. Note that $0 \leq \mu_1 \leq 2^n$ since $0 \leq \mu_1 2^{-\lambda_1} \leq \frac{1}{2}$. Also, $\mathcal{K}(C) = (2^n - \mu_1)\, 2^{-\lambda_1} = \frac{1}{2} - \mu_1 2^{-\lambda_1}$.

Define the following (random) set:

$$F_1 = 0A^n - C.$$

- For Overlap Case 1 below:

  Let D be a set of size $(\frac{3}{4} - \mu_1 2^{-\lambda_1})2^{\lambda_2} - \mu_2$ chosen uniformly at random from among the 2^{n+l-2} length-λ_2 elements of $1A^{l-2}1A^n$, and let

$$F_2 = (1A^{l-2}1A^n - D) \cup 1A^{l-2}C \cup CA^{l-2-n}1A^n.$$

- For Overlap Cases 2 and 3 below:

  Let D be a set of size $(\frac{3}{4} - \mu_1 2^{-\lambda_1})2^{\lambda_2} - \mu_2$ chosen uniformly at random from among the $2^{l-2} \cdot |C|$ length-λ_2 elements of $1A^{l-2}C$, and let

$$F_2 = 1A^{l-2}1A^n \cup (1A^{l-2}C - D) \cup CA^{l-2-n}1A^n.$$

Then

$$\begin{aligned}\mathcal{K}(D) &= |D| \cdot 2^{-\lambda_2} \\ &= \frac{3}{4} - \mu_1 2^{-\lambda_1} - \mu_2 2^{-\lambda_2} \\ &\leq \frac{3}{4} - \mu_1 2^{-\lambda_1} - \frac{1}{2} + \frac{\mu_1 2^{-\lambda_1}}{2} \\ &= \frac{1}{4} - \frac{\mu_1 2^{-\lambda_1}}{2}.\end{aligned} \tag{4.16}$$

This means there are enough words from which to choose D, since $\frac{1}{4} - \frac{\mu_1 2^{-\lambda_1}}{2} \leq \frac{1}{4} = \mathcal{K}\left(1A^{l-2}1A^n\right)$ and $\frac{1}{4} - \frac{\mu_1 2^{-\lambda_1}}{2} = \frac{\mathcal{K}(C)}{2} = \mathcal{K}\left(1A^{l-2}C\right)$.

Note that the $(n+1)$-bit prefixes and suffixes of words in F_2 either start with 1 or else lie in C, whereas all words in F_1 start with 0 and cannot lie in C. So no word in F_1 can be a prefix or a suffix of a word in F_2.

Also in all 3 cases,

$$\begin{aligned}|F_1| &= |0A^n| - |C| = 2^n - (2^n - \mu_1) = \mu_1 \\ |F_2| &= |1A^{l-2}1A^n| + |1A^{l-2}C| + |CA^{l-2-n}1A^n| - |D| \\ &= 2^{n+l-2} + 2^{l-2}(2^n - \mu_1) + 2^{l-2}(2^n - \mu_1) - \left(\frac{3}{4} - \mu_1 2^{-\lambda_1}\right) 2^{\lambda_2} + \mu_2 \\ &= \mu_2\end{aligned}$$

so the set F_1 contains μ_1 words, each of length λ_1, and F_2 contains μ_2 words, each of length λ_2.

In each case, we construct a third random set F_3 consisting of μ_3 words, each of length λ_3. The random set

$$F = F_1 \cup F_2 \cup F_3$$

on average forms the desired fix-free code. The union of non-random instances of F_1, F_2, and F_3 will then yield the asserted fix-free code.

- Overlap Case 1: $2l - k < 1$.

Let

$$Y_{i,j} = \begin{cases} CA^{l-2-n}0A^{n+k-l} & \text{if } i{=}j{=}0 \\ 1A^{l-2}(0A^n{-}C)A^{k-l} & \text{if } i{=}1,\ j{=}0 \\ DA^{k-l} & \text{if } i{=}j{=}1 \end{cases} \quad W_{i,j} = \begin{cases} A^{k-l}0A^{l-2}C & \text{if } i{=}j{=}0 \\ A^{k-l}(0A^n{-}C)A^{l-2-n}1A^n & \text{if } i{=}0,\ j{=}1 \\ A^{k-l}D & \text{if } i{=}j{=}1. \end{cases}$$

Let $\mathcal{I} = A^4 - (01A^2 \cup A^2 10)$ and define the set F_3 by:

$$F_3 = \bigcup_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} (Y_{Z_1,Z_2} \cap W_{Z_3,Z_4})$$

where $Y_{Z_1,Z_2} \cap W_{Z_3,Z_4} \subseteq Z_1A^{l-2}Z_2A^{k-2l}Z_3A^{l-2}Z_4A^n$. Here F_3 is comprised of only 9 of the 16 possible sets obtained from the pattern $Z_1A^{l-2}Z_2A^{k-2l}Z_3A^{l-2}Z_4A^n$, namely by excluding patterns with $(Z_1, Z_2) = (0, 1)$ or $(Z_3, Z_4) = (1, 0)$. One can verify that no words in F_1 or F_2 are either prefixes or suffixes of any words in F_3.

The expected Kraft sum of F_3 is then lower bounded as follows:

$$\begin{aligned} E[\mathcal{K}(F_3)] &= \sum_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} E[\mathcal{K}(Y_{Z_1,Z_2} \cap W_{Z_3,Z_4})] \\ &= \frac{\mathcal{K}(C)^2}{4} + \frac{\mathcal{K}(C)(\frac{1}{2} - \mathcal{K}(C))}{4} + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} + \frac{\mathcal{K}(C)(\frac{1}{2} - \mathcal{K}(C))}{4} \\ &\quad + \frac{(\frac{1}{2} - \mathcal{K}(C))^2}{4} + \frac{\mathcal{K}(D)(\frac{1}{2} - \mathcal{K}(C))}{2} + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} \\ &\quad + \frac{\mathcal{K}(D)(\frac{1}{2} - \mathcal{K}(C))}{2} + \mathcal{K}(D)^2 \qquad (4.17) \\ &= \left(\mathcal{K}(D) - \frac{1}{4}\right)^2 + \mathcal{K}(D) \\ &\geq \mathcal{K}(D) \\ &= \frac{3}{4} - \mu_1 2^{-\lambda_1} - \mu_2 2^{-\lambda_2} \qquad (4.18) \end{aligned}$$

where (4.17) follows from Lemma 4.6.1 when both $Z_1 = Z_2$ and $Z_3 = Z_4$, from Lemma 4.7.1 when both $(Z_1, Z_2) = (1, 0)$ and $Z_3 = Z_4$, from Lemma 4.8.2 when $(Z_3, Z_4) = (0, 1)$; and (4.18) follows from (4.16).

The current case is then finished by applying the same reasoning used following (4.4) to the end of Part 1, Overlap Case 1.

- Overlap Case 2: $2l - k = 1$.

Let

$$Y_{i,j} = \begin{cases} CA^{l-2-n}0A^{n+k-l} & \text{if } i{=}j{=}0 \\ (D \cup 1A^{l-2}(0A^n{-}C))A^{k-l} & \text{if } i{=}1,\ j{=}0 \end{cases}$$

$$W_{i,j} = \begin{cases} A^{k-l}0A^{l-2}C & \text{if } i{=}j{=}0 \\ A^{k-l}(0A^n{-}C)A^{l-2-n}1A^n & \text{if } i{=}0,\ j{=}1. \end{cases}$$

Let $\mathcal{I} = A^3 - A1A$ and define the set F_3 by:

$$F_3 = \bigcup_{(Z_1,Z_2,Z_3)\in\mathcal{I}} (Y_{Z_1,Z_2} \cap W_{Z_2,Z_3})$$

where the terms in the union satisfy $Y_{Z_1,0} \cap W_{0,Z_3} \subseteq Z_1A^{l-2}0A^{l-2}Z_3A^n$. In this case, F_3 is comprised of 4 of the 8 possible sets obtained from the pattern $Z_1A^{l-2}Z_2A^{l-2}Z_3A^n$, namely excluding (Z_1, Z_2, Z_3) being $(0,1,0)$, $(1,1,0)$, $(0,1,1)$, or $(1,1,1)$. Therefore, these conditions are equivalent to $Z_2 \neq 1$.

The expected Kraft sum of F_3 can be lower bounded as follows:

$$\begin{aligned} E[\mathcal{K}(F_3)] &= \sum_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} E[\mathcal{K}(Y_{Z_1,Z_2} \cap W_{Z_2,Z_3})] \\ &= \frac{\mathcal{K}(C)^2}{2} + \frac{\mathcal{K}(C)\left(\frac{1}{2} - \mathcal{K}(C)\right)}{2} + \left(\mathcal{K}(C)\mathcal{K}(D) + \frac{\mathcal{K}(C)\left(\frac{1}{2} - \mathcal{K}(C)\right)}{2}\right) \\ &\quad + \left(0 + \frac{\frac{1}{2} - \mathcal{K}(C)}{4}\right) \qquad (4.19) \\ &= \frac{\mathcal{K}(C)}{2}\left(\frac{1}{2} - \mathcal{K}(C)\right) + \mathcal{K}(C)\mathcal{K}(D) + \frac{1}{8} \\ &\geq \left(\frac{\mathcal{K}(C)}{2} - \mathcal{K}(D)\right)\left(\frac{1}{2} - \mathcal{K}(C)\right) + \mathcal{K}(D) \qquad (4.20) \\ &\geq \mathcal{K}(D) \qquad (4.21) \\ &= \frac{3}{4} - \mu_1 2^{-\lambda_1} - \mu_2 2^{-\lambda_2} \qquad (4.22) \end{aligned}$$

where (4.19) follows from Lemma 4.6.1 when $Z_1 = Z_2 = Z_3 = 0$, from Lemma 4.7.1 when $(Z_1, Z_2, Z_3) = (1,0,0)$, and otherwise from Lemma 4.8.2 and the fact that $D \cap A^{k-l}(0A^n{-}C) = \varnothing$; (4.20) follows since $\frac{1}{8} \geq \frac{\mathcal{K}(D)}{2}$; (4.21) follows since $\mathcal{K}(C) \leq \frac{1}{2}$ and $\mathcal{K}(D) \leq \frac{\mathcal{K}(C)}{2}$; and (4.22) follows from (4.16).

The current case is then finished by applying the same reasoning used following (4.4) to the end of Part 1, Overlap Case 1.

- Overlap Case 3: $2l - k > 1$.

  Let

$$Y_{i,j} = \begin{cases} CA^{l-2-n}0A^{n+k-l} & \text{if } i{=}j{=}0 \\ (D \cup 1A^{l-2}(0A^n{-}C))A^{k-l} & \text{if } i{=}1,\ j{=}0 \end{cases}$$

$$W_{i,j} = \begin{cases} A^{k-l}0A^{l-2}C & \text{if } i{=}j{=}0 \\ A^{k-l}(0A^n{-}C)A^{l-2-n}1A^n & \text{if } i{=}0,\ j{=}1 \\ A^{k-l}D & \text{if } i{=}1,\ j{=}0. \end{cases}$$

  Let $\mathcal{I} = A^4 - (A^2 1 A \cup A1A1)$ and define the set F_3 by:

$$F_3 = \bigcup_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} (Y_{Z_1,Z_3} \cap W_{Z_2,Z_4}).$$

  where $Y_{Z_1,Z_3} \cap W_{Z_2,Z_4} \subseteq Z_1 A^{k-l-1} Z_2 A^{2l-k-2} Z_3 A^{k-l-1} Z_4 A^n$. Here F_3 is comprised of only 6 of the 16 possible sets, namely by excluding patterns with $(Z_1, Z_3) \in \{(0,1),(1,1)\}$ or $(Z_2, Z_4) = (1,1)$ from the union.

  Let 1_a be the indicator function for the condition $(k-l) \mid (2l-k-1)$, let 1_b be the indicator function for the condition $(2l-k-1) \mid (k-l)$, let 1_c be the indicator function for the condition $n > k-l-1$, and let 1_d be the indicator function for the condition $n > 2l-k-2$.

  Regarding 1_a and 1_b, note that $k-l$ is the length of any word in $Z_1 A^{k-l-1}$, and $2l-k-1$ is the length of any word in $Z_2 A^{2l-k-2}$. If $1_c = 1$, then any word from $0A^n$ that is a prefix of a term in the union above must extend at least to the bit Z_2, which would cause, for example, overlap in prefixes of $Y_{0,0}$ that lie in C and subwords of $W_{0,1}$ that lie in $0A^n{-}C$. Also, if $1_d = 1$, then any word from $0A^n$ that is a subword in a term in the union above that starts at the Z_2 position must extend at least to the bit Z_3, which would cause overlap in subwords of $Y_{1,0}$ that lie in $0A^n{-}C$ and subwords of $W_{0,1}$ that lie in $0A^n{-}C$. It turns out that there are four such complications that arise in this overlap case, which are considered in cases (ix)–(xii) of Lemma 4.8.2.

If $1_a = 1_b = 1_c = 1_d = 0$, then the expected Kraft sum of F_3 is

$$\begin{aligned} E[\mathcal{K}(F_3)] &= \sum_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} E[\mathcal{K}(Y_{Z_1,Z_3} \cap W_{Z_2,Z_4})] && (4.23) \\ &= \frac{\mathcal{K}(C)^2}{4} + \frac{\mathcal{K}(C)(\frac{1}{2} - \mathcal{K}(C))}{4} + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} \\ &\quad + \frac{\mathcal{K}(C)(\frac{1}{2} - \mathcal{K}(C))}{4} + \frac{\mathcal{K}(D)(\frac{1}{2} - \mathcal{K}(C))}{2} + \frac{(\frac{1}{2} - \mathcal{K}(C))^2}{4} \\ &\quad + \mathcal{K}(D)^2 + \frac{\mathcal{K}(D)(\frac{1}{2} - \mathcal{K}(C))}{2} && (4.24) \\ &= \left(\frac{1}{4} + \mathcal{K}(D)\right)^2 \end{aligned}$$

where (4.24) follows from Lemma 4.6.1 when $Z_1 = Z_2 = Z_3 = Z_4 = 0$, from Lemma 4.7.1 for part of the case when $(Z_1, Z_2, Z_3, Z_4) = (1, 0, 0, 0)$, and otherwise from Lemma 4.8.2 using the fact that $1_a = 1_b = 1_c = 1_d = 0$.

Of the 6 terms in the summation of (4.23), the 3 terms corresponding to (Z_1, Z_2, Z_3, Z_4) equaling $(0, 0, 0, 0)$, $(0, 1, 0, 0)$, and $(1, 0, 0, 0)$, remain the same even when it's not the case that $1_a = 1_b = 1_c = 1_d = 0$. The values of the remaining 3 terms of the summation, i.e., when (Z_1, Z_2, Z_3, Z_4) is $(0, 0, 0, 1)$, $(1, 0, 0, 1)$, or $(1, 1, 0, 1)$, are obtained from Lemma 4.8.2 using

$$\begin{aligned} &E[\mathcal{K}\left(CA^{l-2-n}0A^{n+k-l} \cap A^{k-l}(0A^n - C)A^{l-2-n}1A^n\right)] \\ &\qquad = \frac{\mathcal{K}(C)\mathcal{K}(0A^n - C)}{4} - 1_a 1_d \cdot \frac{\mathcal{K}(C)(\frac{1}{2} - \mathcal{K}(C))}{4(2^n - 1)} && (4.25) \\ &E[\mathcal{K}\left(1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}(0A^n - C)A^{l-2-n}1A^n\right)] \\ &\qquad = \frac{\mathcal{K}(0A^n - C)^2}{4} - 1_b 1_c \cdot \frac{\frac{1}{2} - \mathcal{K}(C)}{4(2^n - 1)}\mathcal{K}(C) && (4.26) \\ &E[\mathcal{K}\left(DA^{k-l} \cap A^{k-l}(0A^n - C)A^{l-2-n}1A^n\right)] \\ &\qquad = \frac{\mathcal{K}(D)\mathcal{K}(0A^n - C)}{2} + 1_b 1_c \cdot \frac{\frac{1}{2} - \mathcal{K}(C)}{2(2^n - 1)} \cdot \mathcal{K}(D) && (4.27) \\ &E[\mathcal{K}\left(DA^{k-l} \cap A^{k-l}D\right)] = \mathcal{K}(D)^2 - 1_a \cdot \frac{\mathcal{K}(D)}{|C| \cdot 2^{l-2} - 1}\left(\frac{\mathcal{K}(C)}{2} - \mathcal{K}(D)\right). && (4.28) \end{aligned}$$

The first expressions on the right hand sides of (4.25)–(4.28) correspond to those in the calculations used to obtain (4.24), i.e., when $1_a = 1_b = 1_c = 1_d = 0$. Therefore, in general, the expected Kraft sum of F_3 is

given by

$$E[\mathcal{K}(F_3)] = \left(\frac{1}{4} + \mathcal{K}(D)\right)^2 - 1_b 1_c \cdot \frac{(\frac{1}{2} - \mathcal{K}(C))(\frac{\mathcal{K}(C)}{2} - \mathcal{K}(D))}{2(2^n - 1)}$$
$$- 1_a 1_d \cdot \frac{\mathcal{K}(C)\left(\frac{1}{2} - \mathcal{K}(C)\right)}{4(2^n - 1)} - 1_a \cdot \frac{\mathcal{K}(D)\left(\frac{\mathcal{K}(C)}{2} - \mathcal{K}(D)\right)}{|C| \cdot 2^{l-2} - 1}. \tag{4.29}$$

We will show that for any binary values of 1_a, 1_b, 1_c, and 1_d, we have $E[\mathcal{K}(F_3)] \geq \mathcal{K}(D)$. Since $\mathcal{K}(C) \leq 1/2$ and $\mathcal{K}(D) \leq \mathcal{K}(C)/2$, the quantities multiplying $1_b 1_c$, $1_a 1_d$, and 1_a are all non-positive. Thus, it suffices to show that with $1_a = 1_b = 1_c = 1_d = 1$, the quantity in (4.29) minus $\mathcal{K}(D)$ is non-negative for any $\mathcal{K}(C) \in \{0\} \cup [\frac{1}{2^{n+1}}, \frac{1}{2} - \frac{1}{2^{n+1}}] \cup \{\frac{1}{2}\}$ and $\mathcal{K}(D) \in [0, \frac{\mathcal{K}(C)}{2} - \frac{1}{2^{n+l}}] \cup \{\frac{\mathcal{K}(C)}{2}\}$. These ranges for $\mathcal{K}(C)$ and $\mathcal{K}(D)$ are sufficient to finish the proof, since $|C|$ and $|D|$ are integers, and so it is not possible that $\mathcal{K}(C) \in (0, \frac{1}{2^{n+1}}) \cup (\frac{1}{2} - \frac{1}{2^{n+1}}, \frac{1}{2})$ or $\mathcal{K}(D) \in (\frac{\mathcal{K}(C)}{2} - \frac{1}{2^{n+l}}, \frac{\mathcal{K}(C)}{2})$.

Since $2l - k > 1$ and $k \geq 3$, we have $l \geq 3$. If $\mathcal{K}(C) = 0$, then the original multiset of lengths contains only two distinct values, and this case is covered by Theorem 4.3.1. So suppose $\mathcal{K}(C) \geq 1/2^{n+1}$. Then

$$E[\mathcal{K}(F_3)] - \mathcal{K}(D) = \left(\frac{1}{4} - \mathcal{K}(D)\right)^2 - 1_b 1_c \cdot \frac{(\frac{1}{2} - \mathcal{K}(C))\left(\frac{\mathcal{K}(C)}{2} - \mathcal{K}(D)\right)}{2(2^n - 1)}$$
$$- 1_a \cdot \frac{\mathcal{K}(D)\left(\frac{\mathcal{K}(C)}{2} - \mathcal{K}(D)\right)}{|C| \cdot 2^{l-2} - 1} - 1_a 1_d \cdot \frac{\mathcal{K}(C)\left(\frac{1}{2} - \mathcal{K}(C)\right)}{4(2^n - 1)} \tag{4.30}$$
$$\geq 0 \tag{4.31}$$

where (4.31) follows by first setting $1_a = 1_b = 1_c = 1_d = 1$ to minimize (4.30), and then applying Lemma 4.5.15 by setting $x = \mathcal{K}(C)$ and $y = \mathcal{K}(D)$. The current case is then finished by applying the same reasoning used following (4.4) to the end of Part 1, Overlap Case 1.

■

4.8.2 Proof of Theorem 4.8.1(b)

The proof of Theorem 4.8.1(b) uses the following lemma, whose proof can be found in the appendix.

Lemma 4.8.3. *Let $n, l, k \geq 1$ be integers such that $2 \leq l < k$ and $n \geq l - 1$. Let C be a set of a fixed size chosen uniformly at random from $0A^{l-2}1A^{n-(l-1)}$. Let $G = 0A^{l-2}1A^{n-(l-1)} - C$. Let D_1 be a set of a fixed size chosen uniformly at random from $1A^{l-2}1A^n$, and let D_2 be a set of a fixed size chosen uniformly at random from CA^{l-1}. For any $b \in A$, if $2l - k \neq 1$, then*

(i) $E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-l}bA^{l-2}C\right)] = \mathcal{K}(C)/8$

(ii) $E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-l}GA^{l-1}\right)] = (\frac{1}{4} - \mathcal{K}(C))/4$

(iii) $E[\mathcal{K}\left(D_1A^{k-l} \cap A^{k-l}bA^{l-2}C\right)] = \mathcal{K}(C)\,\mathcal{K}(D_1)/2$

(iv) $E[\mathcal{K}\left(D_1A^{k-l} \cap A^{k-l}GA^{l-1}\right)] = \mathcal{K}(D_1)\,(\frac{1}{4} - \mathcal{K}(C))$.

If $2l - k = 1$, *then*

(v) $E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-l}0A^{l-2}C\right)] = \mathcal{K}(C)/4$

(vi) $E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-l}GA^{l-1}\right)] = (\frac{1}{4} - \mathcal{K}(C))/2$

(vii) $E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-l}D_2\right)] = \mathcal{K}(D_2)/2$

(viii) $E[\mathcal{K}\left(D_2A^{k-l} \cap A^{k-l}1A^{l-2}C\right)] = \mathcal{K}(C)\,\mathcal{K}(D_2)$.

Proof of Theorem 4.8.1(b). Here we assume $\lambda_2 < 2\lambda_1$ (or equivalently $n > l - 2$ by (4.1)) and $\frac{1}{4} \leq \mu_1 2^{-\lambda_1} \leq \frac{1}{2}$.

Let C be a set of size $2^n - \mu_1$ chosen uniformly at random from among the 2^{n-1} length-λ_1 elements of $0A^{l-2}1A^{n-l+1}$. Since $\frac{1}{4} \leq \mu_1 2^{-\lambda_1} \leq \frac{1}{2}$, we have $2^{n-1} \leq \mu_1 \leq 2^n$, which implies $0 \leq |C| \leq 2^{n-1}$, so there are enough words from which to choose C.

Define the following (random) set:

$$F_1 = 0A^n - C.$$

- For Overlap Cases 1 and 3 below:

  Let D be a set of size $(\frac{3}{4} - \mu_1 2^{-\lambda_1})2^{\lambda_2} - \mu_2$ chosen uniformly at random from among the 2^{n+l-2} length-λ_2 elements of $1A^{l-2}1A^n$, and let

$$F_2 = (1A^{l-2}1A^n - D) \cup CA^{l-1}.$$

- For Overlap Case 2 below:

  Let D be a set of size $(\frac{3}{4} - \mu_1 2^{-\lambda_1})2^{\lambda_2} - \mu_2$ chosen uniformly at random from among the $2^{l-1} \cdot |C|$ length-λ_2 elements of CA^{l-1}, and let

$$F_2 = 1A^{l-2}1A^n \cup (CA^{l-1} - D).$$

Then

$$\mathcal{K}(D) = \left(\frac{3}{4} - \mu_1 2^{-\lambda_1}\right) - \mu_2 2^{-\lambda_2}, \tag{4.32}$$

so $0 \leq \mathcal{K}(D) \leq \frac{3}{4} - \mu_1 2^{-\lambda_1} - \frac{1}{2} + \frac{\mu_1 2^{-\lambda_1}}{2} = \frac{1}{4} - \frac{\mu_1 2^{-\lambda_1}}{2}$. This means there are enough words from which to choose D, since $\frac{1}{4} - \frac{\mu_1 2^{-\lambda_1}}{2} \leq \frac{1}{4} - \frac{1}{8} = \frac{1}{8} < \frac{1}{4} = \mathcal{K}\left(1A^{l-2}1A^n\right)$ and $\frac{1}{4} - \frac{\mu_1 2^{-\lambda_1}}{2} = \frac{1}{2}\left(\frac{1}{2} - \mu_1 2^{-\lambda_1}\right) \leq \frac{1}{2} - \mu_1 2^{-\lambda_1} = \mathcal{K}(C) = \mathcal{K}\left(CA^{l-1}\right)$.

Note that the words in F_2 all have first bit equal to 1 or prefix in C, and a 1 in the $(n+1)$th position from the right. These conditions guarantee that no word in F_1 is a prefix or suffix of a word in F_2.

Also, in all 3 cases,

$$\begin{aligned}
|F_1| &= |0A^n| - |C| = 2^n - (2^n - \mu_1) = \mu_1 \\
|F_2| &= |1A^{l-2}1A^n| + |CA^{l-1}| - |D| \\
&= 2^{n+l-2} + 2^{l-1}(2^n - \mu_1) - \left(\frac{3}{4} - \mu_1 2^{-\lambda_1}\right) 2^{\lambda_2} + \mu_2 \\
&= \mu_2
\end{aligned}$$

so the set F_1 contains μ_1 words, each of length λ_1, and F_2 contains μ_2 words, each of length λ_2.

In each case, we will also construct a third random set F_3, consisting of μ_3 words, each of length λ_3. The random set

$$F = F_1 \cup F_2 \cup F_3$$

on average meets the requirements of the desired fix-free code. The union of at least one non-random instance for each of F_1, F_2, and F_3 will then yield the asserted fix-free code.

- Overlap Case 1: $2l - k < 1$.

Let $G = 0A^{l-2}1A^{n-l+1} - C$, and so $E[\mathcal{K}(G)] = \frac{1}{4} - \mathcal{K}(C)$. Let

$$Y_{i,j} = \begin{cases} 1A^{l-2}0A^{n+k-l} & \text{if } i{=}1,\ j{=}0 \\ DA^{k-l} & \text{if } i{=}j{=}1 \end{cases} \qquad W_{i,j} = \begin{cases} A^{k-l}iA^{l-2}C & \text{if } j{=}0 \\ A^{k-l}GA^{l-1} & \text{if } i{=}0,\ j{=}1 \\ A^{k-l}D & \text{if } i{=}j{=}1. \end{cases}$$

Let $\mathcal{I} = A^4 - 0A^3$ and define the set F_3 by:

$$F_3 = \bigcup_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} (Y_{1,Z_2} \cap W_{Z_3,Z_4})$$

where $Y_{1,Z_2} \cap W_{Z_3,Z_4} \subseteq 1A^{l-2}Z_2A^{k-2l}Z_3A^{l-2}Z_4A^n$.

Here F_3 is comprised of only 8 of the 16 possible sets obtained from the pattern $Z_1A^{l-2}Z_2A^{k-2l}Z_3A^{l-2}Z_4A^n$, namely by excluding patterns with $Z_1 = 0$ from the union. One can verify that no words in F_1 or F_2 can be either prefixes or suffixes of any words in F_3.

The expected Kraft sum of F_3 is then lower bounded as follows:

$$\begin{aligned}
E[\mathcal{K}(F_3)] &= \sum_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} E[\mathcal{K}(Y_{Z_1,Z_2} \cap W_{Z_3,Z_4})] \\
&= \frac{\mathcal{K}(C)}{8} + \frac{\frac{1}{4} - \mathcal{K}(C)}{4} + \frac{\mathcal{K}(C)}{8} + \frac{\mathcal{K}(D)}{4} + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} \\
&\quad + \mathcal{K}(D)\left(\frac{1}{4} - \mathcal{K}(C)\right) + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} + \mathcal{K}(D)^2 \qquad (4.33) \\
&= \mathcal{K}(D) + \left(\mathcal{K}(D) - \frac{1}{4}\right)^2 \\
&\geq \mathcal{K}(D) \\
&= \frac{3}{4} - \mu_1 2^{-\lambda_1} - \mu_2 2^{-\lambda_2} \qquad (4.34)
\end{aligned}$$

where (4.33) follows from Lemma 4.6.1 when $Z_1 = Z_3 = Z_4 = 1$, and otherwise from Lemma 4.8.3; and (4.34) follows from (4.32).

The current case is then finished by applying the same reasoning used following (4.4) to the end of Part 1, Overlap Case 1.

- Overlap Case 2: $2l - k = 1$.

Let $G = 0A^{l-2}1A^{n-l+1} - C$, and so $E[\mathcal{K}(G)] = \frac{1}{4} - \mathcal{K}(C)$. Let

$$Y_{i,j} = \begin{cases} DA^{k-l} & \text{if } i{=}0,\ j{=}1 \\ 1A^{l-2}0A^{n+k-l} & \text{if } i{=}1,\ j{=}0 \end{cases} \qquad W_{i,j} = \begin{cases} A^{k-l}iA^{l-2}C & \text{if } j{=}0 \\ A^{k-l}(D \cup GA^{l-1}) & \text{if } i{=}0,\ j{=}1. \end{cases}$$

Let $\mathcal{I} = \{(0,1,0),(1,0,0),(1,0,1)\}$ and define the set F_3 by:

$$F_3 = \bigcup_{(Z_1,Z_2,Z_3)\in\mathcal{I}} (Y_{Z_1,Z_2} \cap W_{Z_2,Z_3})$$

where $Y_{Z_1,Z_2} \cap W_{Z_2,Z_3} \subseteq Z_1 A^{l-2} Z_2 A^{l-2} Z_3 A^n$. In this case, F_3 is comprised of 3 of the 8 possible sets obtained from the pattern $Z_1 A^{l-2} Z_2 A^{l-2} Z_3 A^n$.

The expected Kraft sum of F_3 can be lower bounded as follows:

$$\begin{aligned}
E[\mathcal{K}(F_3)] &= \sum_{(Z_1,Z_2,Z_3)\in\mathcal{I}} E[\mathcal{K}(Y_{Z_1,Z_2} \cap W_{Z_2,Z_3})] \\
&= \mathcal{K}(C)\,\mathcal{K}(D) + \frac{\mathcal{K}(C)}{4} + \frac{\mathcal{K}(D)}{2} + \frac{\frac{1}{4} - \mathcal{K}(C)}{2} && (4.35) \\
&= \frac{(1-2\mathcal{K}(C))(1-4\mathcal{K}(D))}{8} + \mathcal{K}(D) \\
&\geq \mathcal{K}(D) && (4.36) \\
&= \frac{3}{4} - \mu_1 2^{-\lambda_1} - \mu_2 2^{-\lambda_2} && (4.37)
\end{aligned}$$

where (4.35) follows from Lemma 4.8.3; (4.36) follows since $\mathcal{K}(C) \leq \frac{1}{2}$ and $\mathcal{K}(D) \leq \frac{1}{4}$; and (4.37) follows from (4.32).

The current case is then finished by applying the same reasoning used following (4.4) to the end of Part 1, Overlap Case 1.

- Overlap Case 3: $2l - k > 1$.

  This case follows from the same reasoning as in Overlap Case 1, except using $\geq$ in (4.33), since in this case (i.e., when $2l - k > 1$) Lemma 4.6.1 shows

$$E[\mathcal{K}(Y_{1,1} \cap W_{1,1})] = E[\mathcal{K}\left(DA^{k-l} \cap A^{k-l}D\right)] \geq \mathcal{K}(D)^2 .$$

■

4.8.3 Proof of Theorem 4.8.1(c)

The proof of Theorem 4.8.1(c) uses the following lemma, whose proof can be found in the appendix.

Lemma 4.8.4. *Let $n, l, k \geq 1$ be integers such that $2 \leq l < k$ and $n \geq l - 1$. Let C_0 be a set of a fixed size chosen uniformly at random from $0A^{l-2}0A^{n-l+1}$, and let $C = 0A^{l-2}1A^{n-l+1} \cup C_0$. For $i \in \{0,1\}$, let D_i be a set of a fixed size chosen uniformly at random from $iA^{l-2}1A^n$. For any $b \in A$, if $2l - k \neq 1$, then*

(i) $E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-l}bA^{l-2}C\right)] = \mathcal{K}(C)/8$

(ii) $E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}bA^{l-2}C\right)] = \mathcal{K}(C)(\mathcal{K}(C) - \frac{1}{4})/2$

(iii) $E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}D_1\right)] = (\mathcal{K}(C) - \frac{1}{4})\mathcal{K}(D_1)$

(iv) $E[\mathcal{K}\left(D_1A^{k-l} \cap A^{k-l}bA^{l-2}C\right)] = \mathcal{K}(C)\mathcal{K}(D_1)/2.$

If $2l - k = 1$, *then*

(v) $E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-l}0A^{l-2}C\right)] = \mathcal{K}(C)/4$

(vi) $E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}0A^{l-2}C\right)] = \mathcal{K}(C)(\mathcal{K}(C) - \frac{1}{4})$

(vii) $E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-l}D_0\right)] = \mathcal{K}(D_0)/2$

(viii) $E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}D_0\right)] = 2(\mathcal{K}(C) - \frac{1}{4})\mathcal{K}(D_0)$

(ix) $E[\mathcal{K}\left(D_0A^{k-1} \cap A^{k-l}1A^{l-2}C\right)] = \mathcal{K}(C)\mathcal{K}(D_0).$

Proof of Theorem 4.8.1(c). Here we assume $\lambda_2 < 2\lambda_1$ (or equivalently $n > l - 2$ by (4.1)) and $\mu_1 2^{-\lambda_1} < \frac{1}{4}$ and $\frac{1}{4} \leq \mu_2 2^{-\lambda_2} \leq \frac{1}{2}$.

Let $C = C_1 \cup C_0$ be a set of size $2^n - \mu_1$, where $C_1 = 0A^{l-2}1A^{n-l+1}$ and C_0 is chosen uniformly at random from among the 2^{n-1} length-λ_1 elements of $0A^{l-2}0A^{n-l+1}$. Note that $|C_0| = 2^n - \mu_1 - 2^{n-1} = 2^{n-1} - \mu_1$ and

$$\mathcal{K}(C_0) = \mathcal{K}(C) - \mathcal{K}(C_1) = \mathcal{K}(C) - \frac{1}{4}.$$

Since $0 \leq \mu_1 2^{-\lambda_1} < \frac{1}{4}$, we have $0 \leq \mu_1 \leq 2^{n-1}$, which shows $0 \leq |C_0| \leq 2^{n-1} = |0A^{l-2}0A^{n-l+1}|$, and so there are enough words from which to choose C_0. Define the following (random) set:

$$F_1 = 0A^n - C.$$

- For Overlap Cases 1 and 3 below:

  Let D be a set of size $2^{\lambda_2 - 1} - \mu_2$ chosen uniformly at random from among the 2^{n+l-2} length-λ_2 elements of $1A^{l-2}1A^n$, and let

$$F_2 = (1A^{l-2}1A^n - D) \cup 0A^{l-2}1A^n.$$

- For Overlap Case 2 below:

  Let D be a set of size $2^{\lambda_2-1}-\mu_2$ chosen uniformly at random from among the 2^{n+l-2} length-λ_2 elements of $0A^{l-2}1A^n$, and let

$$F_2 = 1A^{l-2}1A^n \cup (0A^{l-2}1A^n - D).$$

Then $\mathcal{K}(D) = \frac{1}{2} - \mu_2 2^{-\lambda_2}$, so $0 \leq \mathcal{K}(D) \leq \frac{1}{2} - \frac{1}{4} = \frac{1}{4}$. This means there are enough words from which to choose D, since $\frac{1}{4} = \mathcal{K}\left(1A^{l-2}1A^n\right) = \mathcal{K}\left(0A^{l-2}1A^n\right)$. Note that none of the words in F_2 start with a word from $0A^{l-2}0A^{n-l+1}$ or end with a word from $0A^n$, and so no word in $F_1 \subseteq 0A^{l-2}0A^{n-l+1}$ is a prefix or suffix of any word in F_2.

Also, in all 3 cases,

$$\begin{aligned}
|F_1| &= |0A^n| - |C| = 2^n - (2^n - \mu_1) = \mu_1 \\
|F_2| &= |1A^{l-2}1A^n| + |0A^{l-2}1A^n| - |D| \\
&= 2^{n+l-2} + 2^{n+l-2} - 2^{\lambda_2-1} + \mu_2 \\
&= \mu_2
\end{aligned}$$

so the set F_1 contains μ_1 words, each of length λ_1, and F_2 contains μ_2 words, each of length λ_2.

In each case, we will also construct a third random set F_3, consisting of μ_3 words, each of length λ_3. The random set

$$F = F_1 \cup F_2 \cup F_3$$

on average forms the desired fix-free code. The union of non-random instances of F_1, F_2, and F_3 will then yield the asserted fix-free code.

- Overlap Case 1: $2l - k < 1$.

  Let

$$Y_{i,j} = \begin{cases} C_0 A^{k-1} & \text{if } i{=}j{=}0 \\ 1A^{l-2}0A^{n+k-l} & \text{if } i{=}1,\ j{=}0 \\ DA^{k-l} & \text{if } i{=}j{=}1 \end{cases} \qquad W_{i,j} = \begin{cases} A^{k-l}iA^{l-2}C & \text{if } j{=}0 \\ A^{k-l}D & \text{if } i{=}j{=}1. \end{cases}$$

Let $\mathcal{I} = A^4 - (01A^2 \cup A^201)$ and define the set F_3 by:

$$F_3 = \bigcup_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} (Y_{Z_1,Z_2} \cap W_{Z_3,Z_4})$$

where $Y_{Z_1,Z_2} \cap W_{Z_3,Z_4} \subseteq Z_1A^{l-2}Z_2A^{k-2l}Z_3A^{l-2}Z_4A^n$.

Here F_3 is comprised of only 9 of the 16 possible sets obtained from the pattern $Z_1A^{l-2}Z_2A^{k-2l}Z_3A^{l-2}Z_4A^n$, namely by excluding patterns with $(Z_1, Z_2) = (0, 1)$ or $(Z_3, Z_4) = (0, 1)$ from the union. One can verify that no words in F_1 or F_2 can be either prefixes or suffixes of any words in F_3.

The expected Kraft sum of F_3 is then lower bounded as follows:

$$\begin{aligned}
E[\mathcal{K}(F_3)] &= \sum_{(Z_1,Z_2,Z_3,Z_4)\in\mathcal{I}} E[\mathcal{K}(Y_{Z_1,Z_2} \cap W_{Z_3,Z_4})] \\
&= \frac{\mathcal{K}(C)(\mathcal{K}(C) - \frac{1}{4})}{2} + \frac{\mathcal{K}(C)(\mathcal{K}(C) - \frac{1}{4})}{2} + \mathcal{K}(D)\left(\mathcal{K}(C) - \frac{1}{4}\right) + \frac{\mathcal{K}(C)}{8} \\
&\quad + \frac{\mathcal{K}(C)}{8} + \frac{\mathcal{K}(D)}{4} + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} + \frac{\mathcal{K}(C)\mathcal{K}(D)}{2} + \mathcal{K}(D)^2 \qquad (4.38) \\
&= \mathcal{K}(C) + \mathcal{K}(D) - \frac{1}{4} + \left(\mathcal{K}(C) + \mathcal{K}(D) - \frac{1}{2}\right)^2 \\
&\geq \mathcal{K}(C) + \mathcal{K}(D) - \frac{1}{4} \\
&= \frac{3}{4} - \mu_1 2^{-\lambda_1} - \mu_2 2^{-\lambda_2} \qquad (4.39)
\end{aligned}$$

where (4.38) follows from Lemma 4.6.1 when $Z_1 = Z_3 = Z_4 = 1$, and otherwise from Lemma 4.8.4; and (4.39) follows from the quantities $|C|$ and $|D|$ stated earlier in the proof.

The current case is then finished by applying the same reasoning used following (4.4) to the end of Part 1, Overlap Case 1.

- Overlap Case 2: $2l - k = 1$.

Let

$$Y_{i,j} = \begin{cases} C_0A^{k-1} & \text{if } i{=}j{=}0 \\ 1A^{l-2}0A^{n+k-l} & \text{if } i{=}1,\ j{=}0 \\ DA^{k-l} & \text{if } i{=}0,\ j{=}1 \end{cases} \qquad W_{i,j} = \begin{cases} A^{k-l}iA^{l-2}C & \text{if } j{=}0 \\ A^{k-l}D & \text{if } i{=}0,\ j{=}1. \end{cases}$$

Let $\mathcal{I} = A^3 - (11A \cup A11)$ and define the set F_3 by:

$$F_3 = \bigcup_{(Z_1,Z_2,Z_3)\in\mathcal{I}} (Y_{Z_1,Z_2} \cap W_{Z_2,Z_3})$$

where $Y_{Z_1,Z_2} \cap W_{Z_2,Z_3} \subseteq Z_1 A^{l-2} Z_2 A^{l-2} Z_3 A^n$. In this case, F_3 is comprised of only 5 of the 8 possible sets obtained from the pattern $Z_1 A^{l-2} Z_2 A^{l-2} Z_3 A^n$, namely excluding (Z_1, Z_2, Z_3) being $(1,1,0)$, $(0,1,1)$, or $(1,1,1)$.

The expected Kraft sum of F_3 can be lower bounded as follows:

$$\begin{aligned}
E[\mathcal{K}(F_3)] &= \sum_{(Z_1,Z_2,Z_3)\in\mathcal{I}} E[\mathcal{K}(Y_{Z_1,Z_2} \cap W_{Z_2,Z_3})] \\
&= \mathcal{K}(C)\left(\mathcal{K}(C) - \frac{1}{4}\right) + 2\mathcal{K}(D)\left(\mathcal{K}(C) - \frac{1}{4}\right) + \mathcal{K}(C)\,\mathcal{K}(D) \\
&\quad + \frac{\mathcal{K}(C)}{4} + \frac{\mathcal{K}(D)}{2} && (4.40) \\
&= \mathcal{K}(C) + \mathcal{K}(D) - \frac{1}{4} + \left(\mathcal{K}(C) + \mathcal{K}(D) - \frac{1}{2}\right)^2 + \mathcal{K}(D)\left(\mathcal{K}(C) - \mathcal{K}(D)\right) \\
&\geq \mathcal{K}(C) + \mathcal{K}(D) - \frac{1}{4} && (4.41) \\
&= \frac{3}{4} - \mu_1 2^{-\lambda_1} - \mu_2 2^{-\lambda_2} && (4.42)
\end{aligned}$$

where (4.40) follows from Lemma 4.8.4; (4.41) follows since $\mathcal{K}(D) \leq \frac{1}{4} \leq \mathcal{K}(C)$; and (4.42) follows from the quantities $|C|$ and $|D|$ stated earlier in the proof.

The current case is then finished by applying the same reasoning used following (4.4) to the end of Part 1, Overlap Case 1.

- Overlap Case 3: $2l - k > 1$.

This case follows from the same reasoning as in Overlap Case 1, except using $\geq$ in (4.38), since in this case (i.e., when $2l - k > 1$) Lemma 4.6.1 shows

$$E[\mathcal{K}(Y_{1,1} \cap W_{1,1})] = E[\mathcal{K}\left(DA^{k-l} \cap A^{k-l}D\right)] \geq \mathcal{K}(D)^2 .$$

■

4.8.4 Proof of Theorem 4.8.1(d)

The proof of Theorem 4.8.1(d) uses the following lemma, whose proof can be found in the appendix.

Lemma 4.8.5. *Let $n, l, k \geq 1$ be integers such that $2 \leq l < k$ and $n \geq l - 1$. Let C_0 be a set of a fixed size chosen uniformly at random from $0A^{l-2}0A^{n-l+1}$, and let $C = 0A^{l-2}1A^{n-l+1} \cup C_0$. Let D be a set of a fixed size chosen uniformly at random from $1A^{l-2}C$. Then*

(i) $E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-1}C\right)] = \mathcal{K}(C)/4$

(ii) $E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-1}C\right)] = \mathcal{K}(C)\left(\mathcal{K}(C) - \frac{1}{4}\right)$

(iii) $E[\mathcal{K}\left(DA^{k-l} \cap A^{k-1}C\right)]$

$$= \begin{cases} \mathcal{K}(C)\,\mathcal{K}(D) & \text{if } 2l-k<1 \\ 2(\mathcal{K}(C) - \frac{1}{4})\mathcal{K}(D) & \text{if } 2l-k=1 \\ \mathcal{K}(C)\,\mathcal{K}(D) & \text{if } 2l-k>1 \text{ and } (k-l) \nmid (2l-k-1) \\ \mathcal{K}(C)\,\mathcal{K}(D) + \frac{\mathcal{K}(D)(\mathcal{K}(C)-\frac{1}{4})(\frac{1}{2}-\mathcal{K}(C))}{\mathcal{K}(C)(2^{n-1}-1)} & \text{if } 2l-k>1 \text{ and } (k-l) \mid (2l-k-1). \end{cases}$$

If $2l - k < 1$, then

(iv) $E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-l}D\right)] = \mathcal{K}(D)/4$

(v) $E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}D\right)] = (\mathcal{K}(C) - \frac{1}{4})\mathcal{K}(D)$.

Proof of Theorem 4.8.1(d). Here we assume $\lambda_2 < 2\lambda_1$ (or equivalently $n > l - 2$ by (4.1)) and $\mu_1 2^{-\lambda_1} < \frac{1}{4}$ and $\mu_2 2^{-\lambda_2} > \frac{1}{2}$.

Let $C = 0A^{l-2}1A^{n-l+1} \cup C_0$ be a set of size $2^n - \mu_1$, where C_0 is chosen uniformly at random from among the 2^{n-1} length-λ_1 elements of $0A^{l-2}0A^{n-l+1}$. Note that $|C_0| = 2^n - \mu_1 - 2^{n-1} = 2^{n-1} - \mu_1$, and thus $|C_0| \leq 2^{n-1}$, so there are enough words from which to choose C_0. Let D be a set of size $\mu_2 - 2^{\lambda_2 - 1}$ chosen uniformly at random from among the $2^{l-2} \cdot |C|$ length-λ_2 elements of $1A^{l-2}C$. Define the following (random) sets:

$$F_1 = 0A^n - C$$
$$F_2 = 1A^{l-2}1A^n \cup 0A^{l-2}1A^n \cup D.$$

Then $\mathcal{K}(D) = \mu_2 2^{-\lambda_2} - \frac{1}{2}$, so $0 \leq \mathcal{K}(D) \leq (\frac{3}{4} - \mu_1 2^{-\lambda_1}) - \frac{1}{2} = \frac{1}{4} - \mu_1 2^{-\lambda_1} \leq \frac{1}{2}(\frac{1}{2} - \mu_1 2^{-\lambda_1}) = \mathcal{K}\left(1A^{l-2}C\right)$, which means there are enough words from which to choose D. None of the words in F_2 start with a word from

$0A^{l-2}0A^{n-l+1}$ or end with a word from F_1, and so no word in $F_1 \subseteq 0A^{l-2}0A^{n-l+1}$ is a prefix or suffix of any word in F_2. Also in all 3 cases,

$$\begin{aligned}
|F_1| &= |0A^n| - |C| = 2^n - (2^n - \mu_1) = \mu_1 \\
|F_2| &= |1A^{l-2}1A^n| + |0A^{l-2}1A^n| + |D| \\
&= 2^{n+l-2} + 2^{n+l-2} + \mu_2 - 2^{\lambda_2 - 1} \\
&= \mu_2
\end{aligned}$$

so the set F_1 contains μ_1 words, each of length λ_1, and F_2 contains μ_2 words, each of length λ_2.

In each case, we will also construct a third random set F_3, consisting of μ_3 words, each of length λ_3. The random set

$$F = F_1 \cup F_2 \cup F_3$$

on average forms the desired fix-free code. The union of non-random instances of F_1, F_2, and F_3 will then yield the asserted fix-free code.

- Overlap Case 1: $2l - k < 1$.

  Define the following sets:

$$\begin{aligned}
F_{3,1} &= 1A^{l-2}0A^{n+k-l} \cap A^{k-1}C \\
F_{3,2} &= C_0A^{k-1} \cap A^{k-1}C \\
F_{3,3} &= DA^{k-l} \cap A^{k-1}C \\
F_{3,4} &= 1A^{l-2}0A^{n+k-l} \cap A^{k-l}D \\
F_{3,5} &= C_0A^{k-1} \cap A^{k-l}D \\
F_3 &= (F_{3,1} \cup F_{3,2}) - (F_{3,3} \cup F_{3,4} \cup F_{3,5}).
\end{aligned}$$

  Each set $F_{3,p}$ consists of words of length λ_3, and these sets are random, since they involve the random sets C or D. It is easy to verify that none of the words of F_1 (respectively, F_2) are prefixes or suffixes of any words in F_2 or F_3 (respectively, F_3), and that $F_{3,1}$ and $F_{3,2}$ are disjoint. Note that $F_{3,3} \cup F_{3,4} \cup F_{3,5}$ is the

set of all words of $F_{3,1} \cup F_{3,2}$ that have some word of D as a prefix or suffix. We have

$$\begin{aligned}
E[\mathcal{K}(F_{3,1} \cup F_{3,2})] &= E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-1}C\right)] + E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-1}C\right)] && (4.43)\\
&= \frac{\mathcal{K}(C)}{4} + \mathcal{K}(C_0)\,\mathcal{K}(C) && (4.44)\\
&= \mathcal{K}(C)^2 \\
E[\mathcal{K}(F_{3,3} \cup F_{3,4} \cup F_{3,5})] &\leq E[\mathcal{K}\left(DA^{k-l} \cap A^{k-1}C\right)] + E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-l}D\right)] \\
&\quad + E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}D\right)] \\
&= \mathcal{K}(C)\,\mathcal{K}(D) + \frac{\mathcal{K}(D)}{4} + \mathcal{K}(C_0)\,\mathcal{K}(D) && (4.45)\\
&= 2\mathcal{K}(C)\,\mathcal{K}(D) \\
E[\mathcal{K}(F_3)] &= E[\mathcal{K}(F_{3,1} \cup F_{3,2})] - E[\mathcal{K}(F_{3,3} \cup F_{3,4} \cup F_{3,5})] && (4.46)\\
&\geq \mathcal{K}(C)^2 - 2\mathcal{K}(C)\,\mathcal{K}(D).
\end{aligned}$$

$$\begin{aligned}
&E[\mathcal{K}(F_1 \cup F_2 \cup F_3)] \\
&= E[\mathcal{K}(F_1)] + E[\mathcal{K}(F_2)] + E[\mathcal{K}(F_3)] \\
&= E[\mathcal{K}(0A^n{-}C)] + E[\mathcal{K}\left(1A^{l-2}1A^n\right)] + E[\mathcal{K}\left(0A^{l-2}1A^n\right)] + E[\mathcal{K}(D)] + E[\mathcal{K}(F_3)] \\
&\geq \frac{1}{2} - \mathcal{K}(C) + \frac{1}{4} + \frac{1}{4} + \mathcal{K}(D) + \mathcal{K}(C)^2 - 2\mathcal{K}(C)\,\mathcal{K}(D) && (4.47)\\
&= \frac{3}{4} + \left(\frac{1}{2} - \mathcal{K}(C)\right)\left(\frac{1}{2} - \mathcal{K}(C) + 2\mathcal{K}(D)\right) \\
&\geq \frac{3}{4} && (4.48)
\end{aligned}$$

where (4.43) follows since $F_{3,1}$ and $F_{3,2}$ are disjoint; (4.44) and (4.45) follow from Lemma 4.8.5; (4.46) follows from $F_{3,3} \cup F_{3,4} \cup F_{3,5} \subseteq F_{3,1} \cup F_{3,2}$; (4.47) follows from Lemma 4.5.2; and (4.48) follows since $\mathcal{K}(C) \leq \frac{1}{2}$ and $\mathcal{K}(D) \geq 0$.

The current case is then finished by applying the same reasoning used following (4.4) to the end of Part 1, Overlap Case 1.

- Overlap Case 2: $2l - k = 1$.

  Define the following sets:

$$F_{3,1} = 1A^{l-2}0A^{n+l-1} \cap A^{k-1}C$$
$$F_{3,2} = C_0A^{k-1} \cap A^{k-1}C$$
$$F_3 = (F_{3,1} \cup F_{3,2}) - (DA^{k-l} \cap A^{k-1}C).$$

The sets $F_{3,1}$, $F_{3,2}$, and $(DA^{k-l} \cap A^{k-1}C)$ consist of words of length λ_3, and these sets are random since they involve the random sets C or D. It is easy to verify that none of the words of F_1 (respectively, F_2) are prefixes or suffixes of any words in F_2 or F_3 (respectively, F_3), and that $F_{3,1}$ and $F_{3,2}$ are disjoint. Also note that $DA^{k-l} \cap A^{k-1}C$ is the set of all words of $F_{3,1} \cup F_{3,2}$ that have some word of D as a prefix or suffix.

Then we have

$$E[\mathcal{K}(F_{3,1} \cup F_{3,2})] = E[1A^{l-2}0A^{n+l-1} \cap A^{k-1}C] + E[C_0A^{k-1} \cap A^{k-1}C] \tag{4.49}$$
$$= \frac{\mathcal{K}(C)}{4} + \mathcal{K}(C_0)\,\mathcal{K}(C) = \mathcal{K}(C)^2 \tag{4.50}$$
$$E[\mathcal{K}\left(DA^{k-l} \cap A^{k-1}C\right)] = 2\left(\mathcal{K}(C) - \frac{1}{4}\right)\mathcal{K}(D) \tag{4.51}$$
$$E[\mathcal{K}(F_3)] = \mathcal{K}(C)^2 - 2\left(\mathcal{K}(C) - \frac{1}{4}\right)\mathcal{K}(D) \tag{4.52}$$
$$E[\mathcal{K}(F_1 \cup F_2 \cup F_3)] = E[\mathcal{K}(F_1)] + E[\mathcal{K}(F_2)] + E[\mathcal{K}(F_3)]$$
$$= E[\mathcal{K}(0A^n - C)] + E[\mathcal{K}\left(1A^{l-2}1A^n\right)]$$
$$+ E[\mathcal{K}\left(0A^{l-2}1A^n\right)] + E[\mathcal{K}(D)] + E[\mathcal{K}(F_3)]$$
$$= \left(\frac{1}{2} - \mathcal{K}(C)\right) + \frac{1}{4} + \frac{1}{4} + \mathcal{K}(D) + \mathcal{K}(C)^2 - 2\left(\mathcal{K}(C) - \frac{1}{4}\right)\mathcal{K}(D) \tag{4.53}$$
$$= \frac{3}{4} + \left(\frac{1}{2} - \mathcal{K}(C)\right)^2 + 2\mathcal{K}(D)\left(\frac{3}{4} - \mathcal{K}(C)\right)$$
$$\geq \frac{3}{4} \tag{4.54}$$

where (4.49) follows since $F_{3,1}$ and $F_{3,2}$ are disjoint; (4.51) follows from Lemma 4.5.11; (4.52) follows from $DA^{k-l} \cap A^{k-1}C \subseteq F_{3,1} \cup F_{3,2}$; (4.53) follows from Lemma 4.5.2; and (4.54) follows since $\mathcal{K}(C) \leq \frac{1}{2} < \frac{3}{4}$.

The current case is then finished by applying the same reasoning used following (4.4) to the end of Part 1, Overlap Case 1.

- Overlap Case 3: $2l - k > 1$.

If $n = 1$ then $\mu_1 2^{-\lambda_1} < \frac{1}{4}$ implies $\mu_1 2^{-\lambda_1} = 0$, in which case the proof is covered by Theorem 4.3.1 since then there are codewords of only two distinct lengths λ_2 and λ_3. So assume $n \geq 2$.

Of the calculated expected values of the Kraft sums of $F_{3,1}$, $F_{3,2}$, $F_{3,3}$, $F_{3,4}$, and $F_{3,5}$ in Overlap Case 1, the only quantity that changes under the condition of Overlap Case 3 is $E[\mathcal{K}(F_{3,3})]$, as seen in Lemma 4.5.11. In particular, since $2l - k > 1$ in this case, we have

$$E[\mathcal{K}(F_{3,3})] = E[\mathcal{K}\left(DA^{k-l} \cap A^{k-1}C\right)] \leq \mathcal{K}(C)\,\mathcal{K}(D) + \frac{\mathcal{K}(D)\,(\mathcal{K}(C) - \frac{1}{4})(\frac{1}{2} - \mathcal{K}(C))}{\mathcal{K}(C)\,(2^{n-1} - 1)}.$$

Therefore, in the calculation of $E[\mathcal{K}(F_1 \cup F_2 \cup F_3)]$, we get the lower bound

$$\begin{aligned}
&E[\mathcal{K}(F_1 \cup F_2 \cup F_3)] \\
&\geq \frac{3}{4} + \left(\frac{1}{2} - \mathcal{K}(C)\right)\left(\frac{1}{2} - \mathcal{K}(C) + 2\mathcal{K}(D)\right) - \frac{\mathcal{K}(D)\,(\mathcal{K}(C) - \frac{1}{4})(\frac{1}{2} - \mathcal{K}(C))}{\mathcal{K}(C)\,(2^{n-1} - 1)} \\
&= \frac{3}{4} + \left(\frac{1}{2} - \mathcal{K}(C)\right)\left(\frac{1}{2} - \mathcal{K}(C) + 2\mathcal{K}(D) \cdot \frac{\mathcal{K}(C)\,(2^n - 3) + \frac{1}{4}}{\mathcal{K}(C)\,(2^n - 2)}\right) \\
&\geq \frac{3}{4} \qquad (4.55)
\end{aligned}$$

where (4.55) follows from $\mathcal{K}(C) \leq \frac{1}{2}$ and $2^n \geq 4$. The current case is then finished by applying the same reasoning used following (4.4) to the end of Part 1, Overlap Case 1.

■

4.A Proofs of lemmas

Proof of Lemma 4.5.1. Suppose a sequence of positive integers consists of $\mu_n > 0$ occurrences of integer l_n, for $1 \leq n \leq M$. Suppose its Kraft sum is less than $3/4$ and define

$$\mu'_n = \begin{cases} \mu_n & \text{if } 1 \leq n \leq M-1 \\ 3 \cdot 2^{l_M - 2} - \sum_{k=1}^{M-1} \mu_k 2^{l_M - k} & \text{if } n = M. \end{cases}$$

Note that

$$\mu'_M = \left(\frac{3}{4} - \sum_{k=1}^{M} \mu_k 2^{-k} \right) 2^{l_M} + \mu_M > \mu_M$$

and the new Kraft sum is

$$\sum_{n=1}^{M} \mu'_n 2^{-l_n} = \sum_{n=1}^{M-1} \mu_n 2^{-l_n} + \frac{3}{4} - \sum_{k=1}^{M} \mu_k 2^{-k} + \mu_M 2^{-l_M} = \frac{3}{4}.$$

If the sequence with multiplicities $\{\mu'_n\}$ has a fix-free code, then discarding any $\mu'_M - \mu_M$ codewords of length M yields a fix-free code for the sequence with multiplicities $\{\mu_n\}$. ■

Proof of Lemma 4.5.3. Suppose $w \in R_l(U)$. Then the ith bit of the length-l prefix of w must be the same as the ith bit of the length-l suffix of w (which lies at position $i + m - l$). In other words, $w \in R_l(U)$ if and only if $w \in U$ and $w_i = w_{i+m-l}$ for all $1 \leq i \leq l$. The condition on w_i is equivalent to w_i being constant whenever i is congruent to p mod $(m - l)$, and $1 \leq i \leq m$, and $p \in \{1, \ldots, m - l\}$.

For any word $w \in R_l(U)$, the constant bit value w_i associated with each congruence class can be assigned independently of any other congruence class. Thus, the cardinality of $R_l(U)$ is equal to the product of the number N_p of allowable constant bit values for each congruence class. That is, $|R_l(U)| = \prod_{p=1}^{m-l} N_p$.

Let $I_p = \{i \in \{1, \ldots, m\} \mid i \equiv p \bmod (m-l)\}$ be the set of positions in w that are in the pth congruence class. If $U_i = A$ for each $i \in I_p$, then $N_p = 2$, since any word $w \in R_l(U)$ could have either a 0 or 1 in the positions of I_p. If there exist $i, j \in I_p$ such that $U_i = 0$ and $U_j = 1$, then $N_p = 0$, since there is no way to label the positions in I_p with a constant bit value. Otherwise, $N_p = 1$, since then there exists at least one $i \in I_p$ such that $U_i \in \{0, 1\}$, and $U_j \in \{U_i, A\}$ for every other $j \in I_p$. Hence, N_p equals the cardinality of the intersection of the sets U_i taken over all $i \in I_p$. ■

Proof of Lemma 4.5.4. For each $i \in \{1, 2\}$, let $g_i = |\{j : (X_i)_j = A\}|$ be the number of positions in X_i that are

not fixed points. Then for all $u \in U_1 \cap U_2$, by independence, we have

$$\begin{aligned}
E\left[\mathcal{K}\left(W_1 \cap W_2\right)\right] &= E\left[\sum_{u \in U_1 \cap U_2} 1_{W_1 \cap W_2}(u) 2^{-m}\right] \\
&= 2^{-m} \sum_{u \in U_1 \cap U_2} P(u \in (W_1 \cap W_2)) \\
&= 2^{-m} \sum_{u \in U_1 \cap U_2} P(u \in W_1) P(u \in W_2) \\
&= 2^{-m} \sum_{u \in U_1 \cap U_2} P(u \in A^a Y_1 A^b) P(u \in A^c Y_2 A^d) \\
&= 2^{-m} \sum_{u \in U_1 \cap U_2} \prod_{i=1}^{2} \frac{|Y_i| \cdot 2^{m-m_i}}{2^{g_i+m-m_i}} \\
&= \frac{|U_1 \cap U_2|}{2^m} \cdot \prod_{i=1}^{2} \frac{|Y_i|/2^m}{2^{g_i}/2^m} \\
&= \mathcal{K}\left(U_1 \cap U_2\right) \prod_{i=1}^{2} \frac{\mathcal{K}\left(Y_i\right)}{\mathcal{K}\left(X_i\right)}. \qquad (4.56)
\end{aligned}$$

Let f_V denote the set of positions where V has a fixed point. Then $f_{U_1 \cap U_2} = f_{U_1} \cup f_{U_2}$, so using Lemma 4.5.2,

$$\begin{aligned}
\mathcal{K}\left(U_1 \cap U_2\right) = 2^{-|f_{U_1 \cap U_2}|} &= 2^{-|f_{U_1} \cup f_{U_2}|} \\
&= 2^p \cdot 2^{-|f_{U_1}|} 2^{-|f_{U_2}|} \\
&= 2^p \cdot \mathcal{K}\left(U_1\right) \mathcal{K}\left(U_2\right).
\end{aligned}$$

Combining this with (4.56) proves the lemma. ■

Proof of Corollary 4.5.5. By Lemma 4.5.4,

$$\begin{aligned}
E[\mathcal{K}\left(A^a Y A^b \cap U\right)] &= E[\mathcal{K}\left((A^a Y A^b \cap U) \cap (A^{n+k} \cap A^{n+k})\right)] \\
&= \mathcal{K}\left(U\right) \cdot \frac{\mathcal{K}\left(Y\right)}{\mathcal{K}\left(X\right)} \cdot \mathcal{K}\left(A^{n+k}\right) \cdot \frac{\mathcal{K}\left(A^{n+k}\right)}{\mathcal{K}\left(A^{n+k}\right)} \\
&= \mathcal{K}\left(U\right) \cdot \frac{\mathcal{K}\left(Y\right)}{\mathcal{K}\left(X\right)}.
\end{aligned}$$

■

Proof of Lemma 4.5.6. If $|C| = 0$, then clearly the lemma holds. Suppose $|C| \geq 1$. If $u \in X$, then the probability that $u \in C$ is

$$\frac{\binom{|X|-1}{|C|-1}}{\binom{|X|}{|C|}} = \frac{|C|}{|X|}.$$

Now suppose $|C| \geq 2$. If $u, v \in X$ are distinct, then the probability that $u, v \in C$ is

$$\frac{\binom{|X|-2}{|C|-2}}{\binom{|X|}{|C|}} = \frac{|C|(|C|-1)}{|X|(|X|-1)}.$$

Finally, note that this last equation also fits the $|C| = 1$ case, since then the probability that such particular distinct u and v lie in C is zero, as $|C|$ contains only one element. ■

Proof of Lemma 4.5.7. Let $X = CA^{p+1} \cap bUbA^n \cap A^{p+1}C$. By Lemma 4.5.3, $|R_{n+1}(bUbA^n)| = |U|$, and so $|bUbA^n - R_{n+1}(bUbA^n)| = |U| \cdot 2^n - |U| = |U| \cdot (2^n - 1)$. A word of $R_{n+1}(bUbA^n)$ is in X if its $(n+1)$-bit prefix (which is also its $(n+1)$-bit suffix) is selected during the construction of C, and a word of $bUbA^n - R_{n+1}(bUbA^n)$ is in X if the distinct $(n+1)$-bit prefix and suffix are both selected during the construction of C. Thus the expected number of words of $bUbA^n$ with a prefix and a suffix in C is

$$\begin{aligned}
E[|CA^{p+1} \cap bUbA^n \cap A^{p+1}C|] &= E\left[\sum_{v \in bUbA^n} 1_{CA^{p+1} \cap A^{p+1}C}(v)\right] \\
&= \sum_{v \in bUbA^n} P\{v \in CA^{p+1} \cap A^{p+1}C\} \\
&= \sum_{v \in bUbA^n} P\{\exists w \in C : v \in wA^{p+1} \cap A^{p+1}w\} \\
&\quad + \sum_{v \in bUbA^n} P\{v \in CA^{p+1} \cap A^{p+1}C,\ \nexists w \in C : v \in wA^{p+1} \cap A^{p+1}w\} \\
&= |U| \cdot \frac{|C|}{2^n} + |U| \cdot (2^n - 1)\frac{|C| \cdot (|C|-1)}{2^n(2^n-1)} \qquad (4.57) \\
&= |U| \cdot \frac{|C|^2}{2^n},
\end{aligned}$$

where (4.57) follows using Lemma 4.5.6. These words all have length $p + 2 + n$, so their expected Kraft sum is

$$\begin{aligned}
E[\mathcal{K}\left(CA^{p+1} \cap bUbA^n \cap A^{p+1}C\right)] &= \frac{E[|CA^{p+1} \cap bUbA^n \cap A^{p+1}C|]}{2^{p+n+2}} \\
&= \frac{1}{2^{p+2+n}} \cdot |U| \cdot \frac{|C|^2}{2^n} \\
&= \frac{|U|}{2^p} \cdot \left(\frac{|C|}{2^{n+1}}\right)^2 \\
&= \mathcal{K}(U)\,\mathcal{K}(C)^2.
\end{aligned}$$

■

Proof of Lemma 4.5.8. First suppose $n = 0 = l - 2$. Then either $D = \varnothing$ or $D = \{11\}$. Since $k \geq 3$, we have

$2l-k \leq 1$. If $D=\varnothing$, then clearly $E[\mathcal{K}\left(DA^{k-l}\cap A^{k-l}D\right)]=0$, and so the lemma holds. If $D=\{11\}$ and $2l-k=1$, then

$$E[\mathcal{K}\left(DA^{k-l}\cap A^{k-l}D\right)]=E[\mathcal{K}\left(111\right)]=\frac{1}{8}=2\cdot\frac{1}{16}=2\mathcal{K}\left(D\right)^2,$$

and if $2l-k<1$, then

$$E[\mathcal{K}\left(DA^{k-l}\cap A^{k-l}D\right)]=E[\mathcal{K}\left(11A^{k-2l}11\right)]=\frac{1}{16}=\mathcal{K}\left(D\right)^2,$$

by Lemma 4.5.2. Thus the lemma holds when $n=0=l-2$.

Now suppose $n>0$ or $l>2$, so that the set $1A^{l-2}1A^n$ from which we choose D has at least 2 elements. We consider three cases, depending on the value of $2l-k$. In each of the cases, we will define a particular pattern $X\subseteq\{0,1,A\}^{k+n}$ such that the randomly created set $DA^{k-l}\cap A^{k-l}D$ is a subset of the deterministic set X. For each such case, let $G_1=R_{n+l}(X)$ and $G_2=X-G_1$. Then $|G_2|=|X|-|G_1|$, since $G_1\subseteq X$. Note that a word of G_1 is in $DA^{k-l}\cap A^{k-l}D$ if and only if the common $(n+l)$-bit prefix and suffix is in D, and a word of G_2 is in $DA^{k-l}\cap A^{k-l}D$ if and only if the distinct $(n+l)$-bit prefix and suffix are in D. Therefore,

$$\begin{aligned}
&E[\mathcal{K}\left(DA^{k-l}\cap A^{k-l}D\right)]\\
&=E[\mathcal{K}\left(DA^{k-l}\cap X\cap A^{k-l}D\right)]\\
&=E\left[\sum_{u\in X}1_{DA^{k-l}\cap A^{k-l}D}\left(u\right)\cdot 2^{-(n+k)}\right]\\
&=\frac{1}{2^{n+k}}\left(\sum_{u\in G_1}P\{u\in DA^{k-l}\cap A^{k-l}D\}+\sum_{u\in G_2}P\{u\in DA^{k-l}\cap A^{k-l}D\}\right)\\
&=\frac{1}{2^{n+k}}\left(|G_1|\cdot\frac{|D|}{2^{n+l-2}}+|G_2|\cdot\frac{|D|(|D|-1)}{2^{n+l-2}(2^{n+l-2}-1)}\right) &(4.58)\\
&=\frac{|D|}{2^{2n+k+l-2}}\left(|G_1|+(|XA^n|-|G_1|)\cdot\frac{|D|-1}{2^{n+l-2}-1}\right), &(4.59)
\end{aligned}$$

where (4.58) follows from Lemma 4.5.6.

- Case 1: $2l-k<1$.

  Let $X=1A^{l-2}1A^{k-2l}1A^{l-2}1A^n$. By Lemma 4.5.3, $|G_1|=2^{k-l-2}$. Therefore, from (4.59),

$$\begin{aligned}
E[\mathcal{K}\left(DA^{k-l}\cap A^{k-l}D\right)]&=\frac{|D|}{2^{2n+k+l-2}}\left(2^{k-l-2}+\frac{(|D|-1)\left(2^{k+n-4}-2^{k-l-2}\right)}{2^{n+l-2}-1}\right)\\
&=\left(\frac{|D|}{2^{n+l}}\right)^2=\mathcal{K}\left(D\right)^2.
\end{aligned}$$

- Case 2: $2l - k = 1$.

  Let $X = 1A^{l-2}1A^{l-2}1A^n$. By Lemma 4.5.3, $|G_1| = 2^{l-2}$. Therefore, from (4.59),

$$E[\mathcal{K}\left(DA^{k-l} \cap A^{k-l}D\right)] = \frac{|D|}{2^{2n+k+l-2}}\left(2^{l-2} + \frac{(|D|-1)\left(2^{2l-4+n} - 2^{l-2}\right)}{2^{n+l-2}-1}\right)$$
$$= 2\left(\frac{|D|}{2^{n+l}}\right)^2 = 2\mathcal{K}\left(D\right)^2.$$

- Case 3: $2l - k > 1$.

  Let $a = k - l - 1$, $b = 2l - k - 2$, and $X = 1A^a1A^b1A^a1A^n$. By Lemma 4.5.3, $|G_1| = \beta 2^a$, where

$$\beta = \begin{cases} 1 & \text{if } (a+1) \mid (b+1) \\ 1/2 & \text{if } (a+1) \nmid (b+1). \end{cases}$$

  Therefore, from (4.59),

$$\begin{aligned} &E[\mathcal{K}\left(DA^{k-l} \cap A^{k-l}D\right)] \\ &= \frac{|D|}{2^{2n+k+l-2}}\left(|G_1| + (2^{k+n-4} - |G_1|) \cdot \frac{|D|-1}{2^{n+l-2}-1}\right) \\ &= \left(\frac{|D|}{2^{n+l}}\right)^2 + \left(\frac{|D|}{2^{n+l}}\right)\frac{(2\beta-1)(\frac{1}{4} - |D|2^{-l-n})}{2^{n+l-2}-1} \\ &= \mathcal{K}\left(D\right)^2 + \frac{\mathcal{K}\left(D\right)(\frac{1}{4} - \mathcal{K}\left(D\right))(2\beta-1)}{2^{n+l-2}-1}. \end{aligned}$$

■

Proof of Lemma 4.5.10. First suppose D is a set of a fixed size chosen uniformly at random from $1A^{l-2}C$. Then given a word is in $1A^{l-2}CA^{k-l} \cap g(C)$, the probability that that word is in DA^{k-l} is the probability that the $(n+l)$-bit prefix is in D, which is $|D|/|1A^{l-2}C| = \mathcal{K}\left(D\right)/(\mathcal{K}\left(C\right)/2)$. Therefore, letting $X = 1A^{l-2}CA^{k-l} \cap g(C)$

(and noting that $DA^{k-l} \cap g(C) = DA^{k-l} \cap X$),

$$\begin{aligned}
E[\mathcal{K}\left(DA^{k-l} \cap g(C)\right)] &= \frac{1}{2^{n+k}} E\left[\sum_{u \in A^{n+k}} 1_{DA^{k-l} \cap X}(u)\right] \\
&= \frac{1}{2^{n+k}} \sum_{u \in A^{n+k}} E\left[1_{DA^{k-l} \cap X}(u)\right] \\
&= \frac{1}{2^{n+k}} \sum_{u \in A^{n+k}} P(u \in DA^{k-l} \cap X) \\
&= \frac{1}{2^{n+k}} \sum_{u \in A^{n+k}} P(u \in DA^{k-l} \mid u \in X) P(u \in X) \\
&= \frac{\mathcal{K}(D)}{\mathcal{K}(C)/2} \cdot \frac{1}{2^{n+k}} \sum_{u \in A^{n+k}} P(u \in X) \\
&= \frac{\mathcal{K}(D)}{\mathcal{K}(C)/2} E[\mathcal{K}\left(1A^{l-2}CA^{k-l} \cap g(C)\right)].
\end{aligned}$$

The other cases follow similarly. ■

Proof of Lemma 4.5.11. For any C as in the Lemma statement, we have

$$\begin{aligned}
\mathcal{K}\left(CA^{k-l} \cap A^{k-l}C\right) = \mathcal{K}\left(C_1A^{k-l} \cap A^{k-l}C_1\right) + \mathcal{K}\left(C_1A^{k-l} \cap A^{k-l}C_0\right) \\
- \mathcal{K}\left(C_0A^{k-l} \cap A^{k-l}C_1\right) + \mathcal{K}\left(C_0A^{k-l} \cap A^{k-l}C_0\right).
\end{aligned}$$

Using Lemma 4.5.4,

$$\mathcal{K}\left(C_1A^{k-l} \cap A^{k-l}C_1\right) = \begin{cases} 0 & \text{if } k = 2l-1 \\ \frac{1}{16} & \text{otherwise} \end{cases}$$

$$E[\mathcal{K}\left(C_0A^{k-l} \cap A^{k-l}C_1\right)] = \begin{cases} \frac{\mathcal{K}(C_0)}{2} & \text{if } k = 2l-1 \\ \frac{\mathcal{K}(C_0)}{4} & \text{otherwise} \end{cases}$$

$$E[\mathcal{K}\left(C_1A^{k-l} \cap A^{k-l}C_0\right)] = \begin{cases} 0 & \text{if } k = 2l-1 \\ \frac{\mathcal{K}(C_0)}{4} & \text{otherwise,} \end{cases}$$

and by Corollary 4.5.9,

$$E[\mathcal{K}\left(C_0A^{k-l} \cap A^{k-l}C_0\right)]$$
$$= \begin{cases} \mathcal{K}\left(C_0\right)^2 & \text{if } 2l-k<1 \\ 2\mathcal{K}\left(C_0\right)^2 & \text{if } 2l-k=1 \\ \mathcal{K}\left(C_0\right)^2 & \text{if } 2l-k>1 \text{ and } (k-l) \nmid (2l-k-1) \\ \mathcal{K}\left(C_0\right)^2 + \frac{\mathcal{K}(C_0)(\frac{1}{4}-\mathcal{K}(C_0))}{2^{n-1}-1} & \text{if } 2l-k>1 \text{ and } (k-l) \mid (2l-k-1). \end{cases}$$

Thus

$$E[\mathcal{K}\left(CA^{k-l} \cap A^{k-l}C\right)] = \begin{cases} (\mathcal{K}\left(C_0\right)+\frac{1}{4})^2 & \text{if } 2l-k<1 \\ \mathcal{K}\left(C_0\right)(2\mathcal{K}\left(C_0\right)+\frac{1}{2}) & \text{if } 2l-k=1 \\ (\mathcal{K}\left(C_0\right)+\frac{1}{4})^2 & \text{if } 2l-k>1 \text{ and } (k-l) \nmid (2l-k-1) \\ (\mathcal{K}\left(C_0\right)+\frac{1}{4})^2 + \frac{\mathcal{K}(C_0)(\frac{1}{4}-\mathcal{K}(C_0))}{2^{n-1}-1} & \text{if } 2l-k>1 \text{ and } (k-l) \mid (2l-k-1) \end{cases}$$
$$= \begin{cases} \mathcal{K}\left(C\right)^2 & \text{if } 2l-k<1 \\ 2\mathcal{K}\left(C\right)(\mathcal{K}\left(C\right)-\frac{1}{4}) & \text{if } 2l-k=1 \\ \mathcal{K}\left(C\right)^2 & \text{if } 2l-k>1 \text{ and } (k-l) \nmid (2l-k-1) \\ \mathcal{K}\left(C\right)^2 + \frac{(\mathcal{K}(C)-\frac{1}{4})(\frac{1}{2}-\mathcal{K}(C))}{2^{n-1}-1} & \text{if } 2l-k>1 \text{ and } (k-l) \mid (2l-k-1). \end{cases}$$

The lemma now follows using Lemma 4.5.10 since

$$\begin{aligned} E[\mathcal{K}\left(DA^{k-l} \cap A^{k-1}C\right)] &= E[\mathcal{K}\left(DA^{k-l} \cap 1A^{l-2}CA^{k-l} \cap A^{k-1}C\right)] \\ &= \frac{\mathcal{K}\left(D\right)}{\mathcal{K}\left(C\right)/2} \cdot E[\mathcal{K}\left(1A^{l-2}CA^{k-l} \cap A^{k-1}C\right)] \\ &= \frac{\mathcal{K}\left(D\right)}{\mathcal{K}\left(C\right)} \cdot E[\mathcal{K}\left(CA^{k-l} \cap A^{k-l}C\right)] \qquad (4.60) \end{aligned}$$

where (4.60) follows by Lemma 4.5.2. ■

Proof of Lemma 4.5.12. First suppose $n \leq a$. Then

$$\begin{aligned}
&E[\mathcal{K}\left(1A^aCA^{a+b+2} \cap 1A^a0A^b0A^a1A^n \cap A^{a+b+2}CA^{a+1}\right)] \\
&= E[\mathcal{K}\left(1A^a(CA^{b+1} \cap 0A^b0A^n \cap A^{b+1}C)A^{a-n}1A^n\right)] \\
&= \mathcal{K}\left(1A^a\right) E[\mathcal{K}\left(CA^{b+1} \cap 0A^b0A^n \cap A^{b+1}C\right)]\mathcal{K}\left(A^{a-n}1A^n\right) && (4.61) \\
&= \frac{1}{2} \cdot \mathcal{K}\left(C\right)^2 \mathcal{K}\left(A^b\right) \cdot \frac{1}{2}\cdot && (4.62) \\
&= \frac{\mathcal{K}\left(C\right)^2}{4} && (4.63)
\end{aligned}$$

where (4.61) follows from Lemma 4.5.2; (4.62) follows from Lemma 4.5.2 and Lemma 4.5.7; and (4.63) follows from Lemma 4.5.2.

Now suppose $n > a$. By Lemma 4.5.3,

$$|R_{n+1}(0A^b0A^a1A^{n-(a+1)})| = \begin{cases} 2^{b-1} & \text{if } (b+1) \nmid (a+1) \\ 0 & \text{otherwise} \end{cases}. \tag{4.64}$$

Let $X = R_{n+1}(0A^b0A^a1A^{n-(a+1)})$. Then if $(b+1) \nmid (a+1)$, the expected Kraft sum is

$$\begin{aligned}
&E\left[\mathcal{K}\left(1A^aCA^{a+b+2} \cap 1A^a0A^b0A^a1A^n \cap A^{a+b+2}CA^{a+1}\right)\right] \\
&= E\left[\mathcal{K}\left(1A^a(CA^{b+1} \cap 0A^b0A^a1A^{n-(a+1)} \cap A^{b+1}C)A^{a+1}\right)\right] && (4.65) \\
&= \mathcal{K}\left(1A^a\right) E\left[\mathcal{K}\left(CA^{b+1} \cap 0A^b0A^a1A^{n-(a+1)} \cap A^{b+1}C\right)\right] \mathcal{K}\left(A^{a+1}\right) \\
&= \frac{1}{2}E\left[\mathcal{K}\left(CA^{b+1} \cap 0A^b0A^a1A^{n-(a+1)} \cap A^{b+1}C\right)\right] && (4.66) \\
&= \frac{1}{2}\left(\frac{|X|}{2^{n+b+2}} \cdot \frac{|C|}{2^n} + \left(\frac{1}{8} - \frac{|X|}{2^{n+b+2}}\right)\frac{|C|(|C|-1)}{2^n(2^n-1)}\right) && (4.67) \\
&= \frac{1}{2^{n+4}}\left(\frac{|C|}{2^n} + \frac{2^n-1}{2^n} \cdot \frac{|C|(|C|-1)}{2^n-1}\right) && (4.68) \\
&= \frac{|C|^2}{2^{n+4}} \\
&= \frac{\mathcal{K}\left(C\right)^2}{4} && (4.69)
\end{aligned}$$

where (4.66) follows from Lemma 4.5.2; (4.67) follows from the fact that $\mathcal{K}\left(0A^b0A^a1A^{n-(a+1)}\right) = 1/8$ (by Lemma 4.5.2) and from Lemma 4.5.6; (4.68) follows from (4.64).

On the other hand, if $(b+1) \mid (a+1)$, then following the same Kraft sum calculation as in (4.65)-(4.67)

gives

$$
\begin{aligned}
&E[\mathcal{K}\left(1A^aCA^{a+b+2} \cap 1A^a0A^b0A^a1A^n \cap A^{a+b+2}CA^{a+1}\right)] \\
&= \frac{1}{2}\left(\frac{0}{2^{n+b+2}} \cdot \frac{|C|}{2^n} + \left(\frac{1}{8} - \frac{0}{2^{n+b+2}}\right)\frac{|C|(|C|-1)}{2^n(2^n-1)}\right) && (4.70)\\
&= \frac{\mathcal{K}(C)^2}{4} - \frac{1}{2^{n+4}} \cdot \frac{|C|}{2^n} + \frac{1}{2^{n+4}} \cdot \frac{|C|(|C|-1)}{2^n(2^n-1)} && (4.71)\\
&= \frac{\mathcal{K}(C)^2}{4} - \frac{|C|}{2^{2n+4}}\left(1 - \frac{|C|-1}{2^n-1}\right) \\
&= \frac{\mathcal{K}(C)^2}{4} - \frac{\mathcal{K}(C)}{2^{n+3}} \cdot \frac{2^n-|C|}{2^n-1} \\
&= \frac{\mathcal{K}(C)^2}{4} - \frac{\mathcal{K}(C)\left(\frac{1}{2} - \mathcal{K}(C)\right)}{4(2^n-1)}
\end{aligned}
$$

where (4.70) follows from (4.64); and (4.71) follows from (4.68) and (4.69). ■

Proof of Lemma 4.5.13. Let $X = 0A^a0A^b0A^a1A^n$. If $n \leq b$, then using Lemma 4.5.3,

$$
\begin{aligned}
|R_{n+1}(X)| &= |R_{n+1}(0A^a0A^n)A^{b-n}0A^a1A^n| \\
&= |R_{n+1}(0A^a0A^n)| \cdot |A^{b-n}0A^a1A^n| \\
&= 2^a \cdot 2^{a+b} = 2^{2a+b}.
\end{aligned}
$$

If $b < n \leq a+b+1$, then using Lemma 4.5.3,

$$
\begin{aligned}
|R_{n+1}(X)| &= |R_{n+1}(0A^a0A^b0A^{n-(a+b+1)})A^{a+b+1-n}1A^n| \\
&= |R_{n+1}(0A^a0A^b0A^{n-(a+b+1)})| \cdot |A^{a+b+1-n}1A^n| \\
&= \begin{cases} 2^{2a+b+1} & \text{if } (a+1) \mid (b+1) \\ 2^{2a+b} & \text{otherwise.} \end{cases}
\end{aligned}
$$

If $n > a+b+1$, then

$$
\begin{aligned}
|R_{n+1}(X)| &= |R_{n+1}(0A^a0A^b0A^a1A^{n-(a+b+2)})A^{a+b+2}| \\
&= |R_{n+1}(0A^a0A^b0A^a1A^{n-(a+b+2)})| \cdot |A^{a+b+2}| \\
&= 0,
\end{aligned}
$$

using Lemma 4.5.3, since $X_{a+b+3} = 0$, $X_{2a+b+4} = 1$, and $a+b+3 \equiv (2a+b+4) \bmod (a+1)$.

Then using a similar probability calculation as in the proof of Lemma 4.5.7, when $|R_{n+1}(X)| = 2^{2a+b}$ we have

$$\begin{aligned} E[\mathcal{K}(X)] &= E[\mathcal{K}(R_{n+1}(X))] \cdot \frac{|C|}{2^n} + E[\mathcal{K}\left(A^{2a+b+n} - R_{n+1}(X)\right)] \cdot \frac{|C|(|C|-1)}{2^n(2^n-1)} \\ &= \frac{2^{2a+b}}{2^{2a+b+n+4}} \cdot \frac{|C|}{2^n} + \left(\frac{2^{2a+b+n}}{2^{2a+b+n+4}} - \frac{2^{2a+b}}{2^{2a+b+n+4}}\right) \cdot \frac{|C|(|C|-1)}{2^n(2^n-1)} \\ &= \frac{1}{4} \cdot \frac{|C|^2}{2^{2(n+1)}} = \frac{\mathcal{K}(C)^2}{4}. \end{aligned}$$

Otherwise, when $|R_{n+1}(X)| = 2^{2a+b} + \beta 2^{2a+b}$ for $\beta \in \{-1, 1\}$, we have, using the previous calculation,

$$\begin{aligned} E[\mathcal{K}(X)] &= \frac{\mathcal{K}(C)^2}{4} + \beta \frac{2^{2a+b}}{2^{2a+b+n+4}} \left(\frac{|C|}{2^n} - \frac{|C|(|C|-1)}{2^n(2^n-1)}\right) \\ &= \frac{\mathcal{K}(C)^2}{4} + \beta \frac{1}{4(2^n-1)} \mathcal{K}(C) \left(\frac{1}{2} - \mathcal{K}(C)\right). \end{aligned}$$

■

Proof of Lemma 4.5.14. Let

$$\begin{aligned} G_1 &= 1A^a1A^bR_{n+1}(0A^a0A^n) \\ G_2 &= 1A^a1A^b0A^a0A^n - G_1 \\ H_1 &= R_{n+a+b+3}(1A^a1A^b0A^a0A^n) \\ H_2 &= 1A^a1A^b0A^a0A^n - H_1. \end{aligned}$$

Then $H_1 \subseteq G_1$ and $G_2 \subseteq H_2$. Then Lemma 4.5.3 implies

$$|G_1 \cap H_1| = |H_1| = \begin{cases} 2^{a-1} & \text{if } (a+1) \nmid (b+1) \\ 0 & \text{otherwise} \end{cases}$$

$$|G_1 \cap H_2| = |G_1 - H_1| = \begin{cases} 2^{2a+b} - 2^{a-1} & \text{if } (a+1) \nmid (b+1) \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} |G_2 \cap H_2| = |G_2| &= 2^{2a+b+n} - |G_1| \\ &= 2^{2a+b+n} - 2^{a+b}2^a \\ &= 2^{2a+b}(2^n - 1). \end{aligned}$$

Let $S = DA^{a+1} \cap 1A^a1A^b0A^a0A^n \cap A^{a+1}D$. If C is chosen uniformly at random from $0A^n$, and D is chosen uniformly at random from $1A^{a+b+1}C \subseteq 1A^{a+b+1}0A^n$, then for any word of length $n+l+a+1$, the probability it lies in $S \cap G_1 \cap H_1$ is

$$\frac{|C|}{2^n} \cdot \frac{|D|}{|C| \cdot 2^{a+b+1}},$$

the probability it lies in $S \cap G_1 \cap H_2$ is

$$\frac{|C|}{2^n} \cdot \frac{|D| \cdot (|D|-1)}{|C| \cdot 2^{a+b+1} \cdot (|C| \cdot 2^{a+b+1}-1)},$$

and the probability it lies in $S \cap G_2 \cap H_2$ is

$$\frac{|C| \cdot (|C|-1)}{2^n(2^n-1)} \cdot \frac{|D| \cdot (|D|-1)}{|C| \cdot 2^{a+b+1} \cdot (|C| \cdot 2^{a+b+1}-1)}.$$

Therefore, if $(a+1) \nmid (b+1)$, then

$$\begin{aligned}
E[\mathcal{K}(S)] &= E[\mathcal{K}(S \cap G_1 \cap H_1)] + E[\mathcal{K}(S \cap G_1 \cap H_2)] + E[\mathcal{K}(S \cap G_2 \cap H_2)] \\
&= \frac{2^{a-1}}{2^{n+2a+b+4}} \cdot \frac{|C|}{2^n} \cdot \frac{|D|}{|C| \cdot 2^{a+b+1}} \\
&\quad + \frac{2^{2a+b}-2^{a-1}}{2^{n+2a+b+4}} \cdot \frac{|C|}{2^n} \cdot \frac{|D| \cdot (|D|-1)}{|C| \cdot 2^{a+b+1}(|C| \cdot 2^{a+b+1}-1)} \\
&\quad + \frac{2^{2a+b}(2^n-1)}{2^{n+2a+b+4}} \cdot \frac{|C| \cdot (|C|-1)}{2^n(2^n-1)} \cdot \frac{|D| \cdot (|D|-1)}{|C| \cdot 2^{a+b+1}(|C| \cdot 2^{a+b+1}-1)} \\
&= \mathcal{K}(D)^2 \qquad (4.72)
\end{aligned}$$

where (4.72) follows from $\mathcal{K}(D) = \frac{|D|}{2^{n+a+b+3}}$.

On the other hand, if $(a+1) \mid (b+1)$, then

$$\begin{aligned}
&E[\mathcal{K}(S)] \\
&= \mathcal{K}(D)^2 - \frac{2^{a-1}}{2^{n+2a+b+4}} \cdot \frac{|C|}{2^n} \cdot \frac{|D|}{|C| \cdot 2^{a+b+1}} + \frac{2^{a-1}}{2^{n+2a+b+4}} \cdot \frac{|C|}{2^n} \cdot \frac{|D| \cdot (|D|-1)}{|C| \cdot 2^{a+b+1}(|C| \cdot 2^{a+b+1}-1)} \\
&= \mathcal{K}(D)^2 - \frac{\mathcal{K}(D)}{|C| \cdot 2^{a+b+1}-1} \left(\frac{\mathcal{K}(C)}{2} - \mathcal{K}(D) \right).
\end{aligned}$$

■

Proof of Lemma 4.5.15. Let $r = 2^n$ and $s = 2^l$. Then

$$f(x,y) = y^2\left(\frac{xrs}{xrs-2}\right) - y\left(\frac{1}{2} - \frac{\frac{1}{2}-x}{2(r-1)} + \frac{x}{xrs-2}\right) + \frac{1}{16} + \frac{x(\frac{1}{2}-x)}{2(r-1)}.$$

Since

$$f\left(x,\frac{x}{2}\right) = \frac{r}{4(r-1)}\left(x-\frac{1}{2}\right)\left(x-\frac{1}{2}+\frac{1}{2r}\right) \geq 0 \qquad \forall x \leq \frac{1}{2}-\frac{1}{2r}$$
$$f\left(\frac{1}{2},y\right) = \left(\frac{rs}{rs-4}\right)\left(y-\frac{1}{4}\right)\left(y-\frac{1}{4}+\frac{1}{rs}\right) \geq 0 \qquad \forall y \leq \frac{1}{4}-\frac{1}{rs}$$

and $f(1/2,1/4) = 0$, the lemma holds when $y = x/2$ and also when $x = 1/2$. Note that

$$f\left(x,\frac{x}{2}-\frac{1}{rs}\right) = \frac{1}{4}\left(\frac{1}{2}-x\right)\left(\frac{1}{2}-x-\frac{x}{r-1}\right) + \frac{\frac{1}{2}-x}{rs}\left(1-\frac{1}{2(r-1)}\right)$$

which is 0 when $x = 1/2$, and when $x \leq \frac{1}{2} - \frac{1}{2r}$, satisfies

$$f\left(x,\frac{x}{2}-\frac{1}{rs}\right) \geq \frac{1}{4}\cdot\frac{1}{2r}\left(\frac{1}{2r}-\frac{1}{2r}\right) + \frac{1}{2r^2 s}\left(1-\frac{1}{2}\right) > 0.$$

Thus $f(x,y) \geq 0$ when $y = \frac{x}{2} - \frac{1}{rs}$ and $x \in [0, \frac{1}{2} - \frac{1}{2r}] \cup \{\frac{1}{2}\}$, i.e., the lemma holds when $y = \frac{x}{2} - \frac{1}{rs}$.

For all $x \in [\frac{1}{2r}, \frac{1}{2} - \frac{1}{2r}]$, since $xrs \geq (1/2r)rs \geq 4$, we have

$$\left.\frac{\partial f}{\partial y}\right|_{y=\frac{x}{2}-\frac{1}{rs}} = -\left(\frac{1}{2}-x\right)\left(\frac{2r-3}{r-1}\right) - \frac{2x}{xrs-2} < 0.$$

Thus, for any fixed $x \in [0, \frac{1}{2} - \frac{1}{2^{n+1}}]$, the function f is a convex parabola in y, which at $y = \frac{x}{2} - \frac{1}{rs}$ is both non-negative and has a negative slope, and is therefore non-negative for all $y \leq \frac{x}{2} - \frac{1}{rs}$. ■

Proof of Lemma 4.6.1. For cases (i)–(ix), we will assume $2l - k \neq 1$. For these cases, the set

$$Z_1 A^{l-2} Z_2 A^{n+k-l} \cap A^{k-l} Z_3 A^{l-2} Z_4 A^n$$

has bits Z_2 and Z_3 in different positions, so it is either the pattern $Z_1 A^{l-2} Z_2 A^{k-2l} Z_3 A^{l-2} Z_4 A^n$ (when $2l-k<1$) or the pattern $Z_1 A^{k-l-1} Z_3 A^{2l-k-2} Z_2 A^{k-l-1} Z_4 A^n$ (when $2l-k>1$), which in both cases has exactly four fixed bits.

(i) The set $1A^{l-2}0A^{n+k-l} \cap A^{k-l}0A^{l-2}1A^n$ is a pattern with exactly four fixed bits, and thus, by Lemma 4.5.2, its Kraft sum is $1/16$.

(ii),(iii) The set $CA^{k-1} \cap 0A^{l-2}b_1A^{n+k-l} \cap A^{k-l}0A^{l-2}1A^n$ equals $CA^{k-1} \cap 0U$ (where $U \in \{0,1,A\}^{n+k-1}$ is a pattern with exactly three fixed bits), and thus, by Corollary 4.5.5, its expected Kraft sum is $\mathcal{K}(C)/8$. Similar reasoning proves case (iii).

(iv) The set $CA^{k-1} \cap 0A^{l-2}b_1A^{n+k-l} \cap A^{k-l}b_2A^{l-2}C$ equals $CA^{k-1} \cap 0U0A^n \cap A^{k-1}C$ (where $U \in \{0,1,A\}^{k-2}$ is a pattern with exactly two fixed bits) and thus, by Lemma 4.5.7, its expected Kraft sum is $\mathcal{K}(C)^2/4$.

(v),(vi) The set $DA^{k-l} \cap A^{k-l}0A^{l-2}1A^n$ equals $DA^{k-l} \cap U$, where $U \in \{0,1,A\}^{n+k}$ is either the pattern $1A^{l-2}1A^{k-2l}0A^{l-2}1A^n$ (when $2l-k<1$) or the pattern $1A^{k-l-1}0A^{2l-k-2}1A^{k-l-1}1A^n$ (when $2l-k>1$), both of which have exactly four fixed bits. Thus, by Corollary 4.5.5, the set's expected Kraft sum is $\mathcal{K}(D)/4$. Similar reasoning proves case (vi).

(vii),(viii) The set $CA^{k-1} \cap 0A^{l-2}b_1A^{n+k-l} \cap A^{k-l}D$, by Lemma 4.5.4, has expected Kraft sum is $\mathcal{K}(C)\mathcal{K}(D)/2$. Similar reasoning proves case (viii).

(ix) This case follows immediately from Lemma 4.5.8.

For cases (x)–(xvi), we will assume $2l-k=1$. For these cases, the set

$$Z_1A^{l-2}Z_2A^{n+k-l} \cap A^{k-l}Z_3A^{l-2}Z_4A^n$$

is empty if $Z_2 \neq Z_3$, and otherwise is a pattern with exactly three fixed bits.

(x) The set $1A^{l-2}0A^{n+k-l} \cap A^{k-l}0A^{l-2}1A^n$ is a pattern with exactly three fixed bits, and thus, by Lemma 4.5.2. its Kraft sum is $1/8$.

(xi),(xii) The set $CA^{k-1} \cap 0A^{l-2}0A^{n+k-l} \cap A^{k-l}0A^{l-2}1A^n$ equals $CA^{k-1} \cap 0U$, (where $U \in \{0,1,A\}^{n+k-1}$ is a pattern with exactly two fixed bits), and thus, by Corollary 4.5.5, its expected Kraft sum is $\mathcal{K}(C)/4$. Similar reasoning proves case (xii).

(xiii) The set $CA^{k-1} \cap 0A^{l-2}b_1A^{n+k-l} \cap A^{k-l}b_1A^{l-2}C$ equals $CA^{k-1} \cap 0U0A^n \cap A^{k-1}C$ (where $U \in \{0,1,A\}^{k-2}$ is a pattern with exactly one fixed bit), and thus, by Lemma 4.5.7, its expected Kraft sum is $\mathcal{K}(C)^2/2$.

(xiv),(xv) The set $CA^{k-1} \cap 0A^{l-2}1A^{n+k-l} \cap A^{k-l}D$ equals $CA^{k-1} \cap A^{k-l}D$, and thus, by Lemma 4.5.4, its expected Kraft sum is $\mathcal{K}(C)\mathcal{K}(D)$. Similar reasoning proves case (xv).

(xvi) This case follows directly from Lemma 4.5.8.

■

Proof of Lemma 4.7.1. The proof is similar to that of Lemma 4.6.1. For cases (i)–(iii), we will assume $2l - k \neq 1$.

(i) The set $1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}0A^{l-2}1A^n$ equals the set $A^{l-1}(0A^n - C)A^{k-l} \cap U$ (where $U \in \{0, 1, A\}^{n+k}$ is a pattern with exactly four fixed bits), and thus, by Corollary 4.5.5, its expected Kraft sum is $\mathcal{K}(0A^n - C)/8 = (\frac{1}{2} - \mathcal{K}(C))/8$, since $0A^n - C$ is chosen uniformly at random from $0A^n$ (by Lemma 4.5.2).

(ii) The expected Kraft sum of the set

$$\begin{aligned} &1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}0A^{l-2}C \\ &= (1A^{l-2}0A^{n+k-l} \cap A^{k-l}0A^{l-2}C) - (1A^{l-2}CA^{k-l} \cap A^{k-l}0A^{l-2}C) \end{aligned}$$

is $\frac{1}{8}\mathcal{K}(C) - \frac{1}{4}\mathcal{K}(C)^2 = \frac{1}{4}\mathcal{K}(C)(\frac{1}{2} - \mathcal{K}(C))$, since the expected Kraft sums of its two parts are $\frac{1}{8}\mathcal{K}(C)$ (by Lemma 4.6.1) and $\frac{1}{4}\mathcal{K}(C)^2$ (by Lemma 4.5.7).

(iii) The sets C and D_1 are chosen independently of each other, and the locations of the fixed bits of the sets from which they are drawn do not overlap (since $2l - 1 \neq k$). Therefore, the expected Kraft sum of $1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}D_1$, by Lemma 4.5.4, is $(1/2)(\frac{1}{2} - \mathcal{K}(C))\mathcal{K}(D_1)$, since the probability that it contains any particular word of length $n + k$ is the product of the probabilities that the word lies in each of the two intersected sets.

For cases (iv)–(ix), we will assume $2l - k = 1$.

(iv) The set $1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}0A^{l-2}1A^n$ equals the set $A^{l-1}(0A^n - C)A^{k-l} \cap U$ (where $U \in \{0, 1, A\}^{n+k}$ is a pattern with exactly three fixed bits), and thus, by Corollary 4.5.5, its expected Kraft sum is $\frac{1}{4}\mathcal{K}(0A^n - C) = \frac{1}{4}(\frac{1}{2} - \mathcal{K}(C))$, since $0A^n - C$ is chosen uniformly at random from $0A^n$.

(v) By Lemma 4.6.1,

$$E[\mathcal{K}\left(CA^{k-1} \cap 0A^{l-2}1A^{n+k-l} \cap A^{k-l}1A^{l-2}C\right)] = \frac{\mathcal{K}(C)^2}{2},$$

and so by Lemma 4.5.10, using $g(C) = CA^{k-1} \cap 0A^{l-2}1A^{n+k-l}$ and $A^{k-l}D_2 \subseteq A^{k-l}1A^{l-2}C$, we get

$$E[\mathcal{K}\left(CA^{k-1} \cap 0A^{l-2}1A^{n+k-l} \cap A^{k-l}D_2\right)] = \frac{\mathcal{K}(C)^2}{2} \cdot \frac{\mathcal{K}(D_2)}{\mathcal{K}(C)/2} = \mathcal{K}(C)\mathcal{K}(D_2).$$

(vi) The expected Kraft sum of the set

$$1A^{l-2}(0A^n - C)A^{k-l} \cap A^{k-l}0A^{l-2}C$$
$$= 1A^{l-2}0A^{n+k-l} \cap A^{k-l}0A^{l-2}C - 1A^{l-2}CA^{k-l} \cap A^{k-l}0A^{l-2}C.$$

is $\frac{1}{4}\mathcal{K}(C) - \frac{1}{2}\mathcal{K}(C)^2 = \frac{1}{2}\mathcal{K}(C)(\frac{1}{2} - \mathcal{K}(C))$, since the expected Kraft sums of its two parts are $\mathcal{K}(C)/4$ (by Lemma 4.6.1) and $\mathcal{K}(C)^2/2$ (by Lemma 4.5.2 and Lemma 4.5.7).

(vii) Since $1A^{l-2}CA^{k-l} \cap A^{k-l}0A^{l-2}1A^n$ equals $A^{l-1}CA^{k-l} \cap U$ (where $U \in \{0,1,A\}^{n+k}$ is a pattern with exactly three fixed bits), by Corollary 4.5.5 its expected Kraft sum is $\mathcal{K}(C)/4$. Then by Lemma 4.5.10, $E[\mathcal{K}\left(D_2A^{k-l} \cap A^{k-l}0A^{l-2}1A^n\right)] = \frac{(\mathcal{K}(C)/4)\mathcal{K}(D_2)}{\mathcal{K}(C)/2} = \mathcal{K}(D_2)/2$.

(viii) By Lemma 4.5.2 and Lemma 4.5.7, we have

$$E[\mathcal{K}\left(1A^{l-2}CA^{k-l} \cap A^{k-l}0A^{l-2}C\right)] = \mathcal{K}\left(1A^{l-2}\right)E[\mathcal{K}\left(CA^{k-l} \cap A^{l-1}C\right)] = \mathcal{K}(C)^2/2$$

so by Lemma 4.5.10, the claimed expected Kraft sum is $\frac{(\mathcal{K}(C)^2/2)\mathcal{K}(D_2)}{\mathcal{K}(C)/2} = \mathcal{K}(C)\mathcal{K}(D_2)$.

■

Proof of Lemma 4.8.2. The proof is similar to that of Lemma 4.6.1. For cases (i)–(iii), we will assume $2l - k < 1$.

(i) The set

$$CA^{l-2-n}0A^{n+k-l} \cap A^{k-l}(0A^n - C)A^{l-2-n}1A^n$$
$$= CA^{l-2-n}0A^{k-2l}0A^{l-2}1A^n - CA^{l-2-n}0A^{k-2l}CA^{l-2-n}1A^n$$

has expected Kraft sum $\frac{1}{8}\mathcal{K}(C) - \frac{1}{4}\mathcal{K}(C)^2 = \mathcal{K}(C)(\frac{1}{2} - \mathcal{K}(C))/4$, since its first term has expected Kraft sum $\mathcal{K}(C)/8$ (by Corollary 4.5.5) and its second term has expected Kraft sum $\mathcal{K}(C)^2/4$ (by Lemma 4.5.2 and Lemma 4.5.7).

(ii) This case follows from Lemma 4.5.12, since $0A^n - C$ is chosen uniformly at random from $0A^n$, and since $n \leq l - 2$.

(iii) Since $0A^n - C$ and D_1 are chosen independently and the fixed bits of $1A^{l-2}1A^{n+k-l}$ and $A^{k-l}0A^{l-2}1A^n$

do not overlap, Lemma 4.5.4 implies the claimed expected Kraft sum is

$$\mathcal{K}\left(1A^{l-2}1A^{n+k-l}\right)\cdot\frac{\mathcal{K}\left(D_1\right)}{\mathcal{K}\left(1A^{l-2}1A^{n}\right)}\cdot\mathcal{K}\left(A^{k-l}0A^{l-2}1A^{n}\right)\cdot\frac{\mathcal{K}\left(0A^{n}-C\right)}{\mathcal{K}\left(0A^{n}\right)}=\frac{\mathcal{K}\left(D_1\right)\left(\frac{1}{2}-\mathcal{K}\left(C\right)\right)}{2}.$$

For cases (iv)–(v), we will assume $2l-k=1$.

(iv) The set

$$\begin{aligned}&CA^{l-2-n}0A^{n+k-l}\cap A^{k-l}(0A^{n}-C)A^{l-2-n}1A^{n}\\&=CA^{l-2-n}0A^{l-2}1A^{n}-CA^{l-2-n}CA^{l-2-n}1A^{n}\end{aligned}$$

has expected Kraft sum $\frac{1}{4}\mathcal{K}\left(C\right)-\frac{1}{2}\mathcal{K}\left(C\right)^{2}=\mathcal{K}\left(C\right)\left(\frac{1}{2}-\mathcal{K}\left(C\right)\right)/2$, where its first term has expected Kraft sum $\mathcal{K}\left(C\right)/4$ (by Lemma 4.6.1) and its second term has expected Kraft sum $\mathcal{K}\left(C\right)^{2}/2$ (by Lemma 4.5.2 and Lemma 4.5.7).

(v) The expected Kraft sum of

$$1A^{l-2}(0A^{n}-C)A^{k-l}\cap A^{k-l}(0A^{n}-C)A^{l-2-n}1A^{n}=1A^{l-2}(0A^{n}-C)A^{l-2-n}1A^{n}$$

is $(\frac{1}{2}-\mathcal{K}\left(C\right))/4$ by Lemma 4.5.2.

For cases (vi)–(xii), we will assume $2l-k>1$.

(vi) Since $\mathcal{K}\left(CA^{l-2-n}0A^{n+k-l}\cap A^{k-l}1A^{l-2}C\right)=\mathcal{K}\left(C\right)^{2}/4$ by Lemma 4.6.1, the desired expected Kraft sum is $\frac{(\mathcal{K}(C)^{2}/4)\mathcal{K}(D_2)}{\mathcal{K}(C)/2}=\mathcal{K}\left(C\right)\mathcal{K}\left(D_2\right)/2$.

(vii) By Lemma 4.5.10, the expected Kraft sum of the set

$$\begin{aligned}&1A^{l-2}(0A^{n}-C)A^{k-l}\cap A^{k-l}1A^{l-2}C\\&=1A^{k-l-1}1A^{2l-k-2}0A^{k-l-1}C-1A^{k-l-1}1A^{2l-k-2}(CA^{k-l}\cap 0A^{k-l-1}0A^{n}\cap Ak-lC)\end{aligned}$$

is

$$\frac{\mathcal{K}\left(D_2\right)}{\mathcal{K}\left(C\right)/2}\left(\frac{\mathcal{K}\left(C\right)}{8}-\frac{\mathcal{K}\left(C\right)^{2}}{4}\right)=\frac{(\frac{1}{2}-\mathcal{K}\left(C\right))\mathcal{K}\left(D_2\right)}{2}$$

since the expected Kraft sum of the first term is $\mathcal{K}\left(C\right)/8$ (by Lemma 4.5.2) and the expected Kraft sum of the second term is $\mathcal{K}\left(C\right)^{2}/4$ (by Lemma 4.5.2 and Lemma 4.5.7).

(viii) Lemma 4.5.2 and Lemma 4.5.7 imply

$$\begin{aligned}\mathcal{K}\left(1A^{l-2-n}CA^{k-l}\cap A^{k-l}0A^{l-2}C\right) &= \mathcal{K}\left(1A^{k-l-1}0A^{2l-k-2}(CA^{k-l}\cap 0A^{k-l-1}0A^{n}\cap A^{k-l}C)\right)\\ &= \mathcal{K}\left(C\right)^{2}/4\end{aligned}$$

so the claimed expected Kraft sum is $\frac{1}{4}\mathcal{K}\left(C\right)^{2}\frac{\mathcal{K}(D_2)}{\mathcal{K}(C)/2} = \mathcal{K}\left(C\right)\mathcal{K}\left(D_2\right)/2$.

(ix) We have

$$\begin{aligned}&\mathcal{K}\left(CA^{l-2-n}0A^{n+k-l}\cap A^{k-l}(0A^{n}-C)A^{l-2-n}1A^{n}\right)\\ &= \mathcal{K}\left(CA^{l-2-n}0A^{n+k-l}\cap A^{k-l}0A^{l-2}1A^{n}\right)\\ &\quad-\mathcal{K}\left(CA^{k-l}\cap 0A^{k-l-1}0A^{2l-k-2}0A^{k-l-1}1A^{n}\cap A^{k-l}C\right)\\ &= \frac{\mathcal{K}\left(C\right)}{8}-\frac{\mathcal{K}\left(C\right)^{2}}{4}\\ &\quad-\begin{cases}\frac{\mathcal{K}(C)(\frac{1}{2}-\mathcal{K}(C))}{4(2^{n}-1)} & \text{if } n\geq 2l-k-1 \text{ and } (k-l)\mid(2l-k-1)\\ 0 & \text{otherwise}\end{cases}\end{aligned}$$

by Lemma 4.5.2 and Lemma 4.5.13 (with $a = k-l-1$ and $b = 2l-k-2$).

(x) This case follows immediately by Lemma 4.5.12 (since $0A^{n}-C$ is drawn uniformly at random from $0A^{n}$), with $a = k-l-1$ and $b = 2l-k-2$.

(xi) We have

$$\begin{aligned}&\mathcal{K}\left(1A^{l-2}CA^{k-l}\cap A^{k-l}(0A^{n}-C)A^{l-2-n}1A^{n}\right)\\ &\qquad= \mathcal{K}\left(1A^{l-2}CA^{k-l}\cap A^{k-l}0A^{l-2}1A^{n}\right)\\ &\qquad\quad-\mathcal{K}\left(1A^{l-2}CA^{k-l}\cap A^{k-l}CA^{l-2-n}1A^{n}\right)\\ &\qquad= \frac{\mathcal{K}\left(C\right)}{8}-\frac{\mathcal{K}\left(C\right)^{2}}{4}\\ &\qquad\quad+\begin{cases}\frac{\mathcal{K}(C)(\frac{1}{2}-\mathcal{K}(C))}{4(2^{n}-1)} & \text{if } n\geq k-l \text{ and } (2l-k-1)\mid(k-l)\\ 0 & \text{otherwise}\end{cases},\end{aligned}$$

so by Lemma 4.5.10,

$$E[\mathcal{K}\left(D_2A^{k-l} \cap A^{k-l}(0A^n - C)A^{l-2-n}1A^n\right)]$$
$$= \frac{\mathcal{K}(D_2)}{\mathcal{K}(C)/2} \cdot \mathcal{K}\left(1A^{l-2}CA^{k-l} \cap A^{k-l}(0A^n - C)A^{l-2-n}1A^n\right)$$
$$= \frac{(\frac{1}{2} - \mathcal{K}(C))\mathcal{K}(D_2)}{2}$$
$$+ \begin{cases} \frac{\mathcal{K}(D_2)(\frac{1}{2} - \mathcal{K}(C))}{2(2^n-1)} & \text{if } n \geq k-l \text{ and } (2l-k-1) \mid (k-l) \\ 0 & \text{otherwise} \end{cases}.$$

(xii) This case follows directly from Lemma 4.5.14.

■

Proof of Lemma 4.8.3. The proof is similar to that of Lemma 4.6.1. For cases (i)–(iv), we will assume $2l - k \neq 1$.

(i) The set $1A^{l-2}0A^{n+k-l} \cap A^{k-l}bA^{l-2}C$ equals the set $U0A^n \cap A^{k-1}C$ (where $U \in \{0,1,A\}^{k-1}$ is a pattern with exactly three fixed bits), so by Lemma 4.5.2, its expected Kraft sum is $\mathcal{K}(C)/8$.

(ii) The claimed expected Kraft sum is $(1/4)\mathcal{K}(G) = (\frac{1}{4} - \mathcal{K}(C))/4$, by Lemma 4.5.4.

(iii) The claimed expected Kraft sum is $\mathcal{K}(D_1) \cdot (1/4)\mathcal{K}(C)/(1/2) = \mathcal{K}(C)\mathcal{K}(D_1)/2$, by Lemma 4.5.4.

(iv) The claimed expected Kraft sum is $\mathcal{K}(D_1)\mathcal{K}(G) = \mathcal{K}(D_1)(\frac{1}{4} - \mathcal{K}(C))$, by Lemma 4.5.4.

For cases (v)–(viii), we will assume $2l - k = 1$.

(v) The set $1A^{l-2}0A^{n+k-l} \cap A^{k-l}0A^{l-2}C$ equals the set $1A^{l-2}0A^{l-2}C$, which has expected Kraft sum $\mathcal{K}(C)/4$ by Lemma 4.5.2.

(vi) The set $1A^{l-2}0A^{n+k-l} \cap A^{k-l}GA^{l-2}$ equals the set $1A^{l-2}GA^{l-2}$, which has expected Kraft sum $\mathcal{K}(G)/2 = (\frac{1}{4} - \mathcal{K}(C))/2$ by Lemma 4.5.2.

(vii) We have

$$E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-l}CA^{l-1}\right)] = E[\mathcal{K}\left(1A^{l-2}CA^{l-1}\right)] = \mathcal{K}(C)/2$$

by Lemma 4.5.2, so the claimed expected Kraft sum is $(\mathcal{K}(D_2)/\mathcal{K}(C))(\mathcal{K}(C)/2) = \mathcal{K}(D_2)/2$, by Lemma 4.5.10.

(viii) We have

$$R_{n+1}(0A^{l-2}1A^{l-2}0A^{l-2}1A^{n-(l-1)}) = 2^{2(l-2)} = 2^{2l-4} = 2^{k-3}$$

by Lemma 4.5.3. Therefore, using Lemma 4.5.6, the expected Kraft sum of

$$CA^{k-1} \cap A^{k-l}1A^{l-2}C \subseteq 0A^{l-2}1A^{l-2}0A^{l-2}1A^{n-(l-1)}$$

is

$$\begin{aligned}\frac{2^{k-3}}{2^{n+k}} \cdot \frac{|C|}{2^{n-1}} + \left(\frac{1}{16} - \frac{2^{k-3}}{2^{n+k}}\right) \cdot \frac{|C|(|C|-1)}{2^{n-1}(2^{n-1}-1)} &= \frac{|C|}{2^{2(n+1)}} + \frac{1}{8}\left(\frac{2^{n-1}-1}{2^n}\right)\frac{|C|(|C|-1)}{2^{n-1}(2^{n-1}-1)} \\ &= \frac{|C|^2}{2^{2(n+1)}} = \mathcal{K}(C)^2 .\end{aligned}$$

Thus the claimed expected Kraft sum is $(\mathcal{K}(D_2)/\mathcal{K}(C))\mathcal{K}(C)^2 = \mathcal{K}(C)\mathcal{K}(D_2)$, by Lemma 4.5.10.

■

Proof of Lemma 4.8.4. The proof is similar to that of Lemma 4.6.1. For cases (i)–(iv), we will assume $2l - k < 1$.

(i) The set $1A^{l-2}0A^{n+k-l} \cap A^{k-l}bA^{l-2}C$ equals the set $U0A^n \cap A^{k-1}C$ (where $U \in \{0,1,A\}^{k-1}$ is a pattern with exactly three fixed bits), so, by Lemma 4.5.2, its expected Kraft sum is $\mathcal{K}(C)/8$.

(ii) We have

$$R_{n+1}(0A^{l-2}0A^{k-1} \cap A^{k-l}bA^{l-2}0A^{l-2}0A^{n-(l-1)}) = 2^{k-4}$$

by Lemma 4.5.3, since exactly 3 of the first $k-1$ positions in the set above are fixed bits. Therefore, using Lemma 4.5.6,

$$\begin{aligned}E[\mathcal{K}\left(C_0A^{k-l} \cap A^{k-1}bA^{l-2}C_0\right)] &= \frac{2^{k-4}}{2^{n+k}}\frac{|C_0|}{2^{n-1}} + \left(\frac{1}{2^5} - \frac{2^{k-4}}{2^{n+k}}\right)\frac{|C_0|(|C_0|-1)}{2^{n-1}(2^{n-1}-1)} \\ &= \frac{|C_0|}{2^{2n+3}} + \frac{1}{16}\left(\frac{2^{n-1}-1}{2^n}\right)\frac{|C_0|(|C_0|-1)}{2^{n-1}(2^{n-1}-1)} \\ &= \frac{1}{2}\cdot\frac{|C_0|^2}{2^{2(n+1)}} \\ &= \frac{\mathcal{K}(C_0)^2}{2}.\end{aligned}$$

Corollary 4.5.5 then implies

$$\begin{aligned}
E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}bA^{l-2}C\right)] &= E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}bA^{l-2}0A^{l-2}1A^{n-(l-1)}\right)] \\
&\quad + E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}bA^{l-2}C_0\right)] \\
&= \frac{\mathcal{K}(C_0)}{8} + \frac{\mathcal{K}(C_0)^2}{2} \\
&= \frac{\mathcal{K}(C_0)\left(\frac{1}{4} + \mathcal{K}(C_0)\right)}{2} \\
&= \frac{\mathcal{K}(C)\left(\frac{1}{4} - \mathcal{K}(C)\right)}{2}.
\end{aligned}$$

(iii) By Lemma 4.5.4, the claimed expected Kraft sum is $\mathcal{K}(C_0)\,\mathcal{K}(D_1) = (\mathcal{K}(C) - \frac{1}{4})\mathcal{K}(D_1)$.

(iv) By Lemma 4.5.4, the claimed expected Kraft sum is $\mathcal{K}(D_1) \cdot (1/4)\mathcal{K}(C)\,/(1/2) = \mathcal{K}(C)\,\mathcal{K}(D_1)\,/2$.

For cases (v)–(ix), we will assume $2l - k = 1$.

(v) The set $1A^{l-2}0A^{n+k-l} \cap A^{k-l}0A^{l-2}C$ equals the set $1A^{l-2}0A^{l-2}C$, so the claimed expected Kraft sum is $\mathcal{K}(C)\,/4$ by Lemma 4.5.2.

(vi) We have

$$\begin{aligned}
E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}0A^{l-2}C\right)] &= E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}0A^{l-2}0A^{l-1}1A^{n-(l-1)}\right)] \\
&\quad + E\left[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}0A^{l-2}C_0\right)\right].
\end{aligned}$$

The first expected Kraft sum on the right equals $\mathcal{K}(C_0)\,/4$ by Corollary 4.5.5, and the second expected Kraft sum on the right equals $\mathcal{K}(C_0)^2$ by Corollary 4.5.9. Thus the claimed expected Kraft sum is $\mathcal{K}(C_0)\,/4 + \mathcal{K}(C_0)^2 = \mathcal{K}(C_0)\left(\frac{1}{4} + \mathcal{K}(C_0)\right) = (\mathcal{K}(C) - \frac{1}{4})\mathcal{K}(C)$.

(vii) The set $1A^{l-2}0A^{n+k-l} \cap A^{k-l}D_0$ equals the set $1A^{l-2}D_0$, and so the claimed expected Kraft sum is $\mathcal{K}(D_0)\,/2$ by Lemma 4.5.2.

(viii) By Lemma 4.5.4, the claimed expected Kraft sum is $2\mathcal{K}(C_0)\,\mathcal{K}(D_0) = 2(\mathcal{K}(C) - \frac{1}{4})\mathcal{K}(D_0)$.

(ix) We have

$$\begin{aligned}
E[\mathcal{K}\left(D_0A^{k-l} \cap A^{k-l}1A^{l-2}C\right)] &= E[\mathcal{K}\left(D_0A^{k-l} \cap A^{k-l}1A^{l-2}C_0\right)] \\
&\quad + E[\mathcal{K}\left(D_0A^{k-l} \cap A^{k-l}1A^{l-2}0A^{l-2}1A^{n-(l-1)}\right)].
\end{aligned}$$

The first expected Kraft sum on the right equals $2\mathcal{K}(D_0) \cdot (1/8)\mathcal{K}(C_0)/(1/4) = \mathcal{K}(C_0)\mathcal{K}(D_0)$ by Lemma 4.5.4, and the second expected Kraft sum on the right equals $2\mathcal{K}(D_0)(1/8) = \mathcal{K}(D_0)/4$ by Lemma 4.5.4. Thus the claimed expected Kraft sum is

$$\mathcal{K}(C_0)\mathcal{K}(D_0) + \frac{\mathcal{K}(D_0)}{4} = (\mathcal{K}(C) - 1/4)\mathcal{K}(D_0) + \frac{\mathcal{K}(D_0)}{4} = \mathcal{K}(C)\mathcal{K}(D_0).$$

■

Proof of Lemma 4.8.5. The proof is similar to that of Lemma 4.6.1.

(i) We have $1A^{l-2}0A^{n+k-l} \cap A^{k-1}C = 1A^{l-2}0A^{k-l-1}C$, so its expected Kraft sum is $\mathcal{K}(C)/4$ by Lemma 4.5.2.

(ii) If $2l - k \neq 1$, then

$$\begin{aligned} E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-1}C\right)] &= E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}0A^{l-2}C\right)] + E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}1A^{l-2}C\right)] \\ &= \mathcal{K}(C)\left(\mathcal{K}(C) - \frac{1}{4}\right) \end{aligned}$$

by Lemma 4.8.4. If $2l - k = 1$, then

$$E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-1}C\right)] = E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}0A^{l-2}C\right)] = \mathcal{K}(C)\left(\mathcal{K}(C) - \frac{1}{4}\right)$$

by Lemma 4.8.4.

(iii) This case follows directly from Lemma 4.5.11.

For cases (iv) and (v), we will assume $2l - k < 1$.

(iv) We have

$$E[\mathcal{K}\left(1A^{l-2}0A^{n+k-l} \cap A^{k-l}1A^{l-2}C\right)] = E[\mathcal{K}\left(1A^{l-2}0A^{k-2l}1A^{l-2}C\right)] = \mathcal{K}(C)/8$$

by Lemma 4.5.2. Then by Lemma 4.5.10, the claimed expected Kraft sum is
$(\mathcal{K}(D)/(\mathcal{K}(C)/2))(\mathcal{K}(C)/8) = \mathcal{K}(D)/4$.

(v) We have

$$\begin{aligned} E[\mathcal{K}\left(C_0A^{k-1} \cap A^{k-l}1A^{l-2}C\right)] &= E[\mathcal{K}\left(C_0A^{k-2l}1A^{l-2}0A^{l-2}1A^{n-(l-1)}\right)] \\ &\quad + E[\mathcal{K}\left(C_0A^{k-2l}1A^{l-2}C_0\right)] \\ &= \frac{\mathcal{K}\left(C_0\right)}{8} + \frac{\mathcal{K}\left(C_0\right)^2}{2} \\ &= \frac{(\mathcal{K}\left(C\right) - \frac{1}{4})\mathcal{K}\left(C\right)}{2} \end{aligned}$$

by Lemma 4.5.2 and Corollary 4.5.9, and using the fact that $C_0A^{k-2l}1A^{l-2}C_0$ contains exactly half of the words of $C_0A^{k-l-1}C_0$. Then by Lemma 4.5.10, the claimed expected Kraft sum is
$(\mathcal{K}\left(D\right)/(\mathcal{K}\left(C\right)/2))((\mathcal{K}\left(C\right) - \frac{1}{4})\mathcal{K}\left(C\right)/2) = (\mathcal{K}\left(C\right) - \frac{1}{4})\mathcal{K}\left(D\right)$.

■

Chapter 4 is a reprint of the material as it appears in: S. Congero and K. Zeger, "The 3/4 Conjecture for fix-free codes with at most three distinct codeword lengths", *IEEE Transactions on Information Theory*, vol. 69, no. 3, pp. 1452-1485, March 2023.

References

[1] N. Abedini, S. P. Khatri, and S. A. Savari, "A SAT-based scheme to determine optimal fix-free codes", *Data Compression Conference*, March 2010, pp. 169 -– 178.

[2] A. Aghajan and M. Khosravifard, "93% of the $3/4$-conjecture is already verified", *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8182 – 8194, December 2013.

[3] A. Aghajan and M. Khosravifard, "Weakly symmetric fix-free codes", *IEEE Transactions on Information Theory*, vol. 60, no. 9, pp. 5500 – 5515, September 2014.

[4] R. Ahlswede, B. Balkenhol and L. Khachatrian, "Some properties of fix-free codes", 1^{st} *Intas Seminar on Coding Theory and Combinatorics*, Thahkadzor, Armenia, pp. 20 – 33, 1996.

[5] R. Bauer and J. Hagenauer, "Iterative source/channel-decoding using reversible variable length codes", *Data Compression Conference (DCC)*, Snowbird, Utah, pp. 93 – 102, March 2000.

[6] M.-P. Béal, J. Berstel, B. H. Marcus, D. Perrin, C. Reutenauer, and P. Siegel, "Variable-length codes and finite automata", *Selected Topics in Information and Coding Theory*, Series on Coding Theory and Cryptology, World Scientific, pp. 505 – 584, 2010.

[7] J. Berstel and D. Perrin, *Theory of Codes*, Academic Press, 1985.

[8] J. Berstel, C. De Felice, D. Perrin, Ch. Reutenauer, and G. Rindone, "Bifix codes and Sturmian words", *Journal of Algebra*, vol. 369, pp. 146 – 202, 2012.

[9] V. Berthé, C. De Felice, F. Dolce, J. Leroy, D. Perrin, C. Reutenauer, and G. Rindone, "Acyclic, connected and tree sets", *Monatshefte für Mathematik*, vol. 176, pp. 521 – 550, 2015.

[10] V. Berthé, C. De Felice, F. Dolce, J. Leroy, D. Perrin, C. Reutenauer, and G. Rindone, "The finite index basis property", *Journal of Pure and Applied Algebra*, vol. 219, pp. 2521 – 2537, 2015.

[11] V. Berthé, C. De Felice, F. Dolce, J. Leroy, D. Perrin, C. Reutenauer, and G. Rindone, "Bifix codes and interval exchanges", *Journal of Pure and Applied Algebra*, vol. 219, pp. 2781 – 2798, 2015.

[12] V. Berthé, C. De Felice, F. Dolce, J. Leroy, D. Perrin, C. Reutenauer, and G. Rindone, "Maximal bifix decoding", *Discrete Mathematics*, vol. 338, no. 5, pp. 725 – 742, 2015.

[13] M. Bodewig, "Multiplied complete fix-free codes and shiftings regarding the $3/4$-conjecture", *Information Theory, Combinatorics, and Search Theory: In Memory of Rudolf Ahlswede*, Lecture Notes in Computer Science, vol. 7777, pp. 694 – 710, Springer, Berlin, 2013.

[14] M. Bodewig, "An Introduction of greedy extension sets for the application on fix-free codes", *Ph.D thesis*, Aachen University, Germany, 2015.

[15] M. Bystrom, S. Kaiser, and A. Kopansky, "Soft source decoding with applications", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 10, pp. 1108 – 1120, October 2001.

[16] Y. Césari, "Propriétés combinatoires des codes biprefixes", In *Théorie des codes*, (D. Perrin, ed.), pp. 20 – 46. Paris: LITP, 1979.

[17] Y. Césari, "Sur un algorithme donnant les codes bipréfixes finis", *Math. Systems Theory*, vol. 6, pp. 221 – 225, 1982.

[18] T. Cover and J. Thomas, *Elements of Information Theory*, Wiley & Sons, 1991.

[19] C. Deppe and H. Schnettler, "On the $3/4$-conjecture for fix-free codes", *European Conference on Combinatorics, Graph Theory and Applications (EuroComb)*, Berlin, Germany. pp. 111 – 116. 2005.

[20] C. Deppe and H. Schnettler, "On q-ary fix-free codes and directed de Brujin graphs", *IEEE International Symposium on Information Theory*, 2006.

[21] A. S. Fraenkel and S. T. Klein, "Bidirectional Huffman coding", *Computer Journal*, vol. 33, no. 4, pp. 296 – 307, 1990.

[22] S.-S. Gao and G.-F. Tu, "Robust H.263+ video transmission using partial backward decodable bit stream (PBDBS)", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 2, pp. 182 – 187, February 2003.

[23] E. N. Gilbert and E. F. Moore, "Variable-length binary encodings", *Bell System Technical Journal*, vol. 38, pp. 933 – 967, July 1959.

[24] D. Gillman and R. L. Rivest, "Complete variable-length fix-free codes", *Designs, Codes and Cryptography*, vol. 5, no. 2, pp. 109 – 114, March 1995.

[25] B. Girod, "Bidirectionally decodable streams of prefix code-words", *IEEE Communications Letters*, vol. 3, no. 8, pp. 245 – 247, August 1999.

[26] X.Guang, F.-W. Fu, L.-S. Chen, "The existence and synchronization properties of symmetric fix-free codes", *Science China Information Sciences*, pp. 1 – 9, September 2013.

[27] R. Gupta and R. Goel, "A necessary and sufficient condition for the existence of asymmetrical reversible VLCs", *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, no. 4, pp. 314 – 317, February 2019.

[28] K. Harada and K. Kobayashi, "A note on the fix-free property", *IEICE Transactions on Fundamentals*, vol. E82-A, no. 10, pp. 2121 – 2128, October 1999.

[29] D. Huffman, "A method for the construction of minimum-redundancy codes", *Proceedings of IRE*, vol. 40, no. 9, pp. 1098 – 1101, September 1952.

[30] J.-Y. Huo, Y.-L. Chang, L.-H. Ma, and Z. Luo, "On constructing symmetrical reversible variable-length codes independent of the Huffman code", *Journal of Zhehiang University - Science A*, vol. 7, pp. 59 – 62, 2006.

[31] *ISO/IEC 14496-2*, "Information technology - coding of audio/visual objects", Final Draft International Standard, Part 2, Visual, October 1998.

[32] *ITU-T Recommendation H.263*, "Video coding for low bit rate communications," Annex V, 2000.

[33] M. Javad-Kalbasi and M. Khosravifard, "Some tight lower bounds on the redundancy of optimal binary prefix-free and fix-free codes", *IEEE Transactions on Information Theory*, vol. 66, no. 7, pp. 4419 – 4430, July 2020.

[34] W.-H. Jeong, Y.-S. Yoon, and Y.-S. Ho, "Design of reversible variable-length codes using properties of the Huffman code and average length function", *International Conference on Image Processing (ICIP)*, Singapore, vol. 2, pp. 817 – 820, November 2004.

[35] S. Kaiser and M. Bystrom, "Soft decoding of variable-length codes", *IEEE International Conference on Communications (ICC)*, New Orleans, Louisiana, vol. 3, pp. 1203 – 1207, June 2000.

[36] A. Kakhbod and M. Zadimoghaddam, "Some notes on fix-free codes", *42nd Annual Conference on Information Sciences and Systems (CISS)*, Princeton, NJ, pp. 1015 – 1018, March 2008.

[37] A. Kakhbod and M. Zadimoghaddam, "On the construction of prefix-free and fix-free codes with specified codeword compositions", *Discrete Applied Mathematics*, vol. 159, no. 18, pp. 2269 – 2275, December 2011.

[38] M. Khosravifard and T. Gulliver, "On the capability of the Harada-Kobayashi algorithm in finding fix-free codewords", *International Symposium on Information Theory and Its Applications (ISITA)*, Auckland, New Zealand, pp. 1 – 4, December 2008.

[39] M. Khosravifard and T. Gulliver, "The redundancy of an optimal binary fix-free code is not greater than 1 bit", *IEEE Transactions on Information Theory*, vol. 61, no. 6, pp. 3549 – 3558, June 2015.

[40] M. Khosravifard, H. Halabian, and T. Gulliver, "A Kraft-type sufficient condition for the existence of D-ary fix-free codes", *IEEE Transactions on Information Theory*, vol. 56, no. 6, pp. 2920 – 2927, June 2010.

[41] M. Khosravifard and S. Kheradmand, "Some upper bounds on the redundancy of optimal binary fix-free codes", *IEEE Transactions on Information Theory*, vol. 58, no. 6 , pp. 4049 – 4057, June 2012.

[42] J. Kliewer and R. Thobaben, "Iterative joint source-channel decoding of variable-length codes using residual source redundancy", *IEEE Transactions on Wireless Communications*, vol. 4, no. 3, pp. 919 – 929, May 2005.

[43] L. G. Kraft, "A device for quantizing, grouping, and coding amplitude modulated pulses", *Master's Thesis*, Department of Electrical Engineering, MIT Cambridge, MA, 1949.

[44] Zs. Kukorelly and K. Zeger, "Sufficient conditions for existence of binary fix-free codes", *IEEE Transactions on Information Theory*, vol. 51, no. 10, pp. 3433 – 3444, November 2005.

[45] K. Laković and J. Villasenor, "On design of error-correcting reversible variable length codes", *IEEE Communications Letters*, vol. 6, no. 8, pp. 337 – 339, August 2002.

[46] K. Laković and J. Villasenor, "An algorithm for construction of efficient fix-free codes", *IEEE Communications Letters*, vol. 7, no. 8, pp. 391 – 393, August 2003.

[47] M. Leonard, "A property of biprefix codes", *Informatique théorique et Applications/Theoretical Informaties and Applications*, vol. 22, no. 3, pp. 311 – 318, 1988.

[48] C. Lin, J. Wu, and Y. Chuang, "Two Algorithms for Constructing Efficient Huffman-Code-Based Reversible Variable Length Codes", *IEEE Transactions on Communications*, vol. 56, no. 1, pp. 81 – 89, January 2008.

[49] D. Perrin, "Codes bipréfixes et groupes de permutations," *Thèse*, Université Paris 7, 1975.

[50] D. Perrin, "La transitivité du groupe d'un code bipréfixe fini", *Mathematische Zeitschrift*, vol. 153, pp. 283 – 287, 1977.

[51] D. Perrin, "Le degré minimal du groupe d'un code bipréfixe fini", *Journal of Combinatorial Theory, Series A*, vol. 25, pp. 163 – 173, 1978.

[52] D. Perrin, "Completing biprefix codes", In *Automata, Languages and Programming, Lecture Notes in Computer Science*, vol. 140, pp. 397 – 406, 1982.

[53] J. L. Peterson, "Computer programs for detecting and correcting spelling errors", *Communications of the ACM*, vol. 23, pp. 676 – 687, 1980.

[54] C. Reutenauer "Semisimplicity of the algebra associated to a biprefix code", *Semigroup Forum*, vol. 23, pp. 327 – 342, 1981.

[55] S. Savari, S. Yazdi, N. Abedini, and S. Khatri, "On optimal and achievable fix-free codes", *IEEE Transactions on Information Theory*, vol. 58, no. 8 , pp. 5112 – 5129, August 2012.

[56] H. Schnettler, "On the $3/4$-conjecture for fix-free codes: a survey", *arXiv:0709.2598v1*, September 2007.

[57] M. P. Schützenberger "On an application of semigroup methods to some problems in coding", *IRE Transactions on Information Theory*, vol. 2, pp. 47 – 60, September 1956.

[58] M. P. Schützenberger, "On a special class of recurrent events", *Annals of Mathematical Statistics*, vol. 32, pp. 1201 – 1213, 1961.

[59] M. P. Schützenberger, "On a family of submonoids", *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, (Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei), vol. 6, pp. 381 – 391, 1961.

[60] C. E. Shannon, "A mathematical theory of communication", *Bell System Technical Journal*, vol. 27, no. 4, pp. 623 – 666, October 1948.

[61] Y. Takishima, M. Wada, and H. Murakami, "Reversible variable length codes", *IEEE Transactions on Communications*, vol. 43, pp. 158 – 162, February – April 1995.

[62] C.-W. Tsai, T.-J. Huang, K.-L. Fang, and J.-L. Wu, "A hybrid and flexible H.263-based error resilient and testing system", *IEEE Region 10 International Conference on Electrical and Electronic Technology*, vol. 1, pp. 122 – 128, August 2001.

[63] C.-W. Tsai and J.-L. Wu "On constructing the Huffman-code-based reversible variable-length codes", *IEEE Transactions on Communications*, vol. 49, no. 9, pp. 1506 – 1509, September 2001.

[64] C.-W. Tsai and J.-L. Wu, "Modified symmetrical reversible variable-length code and its theoretical bounds", *IEEE Transactions on Information Theory*, vol. 47, no. 6, pp. 2543 – 2548, September 2001.

[65] C.-W. Tsai J.-L. Wu, and S.-W. Liu, "Modified symmetrical reversible variable length code and its theoretical bounds", *Proceedings of the SPIE*, vol. 3974, pp. 606 – 616, April 2000.

[66] H.-W. Tseng and C.-C. Chang, "Construction of symmetrical reversible variable length codes using backtracking", *The Computer Journal* vol. 46, no. 1, pp. 100–105, January 2003.

[67] H. Wang, S.N. Koh, and W.-W. Chang, "Application of reversible variable-length codes in robust speech coding", *IEE Proceedings - Communications*, vol. 152, no. 3, pp. 272 – 276, July 2005.

[68] J. Wang, L.-L. Yang, and L. Hanzo, "Iterative construction of reversible variable-length codes and variable-length error-correcting codes", *IEEE Communications Letters*, vol. 8, no. 11, pp. 671 – 673, November 2004.

[69] J.L.H. Webb, "Efficient table access for reversible variable-length decoding", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 8, pp. 981 – 985, August 2001.

[70] J. Wen and J. D. Villasenor, "A class of reversible variable length codes for robust image and video coding", *IEEE International Conference on Image Processing (ICIP)*, Santa Barbara, CA, vol. 2, pp. 65 – 68, October 1997.

[71] J. Wen and J. D. Villasenor, "Reversible variable length codes for efficient and robust image and video coding", *Data Compression Conference (DCC)*, Snowbird, UT, pp. 471 – 480, March – April 1998.

[72] N. Yadav, K. C. Roy, and Y. Krishan, "Construction of reversible variable length code for digital image processing", *International Journal of Engineering Science and Technology*, vol. 2, no. 10, pp. 5332 – 5336, October 2010.

[73] Z. Yan, S. Kumar, J. Li and C. C. J. Kuo, "Reversible variable length codes (RVLC) for robust coding of 3D topological mesh data", *Data Compression Conference (DCC)*, Snowbird, UT, pp. 560, March 1999.

[74] S. M. H. T. Yazdi and S. A. Savari, "On the relationships among optimal symmetric fix-free codes," *IEEE Transactions on Information Theory*, vol. 60, no. 8, pp. 4567 – 4583, August 2014.

[75] C. Ye and R. W. Yeung, "Some basic properties of fix-free codes", *IEEE Transactions on Information Theory*, vol. 47, no. 1, pp. 72 – 87, January 2001.

[76] S. Yekhanin, "Sufficient conditions of existence of fix-free codes", *IEEE International Symposium on Information Theory (ISIT)*, Washington, D.C., p. 284, June 2001.

[77] S. Yekhanin, "Improved upper bound for the redundancy of fix-free codes", *IEEE Transactions on Information Theory*, vol. 50, no. 11, pp. 2815 – 2818, November 2004.

[78] S. Yekhanin, "Sufficient conditions of existence of fix-free codes", preprint.

[79] A. Zaghian, A. Aghajan, and T.A. Gulliver, "The optimal fix-free code for anti-uniform sources", *Entropy*, vol. 17, no. 3, pp. 1379 – 1386, March 2015.

[80] S. J. Zahabi, A. Aghajan, and M. Khosravifard, "Sequentially-constructible reversible variable length codes", *IEEE Transactions on Communications*. vol. 62 , no. 8, pp. 2605 – 2614, August 2014.

[81] S. J. Zahabi and M. Khosravifard, "On the penalty of optimal fix-free codes", *IEEE Transactions on Information Theory*, vol. 61 , no. 5, pp. 2776 – 2787, May 2015.

[82] A. Zammit, "Reversible variable-length codes", *Master's Dissertation*, University of Malta, 2007.

Chapter 5

Characterizations of Minimal Expected Length Codes

Abstract

A property of prefix codes called strong monotonicity is introduced. Then it is proven that for a prefix code C for a given probability distribution, the following are equivalent: (i) C is expected length minimal; (ii) C is length equivalent to a Huffman code; and (iii) C is complete and strongly monotone. Also, three relations are introduced between prefix code trees called same-parent, same-row, and same-probability swap equivalence, and it is shown that for a given source, all Huffman codes are same-parent, same-probability swap equivalent, and all expected length minimal prefix codes are same-row, same-probability swap equivalent.

5.1 Introduction

Huffman codes [13] were invented in 1952 and today are widely used in many practical data compression applications, such as for text, audio, image, and video coding. They are known to be optimal in the sense that they achieve the minimal possible expected codeword length among all prefix codes for a given finite discrete random source [5].

The main idea in the Huffman algorithm is to construct a code tree from a source by recursively merging two smallest-probability nodes until only one node with probability 1 remains. The initial source probabilities correspond to leaf nodes in the tree, and the binary paths from the tree's root to the leaves are the codewords.

However, for a given source, Huffman codes are not unique, due to choices that arise during the tree construction that can be decided arbitrarily. Specifically, there are three types of choices that can be taken during the tree building: (1) When two nodes are merged, the choice of which node becomes a left child and which becomes a right child is arbitrary; (2) If there are three or more smallest-probability nodes, then which two of them to merge is arbitrary; and (3) If there is a unique smallest-probability node and two or more second-smallest-probability nodes, then which of these to merge with the smallest-probability node is arbitrary. After the tree is constructed all edges from parents to left children are labeled 0 and all edges from parents to right children are labeled 1, or vice versa. Without loss of generality, we will not interpret this binary edge labeling as a choice for generating multiple Huffman codes, since the same-parent node swaps that we address later in the paper can account for such constructions, too. Such arbitrary choices as in (1)–(3) made during the Huffman construction process can affect not only the codeword assignments, but also the Huffman tree structure, and can even change the distribution of codeword lengths.

On one hand, if a source is chosen randomly from a continuous distribution (i.e., the source probabilities are randomly chosen to lie in $[0, 1]$ and to sum to 1), then with probability one there will be no ties among source probabilities and no ties among tree node probabilities in the Huffman construction process. In this case, the only variation of Huffman codes for a given randomly chosen source is due to left-versus-right child assignments when node merges occur during Huffman tree construction.

On the other hand, often source distributions are empirically determined through a frequency counting process, and probability estimates consist of a set of integers, normalized by their sum. In these cases, especially with small data sets, ties in probabilities can occur, and multiple Huffman trees can result. The differences in these Huffman trees can be due to some or all of the arbitrary choices mentioned above that are encountered during Huffman tree construction.

For many applications, the average length of a prefix code is the primary concern, in which case the

choice of which Huffman or other optimal non-Huffman code to use may not matter, although an understanding of such code variations may be of theoretical interest. In some applications, however, the specific binary codewords included in an optimal code may be critical. A survey of lossless coding techniques can be found in [1].

One well-studied example where the codeword assignments matter is the design of lossless codes that are easily synchronizable. Since Huffman codes are variable-length codes, they are subject to loss of synchronization during decoding due to even a single bit error or erasure during transmission or storage. However, Huffman codes are known to often have a self-synchronizing string, which is a binary string (not necessarily a codeword) that, after being decoded starting at any internal tree node, always returns the decoding process to the root of the Huffman tree, thus restoring synchronization. It turns out that for a given source with multiple Huffman codes, some Huffman codes may have shorter self-synchronizing strings than others, and some Huffman codes may not have any self-synchronizing strings at all even if other Huffman codes do. These possibilities are illustrated in the following two examples.

Example 5.1.1 (Same-parent node swap produces shorter self-synchronizing string)**.**
Let H_1 and H_2 be Huffman codes for a source with symbols a,b,c,d, and probabilities $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{8}$, respectively. The Huffman trees for H_1 and H_2 are shown in Figure 5.1. Huffman tree H_2 is obtained from H_1 by exchanging node b and its sibling (i.e., a same-parent node swap). The shortest self-synchronizing strings for Huffman trees H_1 and H_2 are 0 and 00, respectively.

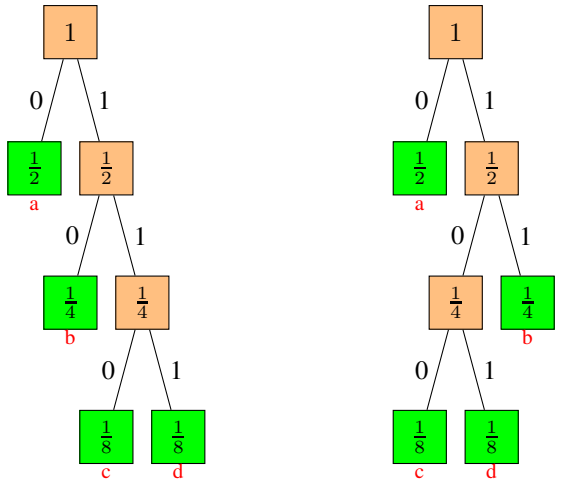


Figure 5.1. One Huffman tree is a same-parent node swap of another and has a shorter self-synchronizing string.

Example 5.1.2 (Same-parent node swap eliminates self-synchronizing string)**.**
Let H_1 and H_2 be Huffman codes for a source with symbols a,b,c,d,e,f,g,h,i, and probabilities $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{8}$, $\frac{1}{8}$, $\frac{1}{8}$,

$\frac{1}{16}$, $\frac{1}{16}$, $\frac{1}{16}$, $\frac{1}{16}$, respectively. The Huffman trees for H_1 and H_2 are shown in Figure 5.2. Huffman tree H_2 is obtained from H_1 by exchanging the leaf b and its sibling (i.e., a same-parent node swap). Huffman tree H_1 has a self-synchronizing string of 0011, which brings each internal node back to the root. Huffman tree H_2 does not have any self-synchronizing string, since any string which brings the root back to itself also brings the parents of a, b, and c back to one of themselves, and thus not to the root.

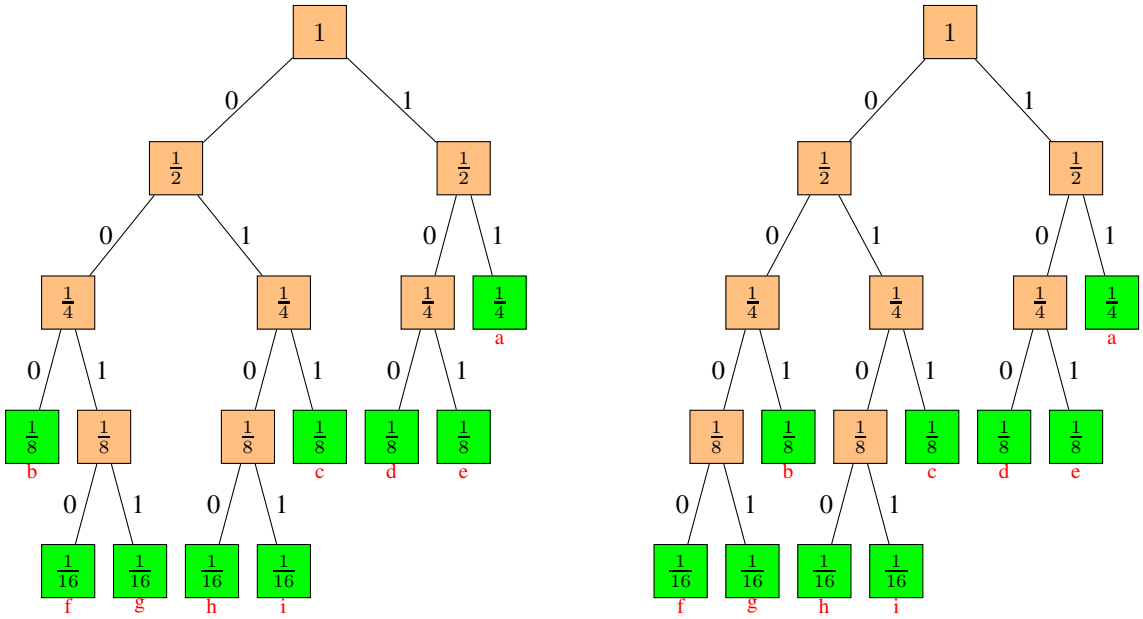


Figure 5.2. Huffman tree H_2 is a same-parent node swap of H_1, but has no self-synchronizing string whereas H_1 does.

Numerous theoretical and algorithmic studies of synchronizable Huffman and non-Huffman optimal prefix codes have made use of the non-uniqueness of Huffman codes to search for short synchronizing binary strings. Some of these investigations include Longo and Galasso [16] in 1982, Ferguson and Rabinowitz [8] in 1984, Escott and Perkins [7] in 1998, Huang and Wu [12] in 2003, and Higgs, Perkins, and Smith [11] (see also [10]) in 2009. Other related work considers correcting bit errors in Huffman encoded data streams, in which case the choice of Huffman code can affect performance. Some of these include: Lee, Chang, Ho, and Lee [14] in 2000, Zhou and Zhang [20] in 2002, Cao [2] in 2006, Cao, Yao, and Chen [3] in 2007, Zhou and Au [19] in 2010, and Wang, Zhao, and Sun [17] in 2018.

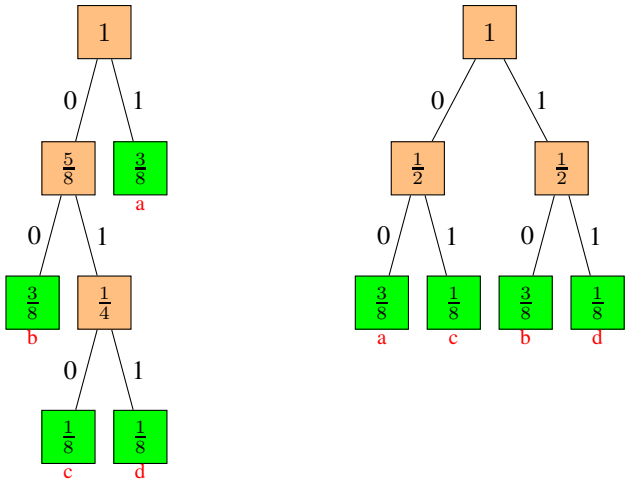
The non-uniqueness of Huffman codes leads to the question of how to effectively describe the similarities among them, as well as their differences from non-Huffman codes. In 1978, Gallager [9] gave a characterization of Huffman codes as precisely those prefix codes possessing a "sibling property", which stipulates that a code is

complete and the nodes of its code tree can be listed in order of non-increasing probability with each node being adjacent in the list to its sibling.

For a given source, the broader class of optimal prefix codes is somewhat larger then its subset of Huffman codes. No characterization analogous to the sibling property has been previously given for optimal prefix codes. Only the sufficient condition given by the sibling property has been known.

One known necessary condition for a prefix code to be optimal is "monotonicity", which states that any code tree node with a larger probability than another node must not appear on a lower row than the smaller-probability node. The following example illustrates that monotonicity is not sufficient for a complete prefix code to be optimal.

Example 5.1.3 (A monotone prefix code that is not optimal)**.**
Let H be a Huffman code for a source with symbols a,b,c,d, and probabilities $\frac{3}{8}$, $\frac{3}{8}$, $\frac{1}{8}$, $\frac{1}{8}$, respectively, and let C be another prefix code for the same source. The code trees for H and C are shown in Figure 5.3. The code C is monotone because any node probability on a given row is at least as large as any node probability on a lower row. However, C is not optimal since its expected length is 2, whereas the Huffman code H has a smaller expected length of $\frac{15}{8}$.



Huffman code H Prefix code C

Figure 5.3. A Huffman tree and code tree illustrating monotonicity without strong monotonicity.

In this paper we first provide a necessary and sufficient characterization of optimal prefix codes by introducing a new criterion called "strong monotonicity". In particular we show that for a prefix code C for a given source, the following are equivalent: (i) C is optimal; (ii) C is length equivalent to a Huffman code; and (iii) C is complete and strongly monotone.

Secondly, we investigate the transformation of code trees to other code trees using the graph theoretic idea of swapping tree nodes. Swapping two nodes with a similar trait can transform one code tree into another code tree that maintains certain properties, and can serve as a method for creating new Huffman codes or new optimal prefix codes from existing ones.

Specifically, we consider swaps of two code tree nodes when they: (i) have the same parent; (ii) lie in the same row; or (iii) have the same probability. For any given source, these three types of node swaps always preserve the expected length of a prefix code. For example, any optimal prefix code will be transformed by any of these operations into another optimal prefix code.

Some prior work related to node swapping has motivated some of our results.

In 1982, Longo and Galasso [16] used same-probability node swaps in the context of determining probability density attraction regions for Huffman codes. They showed that, using only same-probability node swaps, Huffman codes could be transformed into other Huffman codes. However, they ignored the distinguishing effects of same-row node swaps that could transform Huffman codes to optimal non-Huffman codes. Our results more finely characterize the relationship between these codes by using either same-parent or same-row node swaps. Also we use a different proof technique and try to clarify some unaddressed points in [16].

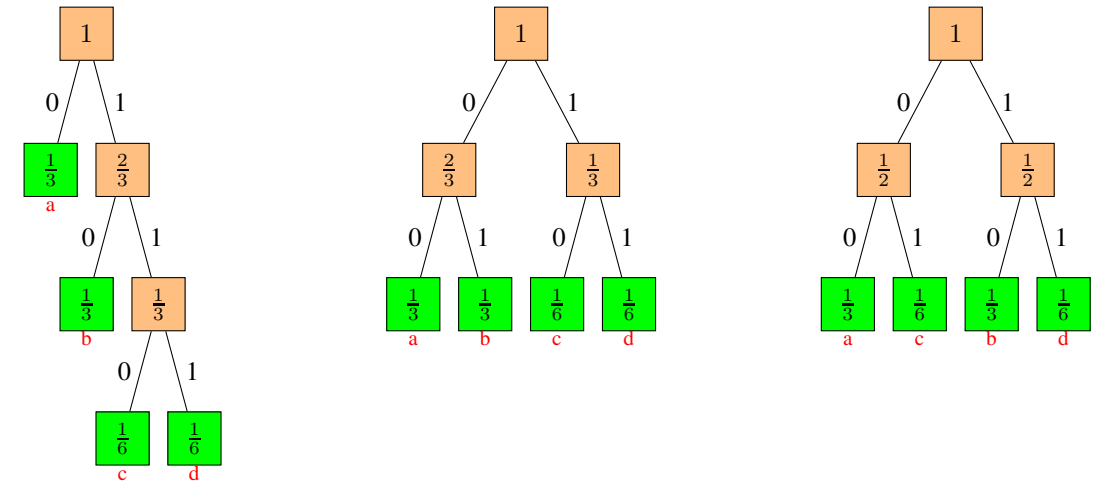
Ferguson and Rabinowitz [8] studied synchronous codes and considered codes that were same-parent swap equivalent (calling them "strongly-equivalent") and length equivalent (calling them "weakly equivalent"), but did not provide results characterizing the class of Huffman or optimal prefix codes.

We characterize both the class of all Huffman codes for a given source and also the broader class of all optimal prefix codes for a given source in terms of same-probability and either same-row or same-parent node swap transformations.

Example 5.1.4 (Two Huffman codes and a third optimal code tree)**.**
Let H_1 and H_2 be two different Huffman codes and let C be a non-Huffman tree for a source with symbols a,b,c,d, and probabilities $\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6}$, respectively. The three trees are shown in Figure 5.4. After nodes c and d were combined during the Huffman construction algorithm for building H_1 and H_2, 3 different nodes had probability $\frac{1}{3}$, leading to different trees resulting from different node merges. All 3 codes achieve the minimum possible average length of 2, but the codes for H_1 and H_2 are not same-row swap equivalent, since the codes are not length equivalent.

However, H_1 and H_2 are same-parent, same-probability swap equivalent. To see this, first transform tree H_1 by exchanging node a with the parent of nodes c and d (i.e., a same-probability node swap). Then perform a same-parent node swap on nodes a and b, and then another same-parent node swap on the two children of the root. The result of these three operations is the Huffman tree H_2.

The code C is not same-parent, same-probability swap equivalent to either H_1 or H_2, but it is same-row, same-probability swap equivalent to both Huffman codes. To see this, transform H_2 to C by exchanging nodes b and c (i.e., a same-row node swap).



Huffman code H_1 Huffman code H_2 optimal code C

Figure 5.4. Two Huffman trees and an optimal third code tree for a single source.

We show that in fact for a given source, any Huffman code can be transformed into any other Huffman code by a sequence of same-parent and same-probability node swaps. Then we show that for a given source, any optimal code can be transformed into any other optimal code by a sequence of same-row and same-probability node swaps. These characterizations of Huffman codes and optimal codes refine a result in [16] and supplement the foundational understanding of optimal lossless source coding.

In what follows we will define terminology and then present our main results. One of these results (Theorems 5.2.4) is exploited in another recent work [4] to prove results about competitive optimality of Huffman codes.

An *alphabet* is a finite set S, and a *source* of size n with alphabet S is a random variable X such that $|S| = n$ and $P(X = y) = P(y)$ for all $y \in S$. We denote the probability of any subset $B \subseteq S$ by $P(B) = \sum_{y \in B} P(y)$.

A *code* for source X is a mapping $C : S \longrightarrow \{0,1\}^*$ and the binary strings $C(1), \ldots, C(n)$ are called *codewords* of C. A *prefix code* is a code where no codeword is a prefix of any other codeword.

A *code tree* for a prefix code C is a rooted binary tree whose leaves correspond to the codewords of C; specifically, the codeword associated with each leaf is the binary word denoting the path from the root to the leaf. The *length* of a code tree node is its path length from the root. The rth *row* of a code tree is the set of nodes whose

length is r, and we will view a code tree's root as being on the top of the tree with the tree growing downward. That is, row r of a code tree is "higher" in the tree than row $r+1$. If x and y are nodes in a code tree, then x is a *descendant* of y if there is a downward path of length zero or more from y to x. Two nodes in a tree are called *siblings* if they have the same parent. For any collection A of nodes in a code tree, let $P(A)$ denote the probability of the set of all leaf descendants of A in the tree.

A (binary) *Huffman tree* is a code tree constructed from a source by recursively merging two smallest-probability nodes[1] until only one node with probability 1 remains. The initial source probabilities correspond to leaf nodes in the tree. A *Huffman code* for a given source is a mapping of source symbols to binary words by assigning the source symbol corresponding to each leaf in the Huffman tree to the binary word describing the path from the root to that leaf.

Given a source with alphabet S and a prefix code C, for each $y \in S$ the length of the binary codeword $C(y)$ is denoted $l_C(y)$. Two codes C_1 and C_2 are *length equivalent* if $l_{C_1}(y) = l_{C_2}(y)$ for every source symbol $y \in S$. The *average length* of a code C for a source with alphabet S is $\sum_{y \in S} l_C(y)P(y)$. A prefix code is *optimal* for a given source if no other prefix code achieves a smaller average codeword length for the source. In particular, Huffman codes are known to be optimal (e.g., see [5]).

A code is *complete* if every non-root node in its code tree has a sibling, or, equivalently, if every node has either zero or two children.[2] A code C for a given source is *monotone* if for any two nodes in the code tree of C, we have $P(u) \geq P(v)$ whenever $l_C(u) < l_C(v)$. Optimal prefix codes are always monotone (Lemma 5.1.6) and are also well-known to be complete.

The *Kraft sum* of a sequence of nonnegative integers $l_1, \ldots, l_k$ is $2^{-l_1} + \cdots + 2^{-l_k}$. We extend the definition of "Kraft sum" to also apply to sets of source symbols with respect to a code or sets of code tree nodes. In each case, we use the notation K_C to denote the Kraft sum. If C is a prefix code for a source with alphabet S, and $U \subset S$, then the Kraft sum of U is

$$K_C(U) = \sum_{x \in U} 2^{-l_C(x)}$$

or equivalently, the Kraft sum of the corresponding sequence of codeword lengths of the set of all leaf descendants of U in the code tree of C. The same summation is used to compute the Kraft sum of a set of code tree nodes.

The following lemma is a standard result in most information theory textbooks and will be used in the proofs of Lemma 5.2.2 and Theorem 5.2.4.

[1] For more details about Huffman codes, the reader is referred to the textbook [5, Section 5.6].

[2] Our usage of the word "complete" has also been referred to in the literature as "full", "extended", "saturated", "exhaustive", and "maximal".

Lemma 5.1.5 (Kraft Inequality converse [5, Theorem 5.2.1]). *If a sequence $l_1, \ldots, l_n$ of positive integers satisfies $2^{-l_1} + \cdots + 2^{-l_n} \leq 1$, then there exists a binary prefix code whose codeword lengths are $l_1, \ldots, l_n$.*

The following lemma is used in the proof of Lemma 5.3.4.

Lemma 5.1.6 (Gallager [9, p. 670]). *For any source, if a prefix code is optimal, then it is monotone.*

As noted earlier, Gallager characterized Huffman codes (Lemma 5.1.8) using the following property.

Definition 5.1.7 (Gallager [9, p. 669]). A binary code tree has the *sibling property* if each node (except the root) has a sibling and if the nodes can be listed in order of non-increasing probability with each node being adjacent in the list to its sibling.

The next lemma is very useful in proving results about Huffman codes, and will be exploited in the proofs of Theorem 5.2.4, Lemma 5.3.4, and Theorem 5.3.8.

Lemma 5.1.8 (Gallager [9, Theorem 1]). *For any source, a prefix code is a Huffman code if and only if its code tree has the sibling property.*

For the remainder of this section, we describe our main results. Theorem 5.2.4 shows that for a prefix code C for a given probability distribution, the following are equivalent: (i) C is optimal; (ii) C is length equivalent to a Huffman code; and (iii) C is complete and strongly monotone. In this theorem, the Huffman code in case (ii) may vary for different choices of C. Figure 5.5 depicts these results along with some known prior art.

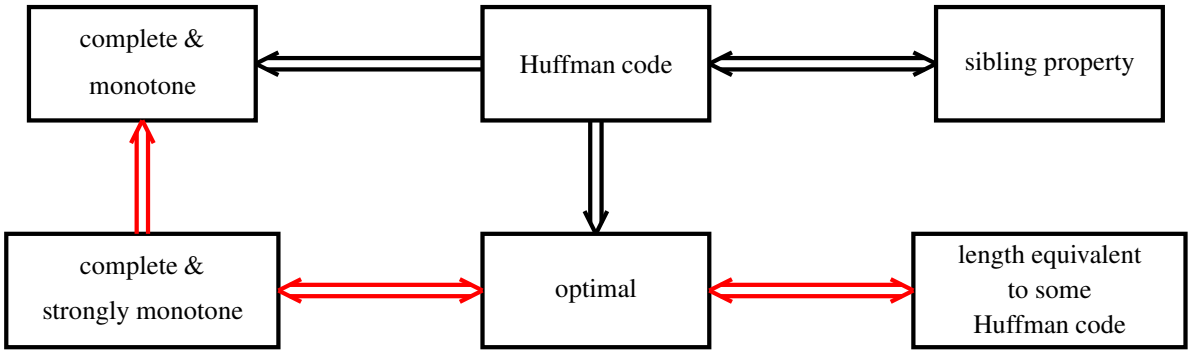


Figure 5.5. Logical implications of prefix code properties for a given source. The red arrows indicate new results presented in this paper.

Next, we describe our remaining results in this paper.

Huffman codes for the same source can differ from each other in multiple ways. For example, "twisting" a Huffman tree about any fixed node in the tree (i.e., swapping two same-parent nodes) creates a new Huffman code for the source (Lemma 5.3.3).

Another transformation of a Huffman tree is to cut branches of the tree at two nodes whose probabilities equal each other and then exchange those subtrees (i.e., perform a same-probability node swap). Two such nodes need not lie on the same row in the tree. This transformation also results in a new Huffman code for the same source (Lemma 5.3.4).

Any combination of these same-parent and same-probability node swaps transforms Huffman trees into Huffman trees. In fact, we show in Theorem 5.3.8 that the converse is also true, namely that all Huffman codes for a given source are related by these transformations.

More general questions exist about the broader class of optimal prefix codes. In this case, we know that any combination of same-parent and same-probability node swaps transforms optimal codes to other optimal codes (Lemma 5.3.2), neither of which is necessarily a Huffman code. It turns out not to be true that any optimal prefix code can be obtained from any other optimal prefix code by a sequence of same-parent and same-probability node swaps (see H_2 and C in Example 5.1.4). We describe another node swap involving cutting branches of a Huffman tree at two different nodes on the same row and then exchanging the hanging subtrees (i.e., swapping same-row nodes). This operation preserves the average length of any prefix code, so it maps optimal prefix codes to optimal prefix codes, but the operation need not transform Huffman codes into other Huffman codes. (see H_1 and H_2 in Example 5.1.4).

We show that for a given source, two complete prefix codes are length equivalent if and only if they are same-row swap equivalent (Theorem 5.3.6). This then implies that any prefix code for a given source is length equivalent to a particular Huffman code (and thus is optimal) if and only if its code tree can be obtained from the Huffman tree by a sequence of same-row node swaps (Corollary 5.3.7).

If we replace same-parent node swaps by the more general same-row node swaps, then we can characterize optimal prefix codes in another way by using node swapping. Specifically, we show that all optimal prefix codes are same-row, same-probability swap equivalent to each other (Theorem 5.3.9).

5.2 Characterization of expected length minimizing prefix codes

In this section we give a new characterization of optimal prefix codes for a given source. While all Huffman codes are optimal and were characterized by Gallager in terms of the sibling property, not all optimal codes are Huffman codes. However, it turns out that any optimal code is length equivalent to some Huffman code for the source as shown in Theorem 5.2.4.

In Theorem 5.2.4 we also prove a second characterization of optimal prefix codes. Specifically, we show that a prefix code is optimal if and only if it is complete and strongly monotone. The combination of completeness and strong monotonicity is weaker than the sibling property, and thus a broader class of prefix codes (namely, the

optimal ones) satisfies this combination.

Definition 5.2.1. Given a source with alphabet S, a prefix code C is *strongly monotone* if $P(A) \geq P(B)$ whenever $A, B \subseteq S$ and $K_C(A) = 2^{-i} > 2^{-j} = K_C(B)$ for some integers i and , j.

The strongly monotone property reduces to Gallager's monotone property when each of A and B consists of all leaf descendants of a single tree node. Example 5.1.3 illustrates that these two properties are not equivalent. Specifically, the example shows that prefix code C is not strongly monotone because $K_C(\{c, d\}) = 2^{-1} > 2^{-2} = K_C(\{a\})$ but $P(\{c, d\}) = \frac{1}{4} < \frac{3}{8} = P(\{a\})$. The code H is strongly monotone since it is a Huffman code.

The following lemma easily follows from the proof of Lemma 5.1.5. This lemma relies on our definition of sources (and thus codes) to be finite. Prefix codes for infinite sources need not satisfy the lemma below (e.g. [15, p. 2027]).

Lemma 5.2.2. *A prefix code is complete if and only if its Kraft sum equals* 1.

Lemma 5.2.3. *If two prefix codes are length equivalent, then each of the following properties holds for one code if and only if it holds for the other code:*

(1) completeness

(2) strong monotonicity

(3) optimality.

Proof. Let S be the source alphabet. Let C and C' be length equivalent prefix codes, i.e., $l_C(y) = l_{C'}(y)$ for all $y \in S$. Then for all $y \in S$,

$$K_C(y) = 2^{-l_C(y)} = 2^{-l_{C'}(y)} = K_{C'}(y).$$

Since

$$\sum_{y \in S} K_C(y) = \sum_{y \in S} K_{C'}(y),$$

Lemma 5.2.2 shows C is complete if and only if C' is complete.

Suppose C is strongly monotone. Let $A, B \subseteq S$ with $K_{C'}(A) = 2^{-i}$ and $K_{C'}(B) = 2^{-j}$ for some integers $0 \leq i < j$. Since $K_C(A) = K_{C'}(A) = 2^{-i}$ and $K_C(B) = K_{C'}(B) = 2^{-j}$, we have $P(A) \geq P(B)$ since C is strongly monotone. Thus C' is also strongly monotone.

Let X be a source random variable. The average length of code C is

$$E[l_C(X)] = \sum_{y \in S} P(y) l_C(y) = \sum_{y \in S} P(y) l_{C'}(y) = E[l_{C'}(X)],$$

so C is optimal if and only if C' is. ■

The following theorem is our first main result.

Theorem 5.2.4. *For a given source, if C is a prefix code, then the following are equivalent:*

(1) C is complete and strongly monotone;

(2) C is length equivalent to a Huffman code; and

(3) C is optimal.

Proof. Let S be the source alphabet, and let X be the source random variable on S. Define $P(u) = P(X = u)$ for all $u \in S$.

$(1) \Longrightarrow (2)$

Suppose C is complete and strongly monotone. Consider the following operation on the code tree for C. Suppose row $k \geq 1$ of the code tree has the property that all non-leaves are listed in order of non-increasing probability moving left-to-right in the row. Permute all nodes in row k such that they are listed in order of non-increasing probability moving left-to-right in the row. Note that the non-leaves remain in the same order among themselves as they were prior to performing the operation, and therefore all nodes in each row $m > k$ remain in the same order in their row as they were prior to performing the operation. Let C' be the new code corresponding to the code tree obtained after this operation. This operation may change the codewords assigned to symbols in S, but it does not change the lengths of any codewords, and so C and C' are length equivalent. Moreover, once this operation has been performed, the probabilities of the parents of the nodes in row k are listed in order of non-increasing probability moving left-to-right in row $(k-1)$.

Therefore, we can apply this node permutation operation iteratively on successively higher rows, beginning on the bottom row of the code tree for C where there are only leaves, and then moving upward, row-by-row. Once the leaves in the bottom row have been ordered, the probabilities of their parents have been ordered, and we can then proceed by induction to conclude that after this operation has been performed on each row, the nodes in each row of the resulting code tree are listed in order of non-increasing probability moving left-to-right in the row. Let C' be the code corresponding to the resulting code tree. Inductively, based on the argument in the previous

paragraph, C' is length equivalent to C, and so C' is complete and strongly monotone by Lemma 5.2.3. Consider the sequence of probabilities of nodes in the code tree for C' listed in raster-scan order, beginning with the root node and proceeding downward, row-by-row, moving left-to-right in each row. Then the probability of each node is adjacent in the sequence to the probability of its sibling. Also, as concluded previously, all probabilities of nodes in the same row are listed in non-increasing order in the sequence. Furthermore, since C' is strongly monotone, the rightmost node u in a given row $k \geq 0$ and the leftmost node v in row $(k+1)$ satisfy $P(u) \geq P(v)$, since $K_{C'}(u) = 2^{-k} > 2^{-(k+1)} = K_{C'}(v)$. Therefore, the sequence of probabilities is listed in order of non-increasing probability, and so the code tree for C' satisfies the sibling property since C' is also complete. Thus, by Lemma 5.1.8, C' is a Huffman code. Since C is length equivalent to C', we are done.

$(2) \Longrightarrow (3)$

Suppose C is length equivalent to a Huffman code H. Then by Lemma 5.2.3, C is optimal because H is.

$(3) \Longrightarrow (1)$

Suppose C is optimal. If C is not complete, then there exists a non-root node u without a sibling in the code tree for C. Replacing the parent of u by u itself results in a code tree where the lengths of all leaf descendants of u have been decreased by 1, and the lengths of all other leaves have remained unchanged. Therefore, the expected length of the new code is $P(u)$ less than the expected length of C. Since $P(u) > 0$, this shows C is not optimal, which is a contradiction. Thus C is complete.

Suppose C is not strongly monotone. Then there exist $A, B \subseteq S$ and integers i and j, such that $0 \leq i < j$, $K_C(A) = 2^{-i}$, $K_C(B) = 2^{-j}$, and $P(A) < P(B)$. Denote the symmetric difference of A and B by $A \Delta B = (A - B) \cup (B - A)$. Then

$$\begin{aligned} K_C(A - B) &= K_C(A) - K_C(A \cap B) = 2^{-i} - K_C(A \cap B) \\ K_C(B - A) &= K_C(B) - K_C(A \cap B) = 2^{-j} - K_C(A \cap B) \\ K_C(S - (A \Delta B)) &= 1 - K_C(A - B) - K_C(B - A) \\ &= 1 - 2^{-i} - 2^{-j} + 2K_C(A \cap B) \end{aligned}$$

and

$$P(A - B) = P(A) - P(A \cap B) < P(B) - P(A \cap B) = P(B - A). \tag{5.1}$$

Let C' be a prefix code such that:

$$l_{C'}(y) = l_C(y) + \begin{cases} j-i & \text{if } y \in A - B \\ i-j & \text{if } y \in B - A \\ 0 & \text{if } y \in S - (A \Delta B). \end{cases} \tag{5.2}$$

Note that such a prefix code exists by Lemma 5.1.5, since

$$\begin{aligned}
&\sum_{y \in S} 2^{-l_{C'}(y)} \\
&= 2^{-(j-i)} \sum_{y \in A-B} 2^{-l_C(y)} + 2^{j-i} \sum_{y \in B-A} 2^{-l_C(y)} + \sum_{y \in S-(A \Delta B)} 2^{-l_C(y)} && (5.3) \\
&= 2^{-(j-i)} K_C(A - B) + 2^{j-i} K_C(B - A) + K_C(S - (A \Delta B)) \\
&= 2^{-j} - 2^{-(j-i)} K_C(A \cap B) + 2^{-i} - 2^{j-i} K_C(A \cap B) + 1 - 2^{-i} - 2^{-j} + 2K_C(A \cap B) && (5.4) \\
&= 1 + (2 - 2^{j-i} - 2^{-(j-i)}) K_C(A \cap B) \\
&\leq 1 && (5.5)
\end{aligned}$$

where (5.3) follows from (5.2); (5.4) follows from $K_C(A) = 2^{-i}$ and $K_C(B) = 2^{-j}$; and (5.5) follows from $j > i$. Equality holds in (5.5) if and only if $K_C(A \cap B) = 0$, since $2^{j-i} \geq 2$. Finally,

$$\begin{aligned}
E[l_{C'}(X)] &= \sum_{y \in A-B} P(y) l_{C'}(y) + \sum_{y \in B-A} P(y) l_{C'}(y) + \sum_{y \in S-(A \Delta B)} P(y) l_{C'}(y) \\
&= \sum_{y \in A-B} P(y)(l_C(y) + (j-i)) + \sum_{y \in B-A} P(y)(l_C(y) - (j-i)) + \sum_{y \in S-(A \Delta B)} P(y) l_C(y) \\
&= (j-i)(P(A - B) - P(B - A)) + E[l_C(X)] \\
&< E[l_C(X)], && (5.6)
\end{aligned}$$

where (5.6) follows from $j - i \geq 1$ and $P(A - B) < P(B - A)$ by (5.1). But then C is not optimal, which is a contradiction. Therefore, C is strongly monotone. ■

We note that a proof of $(3) \implies (2)$ in Theorem 5.2.4 does not seem to have previously appeared explicitly in the literature, although it may be hinted at in the proof of [5, Lemma 5.8.1, p. 123] and also in 1995 by Yamamoto and Itoh [18].

The following corollary immediately follows from Theorem 5.2.4 and describes which codes perform as

well as Huffman codes in terms of average length.

Corollary 5.2.5. *For a given source, a complete prefix code C is optimal if and only if C is strongly monotone.*

5.3 Swapping code tree nodes

Definition 5.3.1. A *node swap* in a code tree is an exchange of the subtrees rooted at two distinct nodes neither of which is a descendant of the other. Such a node swap is called a *same-row* (respectively, *same-probability*) node swap if the two nodes are in the same row (respectively, have the same probability). A *same-parent* node swap is a same-row node swap of two siblings. Prefix codes C_1 and C_2 are *same-parent swap equivalent* (respectively, *same-row swap equivalent*, *same-probability swap equivalent*) if the code tree of C_1 can be transformed into the code tree of C_2 by a sequence of same-parent (respectively, same-row, same-probability) node swaps. Additionally, prefix codes C_1 and C_2 are *same-parent, same-probability swap equivalent* (respectively, *same-row, same-probability swap equivalent*) if the code tree of C_1 can be transformed into the code tree of C_2 by a sequence of same-parent (respectively, same-row) and/or same-probability node swaps.

The relations of same-parent, same-row, and same-probability swap equivalence are all reflexive, symmetric, and transitive, and thus are equivalence relations. Therefore, they each naturally induce equivalence classes of prefix codes.

For a given source, any same-parent node swap transforms a Huffman tree to another Huffman tree, since in the Huffman construction process when two nodes are merged the choice of their order in a sibling pair is arbitrary.

The following two lemmas are straightforward, so we omit their proofs.

Lemma 5.3.2. *For a given source, same-parent, same-row, and same-probability node swaps preserve the expected length of any complete prefix code.*

Lemma 5.3.3. *For a given source, any same-parent node swap of a Huffman code produces another Huffman code.*

Since each merging in a Huffman tree construction has two possible orderings, and there are $n-1$ internal nodes in a binary tree with n leaves, there are 2^{n-1} different Huffman codes in each same-parent swap equivalence class that contains at least one Huffman code. As shown in Lemma 5.3.3, any Huffman code in such an equivalence class can be obtained from another Huffman code in the class by performing a sequence of same-parent node swaps. However, two Huffman codes for a given source need not be related by a sequence of same-parent node swaps (i.e., they can be in different same-parent swap equivalence classes).

The next lemma shows that same-probability node swaps convert Huffman trees to other Huffman trees for the same source.

Lemma 5.3.4. *For a given source, any same-probability node swap of a Huffman code produces another Huffman code.*

Proof. By Lemma 5.1.8, the Huffman tree H satisfies the sibling property. That is, there exists a sequence $\sigma_H = u_1, u_2, \ldots, u_i, \ldots, u_j, \ldots$ of the nodes of H in non-increasing order of their probabilities where siblings appear adjacent in the list.

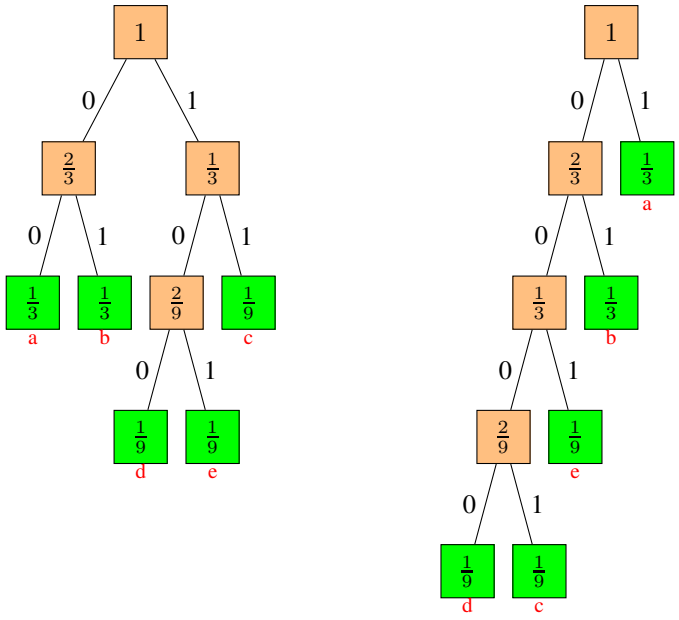
Suppose nodes u_i and u_j are swapped in Huffman tree H to give code tree C, where $P(u_i) = P(u_j)$. Swapping nodes modifies the tree H by altering two edges, and retaining all the same nodes and the other edges.

The nodes of the new code tree C are the same as the nodes of H. Modify the sequence of nodes of H by exchanging u_i with u_j to obtain the sequence $\sigma_C = u_1, u_2, \ldots, u_j, \ldots, u_i, \ldots$, which is a sequence of the nodes of C.

All of the probabilities of the nodes in C remain the same as they were in H (since the parent nodes of u_i and u_j in C have children with the same probabilities as they had in H), so the probabilities of the nodes in the modified sequence σ_C are in non-increasing order. Also, u_i and u_j appear adjacent in σ_C to their siblings, since u_j and u_i, respectively, do so in σ_H. Thus, the second tree satisfies the sibling property, and so is a Huffman tree by Lemma 5.1.8. ■

Note that if a Huffman tree node x lies at least two rows above node y, then Lemma 5.1.6 implies the probability of x is at least as great as the probability of the parent of y, which is strictly greater than the probability of y, so x and y cannot have the same probability. Therefore, in a Huffman tree, any two nodes with the same probability must either lie in the same row or in adjacent rows. This means that any same-probability node swap in a Huffman tree consists of swapping two nodes in the same row or adjacent rows. However, sequences of same-probability node swaps can transform one Huffman tree into another where a node can move farther than one row away, as the following example illustrates.

Example 5.3.5 (Two Huffman trees with a source symbol two rows apart)**.**
Let H_1 and H_2 be two different Huffman codes for a source with symbols a,b,c,d,e and probabilities $\frac{1}{3}$, $\frac{1}{3}$, $\frac{1}{9}$, $\frac{1}{9}$, $\frac{1}{9}$, respectively. The two trees are shown in Figure 5.6. The tree H_1 can be transformed into H_2 by performing a same-probability node swap between leaf a and the parent of leaf c, and subsequently performing a same-probability node swap between leaf c and leaf e. Note that the source symbol c appears in row 2 of H_1 and row 4 of H_2. In fact, one can construct examples where the same source symbol appears in two different Huffman trees in arbitrarily distant rows.



Huffman code H_1 Huffman code H_2

Figure 5.6. Two Huffman trees for the same source with source symbol c appearing on rows differing in level by two.

By Lemma 5.3.3 and Lemma 5.3.4 any sequence of same-parent node swaps and same-probability node swaps converts Huffman codes to Huffman codes for the same source. Thus, the set of all codes obtained by such transformations on a given Huffman code is an equivalence class containing Huffman codes only. In fact, this equivalence class contains all Huffman codes for a given source (Theorem 5.3.8).

Same-row node swaps preserve codeword lengths assigned to source symbols. If a same-row node swap is not a same-parent node swap, then it may convert a Huffman code into a non-Huffman code (e.g., the prefix code C in Figure 5.3). However, these non-Huffman codes are still optimal (Corollary 5.3.7).

The following lemma shows that length equivalence between two complete prefix codes is the same as being able to transform one into the other using only same-row node swaps.

Theorem 5.3.6. *For a given source, two complete prefix codes are length equivalent if and only if they are same-row swap equivalent.*

Proof. Same-row node swaps preserve the lengths of all codewords, so length equivalence follows immediately.

Conversely, suppose C_1 and C_2 are code trees for two length equivalent prefix codes. For any complete code tree for the source, assign labels to its nodes as follows. Label each leaf node by the source symbol of its associated codeword, and label each parent node as the ordered pair (a, b), where a is the label of its left child and b is the label of its right child.

Let d be the common depth (i.e., longest codeword length) of C_1 and C_2. All nodes in row d in both C_1 and C_2 are leaves, and since C_1 and C_2 are length equivalent, the set of leaf labels in row d is identical for C_1 and C_2.

We will show that if the set of node labels in some row $m \in \{1, \ldots, d\}$ is identical for C_1 and C_2, then there exists a sequence of same-row node swaps that transforms C_1 into a code tree C_1' such that the set of node labels in row $m-1$ is identical for C_1' and C_2. Then, since we have already shown the set of node labels in the lowest row d is identical for C_1 and C_2, we conclude by induction that there exists a sequence of same-row node swaps that transforms C_1 into a code tree C_1^* whose root has the same label as the root of C_2. The label of the root node of a code tree specifies the code tree exactly (it's a parenthetically nested list of node pairs used to form the tree), so C_1^* and C_2 are the same code tree, and we conclude C_1 and C_2 are same-row swap equivalent.

Suppose the set of node labels in some row $m \in \{1, \ldots, d\}$ is identical for C_1 and C_2. In other words, the same node labels appear in row m of each code tree, but the node labels are possibly arranged in a different order. Then there exists a permutation that maps the arrangement of node labels in row m of C_1 to the arrangement of node labels in row m of C_2. By a standard group theory result that any finite permutation can be obtained from any other permutation by a sequence of transpositions (e.g., see [6, p. 107]), there exists a sequence of same-row node swaps that achieves this permutation and transforms C_1 into a code tree C_1' whose sequence of node labels in row m is identical to that of C_2. Since sibling nodes are adjacent in this sequence, the set of labels of the parent nodes in row $m-1$ is identical in C_1' and C_2.

Since C_1 and C_2 are length equivalent, the set of leaf node labels in row $m-1$ is identical in C_1 and C_2, and the same is true for C_1' and C_2 since the leaf nodes in row $m-1$ of C_1 were unaffected by the same-row node swaps performed on the nodes in row m. Therefore, the set of node labels in row $m-1$ is identical for C_1' and C_2, and the induction argument is complete. ■

The following theorem shows that prefix code optimality is equivalent to being able to transform the code into some Huffman code using only same-row node swaps.

Corollary 5.3.7. *For a given source, a prefix code is optimal if and only if it is same-row swap equivalent to a Huffman code.*

Proof. It follows immediately from Theorem 5.3.6 and Theorem 5.2.4. ■

Corollary 5.3.7 guarantees for any optimal prefix code the existence of some Huffman code that is same-row swap equivalent. However, different optimal prefix codes need not all be same-row swap equivalent to the same Huffman code. The next theorem indicates, however, that all Huffman codes can be transformed to each

other using only same-parent, same-probability node swap operations. Then, by combining Corollary 5.3.7 and Theorem 5.3.8, we obtain Theorem 5.3.9, which states that all optimal prefix codes are same-row, same-probability swap equivalent.

Theorem 5.3.8. *For a given source, all Huffman codes are same-parent, same-probability swap equivalent to each other.*

Proof. For any complete code tree for the source, assign labels to its nodes as follows. Label each leaf node by the source symbol of its associated codeword, and label each parent node as the ordered pair (a, b), where a is the label of its left child and b is the label of its right child. Given a node label u, let $P(u)$ be the probability of the node labeled u.

Let H_1 and H_2 be two Huffman code trees for a source. Then in particular, H_1 and H_2 are complete by Lemma 5.1.8. Since H_1 and H_2 are code trees for the same source, the set of leaf node labels is identical in H_1 and H_2.

Let $(a_1, b_1), (a_2, b_2), \ldots$ and $(a'_1, b'_1), (a'_2, b'_2), \ldots$ be the sequences of the parent node labels (i.e., pairs of merged child labels) in H_1 and H_2, respectively, listed in the order of each step of the Huffman construction that produced each code tree.

Let i be the smallest index such that $(a_i, b_i) \neq (a'_i, b'_i)$. The set of node labels available for merging during step i is identical in the Huffman construction of H_1 and H_2, since such a node label is either a leaf node label common to both H_1 and H_2, or is a parent node label constructed in a step prior to i. Then in particular, all the node labels in $\{a_i, b_i, a'_i, b'_i\}$ are available for merging at step i in the Huffman construction of both H_1 and H_2. Also, since the Huffman algorithm combines at each step two nodes of smallest probability, we have $\{P(a_i), P(b_i)\} = \{P(a'_i), P(b'_i)\}$.

If $a'_i \notin \{a_i, b_i\}$, then there exists $u_i \in \{a_i, b_i\}$ such that $P(u_i) = P(a'_i)$. Since both u_i and a'_i appear in H_1, we can remove the edges connecting u_i and a'_i to their parent nodes in H_1, and add edges connecting u_i and a'_i to the parent nodes of a'_i and u_i, respectively. Since $P(u_i) = P(a'_i)$, this operation is a same-probability node swap, and by Lemma 5.3.4 the resulting code tree is a Huffman code tree.

After performing this same-probability node swap if necessary, we arrive at a Huffman code tree that combines a'_i and $v_i \in \{a_i, b_i\}$ in step i, where $P(v_i) = P(b'_i)$. If $v_i \neq b'_i$, then an analogous argument as above shows a same-probability node swap can be performed on v_i and b'_i to produce a new Huffman code tree.

After performing this same-probability node swap if necessary, we arrive at a Huffman code tree that combines a'_i and b'_i in step i. If necessary, perform a same-parent node swap on the sibling pair $\{a'_i, b'_i\}$ such that their parent label is (a'_i, b'_i), and name this code tree H'_1, which is a Huffman code tree by Lemma 5.3.3. This

same-parent node swap and the same-probability node swaps described previously leave the labels of parent nodes in H_1 obtained in any steps prior to i unaffected, and as a consequence, the Huffman constructions of H_1' and H_2 produce identical parent node labels at each step up to and including step i.

By induction on i, there exists a sequence of same-parent node swaps and same-probability node swaps that transforms H_1 into a Huffman code tree H_1^*, such that H_1^* and H_2 produce identical parent node labels at every step of the Huffman construction. In particular, the root of H_1^* has the same label as the root of H_2. The label of the root node of a code tree specifies the code tree exactly, so H_1^* and H_2 are the same code tree, and the lemma is proved. ■

Theorem 5.3.9. *For a given source, all optimal prefix codes are same-row, same-probability swap equivalent to each other.*

Proof. Let C_1 and C_2 be optimal prefix codes. By Corollary 5.3.7, C_1 is same-row swap equivalent to some Huffman code H_1, and C_2 is same-row swap equivalent to some Huffman code H_2. Then by Theorem 5.3.8, H_1 is same-parent, same-probability swap equivalent to H_2. Therefore, C_1 and C_2 are same-row, same-probability swap equivalent to each other. ■

Chapter 5 is a reprint of the material as it appears in: S. Congero and K. Zeger, "Characterizations of minimal expected length codes", submitted to *IEEE Transactions on Information Theory*, November 13, 2023.

References

[1] J. Abrahams, "Code and parse trees for lossless source encoding", *Proceedings of the Compression and Complexity of Sequences*, Salerno, Italy, pp. 145 – 171, June 1997.

[2] L. Cao, "Self-synchronization strings in Huffman equivalent codes", *IEEE Information Theory Workshop*, Chengdu, China, pp. 347 – 350, October 2006.

[3] L. Cao, L. Yao, C. W. Chen, "MAP decoding of variable length codes with self-synchronization strings", *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4325 – 4330, September 2007.

[4] S. Congero and K. Zeger, "Competitive advantage of Huffman and Shannon-Fano codes", *IEEE Transactions on Information Theory* (submitted November 13, 2023). Also available on: arXiv:2311.07009 [cs.IT].

[5] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd edition, New Jersey, Wiley-Interscience, 2006.

[6] D. S. Dummit and R. M. Foote, *Abstract Algebra*, 3rd edition, John Wiley & Sons, Inc., 2004.

[7] A. E. Escott and S. Perkins, "Binary Huffman equivalent codes with a short synchronizing codeword", *IEEE Transactions on Information Theory*, vol. 44, no. 1, pp. 346 – 351, January 1998.

[8] T. Ferguson and J. Rabinowitz, "Self-synchronizing Huffman codes", *IEEE Transactions on Information Theory*, vol. 30, no. 4, pp. 687 –6 93, July 1984.

[9] R. G. Gallager, "Variations on a theme by Huffman", *IEEE Transactions on Information Theory*, vol. 24, no. 6, pp. 668 – 8674, November 1978.

[10] M. Higgs, "The construction of variable length codes with good synchronisation properties", *Doctoral thesis*, University of South Wales Prifysgol De Cymru, February 26, 2007.

[11] M. B. J. Higgs, S. Perkins, and D. H. Smith, "The construction of variable length codes with good synchronization properties", *IEEE Transactions on Information Theory*, vol. 55, no. 4, pp. 1696 – 1700, April 2009.

[12] Y.-M. Huang and S.-C. Wu, "Shortest synchronizing codewords of a binary Huffman equivalent code", *International Conference on Information Technology: Coding and Computing*, Las Vegas, NV, pp. 226 – 231, April 2003.

[13] D. A. Huffman, "A method for the construction of minimum-redundancy codes", *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098 – 1101, September 1952.

[14] Y.-S. Lee, W.-S. Chang, H.-H. Ho, C.-Y. Lee, "Construction of error resilient synchronization codeword for variable-length code in image transmission", *International Conference on Image Processing*, vol. 3, pp. 360 – 363, September 2000.

[15] T. Linder, V. Tarokh, and K. Zeger, "Existence of optimal codes for infinite source alphabets", *IEEE Transactions on Information Theory*, vol. 43, no. 6, pp. 2026 – 2028, November 1997.

[16] G. Longo and G. Galasso, "An application of informational divergence to Huffman codes", *IEEE Transactions on Information Theory*, vol. 28, no. 1, pp. 36 – 43, January 1982,

[17] D. Wang, X. Zhao, Q. Sun, "Novel fault-tolerant decompression method of corrupted Huffman files", *Wireless Personal Communications*, vol. 102, no. 4, pp. 2555 – 2574, October 2018.

[18] H. Yamamoto and T. Itoh, "Competitive optimality of source codes", *IEEE Transactions on Information Theory*, vol. 41, no. 6, pp. 2015 – 2019, November 1995.

[19] J. Zhou, O. C. Au, "Error recovery of variable length code over BSC with arbitrary crossover probability", *IEEE Transactions on Communications*, vol. 58, no. 6, pp. 1654 – 1666, June 2010.

[20] G. Zhou and Z. Zhang, "synchronization recovery of variable-length codes", *IEEE Transactions on Information Theory*, vol. 48, no. 1, pp. 219 – 227, February 2002.

Chapter 6

Competitive Advantage of Huffman and Shannon-Fano Codes

Abstract

For any finite discrete source, the competitive advantage of prefix code C_1 over prefix code C_2 is the probability C_1 produces a shorter codeword than C_2, minus the probability C_2 produces a shorter codeword than C_1. For any source, a prefix code is competitively optimal if it has a nonnegative competitive advantage over all other prefix codes. In 1991, Cover proved that Huffman codes are competitively optimal for all dyadic sources. We prove the following asymptotic converse: As the source size grows, the probability a Huffman code for a randomly chosen non-dyadic source is competitively optimal converges to zero. We also prove: (i) For any source, competitively optimal codes cannot exist unless a Huffman code is competitively optimal; (ii) For any non-dyadic source, a Huffman code has a positive competitive advantage over a Shannon-Fano code; (iii) For any source, the competitive advantage of any prefix code over a Huffman code is strictly less than $\frac{1}{3}$; (iv) For each integer $n > 3$, there exists a source of size n and some prefix code whose competitive advantage over a Huffman code is arbitrarily close to $\frac{1}{3}$; and (v) For each positive integer n, there exists a source of size n and some prefix code whose competitive advantage over a Shannon-Fano code becomes arbitrarily close to 1 as $n \longrightarrow \infty$.

6.1 Introduction

In a probabilistic game where multiple players are each rated by a numerical score, one way to designate a particular player A as being superior among their fellow competitors is by "expected score optimality", where no other player B can obtain a better score on average than the score of A. A second way, called "competitive optimality", occurs if for every other player B, the probability of A scoring better than B is at least the probability of B scoring better than A. That is, A has a nonnegative "competitive advantage"

$$P(A \text{ scores better than } B) - P(B \text{ scores better than } A)$$

over all other players B (where tie scores are ignored). In this paper we obtain results about competitive advantage and optimality when lossless source coding is viewed as a game, source codes are the players, the numerical score is the length of the codeword of a randomly chosen source symbol, and a codeword length is deemed to be better than another if its length is shorter. We first formalize some terminology and definitions and then explain the history of the problem and our results.

An *alphabet* is a finite set S, and a *source* of size n with alphabet S is a random variable X such that $|S| = n$ and $P(X = y) = P(y)$ for all $y \in S$. We denote the probability of any subset $B \subseteq S$ by $P(B) = \sum_{y \in B} P(y)$. A source is said to be *dyadic* if $P(y)$ is a nonnegative integer power of $1/2$ for all $y \in S$.

A *code* for a given source X is a mapping $C : S \longrightarrow \{0,1\}^*$ and the binary strings $C(1), \ldots, C(n)$ are called *codewords* of C. A *prefix code* is a code where no codeword is a prefix of any other codeword.

A *code tree* for a prefix code C is a rooted binary tree whose leaves correspond to the codewords of C; specifically, the codeword associated with each leaf is the binary word denoting the path from the root to the leaf. The *length* of a code tree node is its path length from the root. The rth *row* of a code tree is the set of nodes whose length is r, and we will view a code tree's root as being on the top of the tree with the tree growing downward. For example, row r of a code tree is "higher" in the tree than row $r + 1$. If x and y are nodes in a code tree, then x is a *descendent* of y if there is a downward path of length zero or more from y to x. Two nodes in a tree are called *siblings* if they have the same parent. In a code tree, for any collection A of nodes having no common leaf descendents, define $P(A)$ to be the probability of the set of all leaf descendents of A in the tree.

A (binary) *Huffman tree* is a code tree constructed from a source by recursively combining two smallest-probability nodes until only one node with probability 1 remains. The initial source probabilities correspond to leaf nodes in the tree. A *Huffman code* for a given source is a mapping of source symbols to binary words by assigning the source symbol corresponding to each leaf in the Huffman tree to the binary word describing the path from the

root to that leaf. A *Shannon-Fano code* is a prefix code, such that for each $y \in S$ the codeword associated with the source symbol y has length $\lceil \log_2 \frac{1}{P(y)} \rceil$.

Given a source with alphabet S and a prefix code C, for each $y \in S$ the length of the binary codeword $C(y)$ is denoted $l_C(y)$. Two codes C_1 and C_2 are *length equivalent* if $l_{C_1}(y) = l_{C_2}(y)$ for every source symbol $y \in S$. The *average length* of a code C for a source with alphabet S is $\sum_{y \in S} l_C(y)P(y)$. A prefix code is *expected length optimal* for a given source if no other prefix code achieves a smaller average codeword length for the source.

A code is *complete* if every non-root node in its code tree has a sibling, or, equivalently, if every node has either zero or two children. A code C for a given source is *monotone* if for any two nodes in the code tree of C, we have $P(u) \geq P(v)$ whenever $l_C(u) < l_C(v)$. Expected length optimal codes are always monotone (see Lemma 6.2.3).

Huffman codes are known to be expected length optimal, monotone, and complete for every source (e.g., see [7]), whereas Shannon-Fano codes are monotone, but need not be expected length optimal nor complete. Shannon-Fano codes are known to achieve the same average lengths as Huffman codes whenever the source is dyadic. For non-dyadic sources, Shannon-Fano codes always have larger average lengths than Huffman codes, but nevertheless the average length of the Shannon-Fano code (and thereby also the Huffman code) is less than one bit larger than the source entropy [7].

The *Kraft sum* of a sequence of nonnegative integers $l_1, \ldots, l_k$ is $2^{-l_1} + \cdots + 2^{-l_k}$. We extend the definition of "Kraft sum" to also apply to sets of source symbols or sets of nodes in a code tree as follows. The Kraft sum $K(A)$ of any collection A of leaves is the Kraft sum of the corresponding sequence of codeword lengths. The Kraft sum $K(A)$ of any collection A of nodes having no common leaf descendents is the Kraft sum of the set of all leaf descendents of A in the tree. The Kraft sum of a collection of source symbols is the Kraft sum of the corresponding leaves in a code tree. The well-known Kraft inequality and its converse are stated next.

Lemma 6.1.1 (Kraft, e.g., [7, Theorem 5.2.1])**.** *The codeword lengths $l_1, \ldots, l_n$ of any prefix code satisfy $2^{-l_1} + \cdots + 2^{-l_n} \leq 1$. Conversely, if a sequence $l_1, \ldots, l_n$ of positive integers satisfies $2^{-l_1} + \cdots + 2^{-l_n} \leq 1$, then there exists a binary prefix code whose codeword lengths are $l_1, \ldots, l_n$.*

The following lemma[1] expresses an equality condition for Lemma 6.1.1.

Lemma 6.1.2. *A prefix code is complete if and only if its Kraft sum equals* 1.

We have defined sources and prefix codes to be finite throughout this paper, but we note that a complete infinite prefix code need not have Kraft sum 1 (e.g., see [12]).

[1]The proof follows easily from the proof of Theorem 5.1.1 in the Cover-Thomas textbook [7]. A more general result can be found in Theorem 2.5.19 of the Berstel-Perrin-Reutenauer textbook [2].

For a subset A of a source's alphabet and for a Huffman code H for the source, we say the *Huffman-Kraft sum* of A is

$$K(A) = \sum_{x \in A} 2^{-l_H(x)},$$

where we reuse the "K" notation. Note that the Huffman-Kraft sum of A is the (usual) Kraft sum of the Huffman codeword lengths of the symbols in A.

As previously mentioned, one measure of the success of a source code that generally differs from expected length optimality is called "competitive optimality" and has been considered in the form of one-on-one competitions between pairs of prefix codes to determine in each competition which code has the higher probability of producing a shorter codeword for a given source.

For a given source with alphabet S, and for any codes C_1 and C_2, define

$$\begin{aligned} W &= \{i \in S : l_{C_1}(i) < l_{C_2}(i)\} \\ L &= \{i \in S : l_{C_1}(i) > l_{C_2}(i)\} \\ T &= \{i \in S : l_{C_1}(i) = l_{C_2}(i)\}. \end{aligned} \tag{6.1}$$

The sets W, L, and T contain the source values of the *wins*, *losses*, and *ties*, respectively, for code C_1 in a one-on-one competition against code C_2 to see which produces shorter length codewords for the given source. Even though the notation for W, L, and T does not explicitly reference C_1 and C_2, the codes involved will be clear from context. Code C_1 is said to *competitively dominate* code C_2 if $P(W) \geq P(L)$, and *strictly competitively dominate* code C_2 if $P(W) > P(L)$. A prefix code is *competitively optimal* for a source of size n if it competitively dominates all other prefix codes for the same source. The notion of competitive optimality dates back at least to 1980 in the field of financial investment [1].

In 1991, Cover proved that Shannon-Fano codes are competitively optimal for dyadic sources. Since Huffman and Shannon-Fano codes are length equivalent for dyadic sources (via [7, Theorem 5.3.1]), Huffman codes are also competitively optimal in this case, as reworded below.

Theorem 6.1.3 (Cover [6, Theorem 2])**.** *Huffman codes are competitively optimal for all dyadic sources.*

Cover's proof also showed that for all dyadic sources the Huffman code strictly competitively dominates all other (i.e., not length equivalent) prefix codes. Additionally, he showed that for all non-dyadic sources Shannon-Fano codes competitively dominate all other prefix codes if an extra one-bit penalty is assessed to the non-Shannon-Fano code during each symbol encoding. This comparison favorably treats ties as if they were wins for the Shannon-

Fano code and treats one-bit losses for the Shannon-Fano code as if they were ties, thus giving the Shannon-Fano code a one-bit handicap in betting parlance. As dyadic sources are rare among all sources, it is natural to ask whether Cover's (non-handicapped) competitive optimality result extends in some way to non-dyadic sources.

In 1992, Feder [9] showed that for all non-dyadic sources Huffman codes competitively dominate all other prefix codes if an extra one-bit penalty is assessed to the non-Huffman code during each symbol encoding. This result is analogous to (but uses a different proof technique from) Cover's result for Shannon-Fano codes.

In 1995, Yamamoto and Itoh [16] illustrated a non-dyadic source whose Huffman code was not competitively optimal. They presented a prefix code for the source which strictly competitively dominates the Huffman code, and whose win and loss probabilities are $P(W) = 0.5$ and $P(L) = 0.4$, respectively. Their example consists of a source of size 4 and distinct prefix codes C_1, C_2, and C_3, such that C_i competitively dominates C_j whenever $(i, j) \in \{(1, 2), (2, 3), (3, 1)\}$. This example demonstrates that the relation of competitive dominance is not transitive. One of these codes was the Huffman code, so the Huffman code is not competitively optimal, and no competitively optimal code exists in this case. They also provided a sufficient condition for a source not to have a competitively optimal Huffman code.

Their results, however, do not provide an indication of how many or few source codes would have competitively optimal Huffman codes.

In 2001, Yamamoto and Yokoo [17] studied competitive optimality for almost instantaneous variable-to-fixed length codes.

In 2021, Bhatnagar [3] studied the use of competitive optimality for analyzing error probabilities in artificial intelligence systems.

The Cover-Thomas textbook [7, p. 131] gives the following gambling interpretation of competitive optimality and acknowledges the difficulty of mathematically analyzing Huffman codes:

> "To formalize the question of competitive optimality, consider the following two-person zero-sum game: Two people are given a probability distribution and are asked to design an instantaneous code for the distribution. Then a source symbol is drawn from this distribution, and the payoff to player A is 1 or -1, depending on whether the codeword of player A is shorter or longer than the codeword of player B. The payoff is 0 for ties.
>
> Dealing with Huffman code lengths is difficult, since there is no explicit expression for the codeword lengths."

To determine whether a code is competitively optimal, one must determine if the difference between the probabilities of winning and losing a competition for shortest codeword length is nonnegative for every possible opponent code. When a code C_1 is determined not to be competitively optimal, at least one other code C_2 has a larger probability of winning against C_1 than C_1 has against C_2. It has not previously been known how large or small the difference between those probabilities can be when, say, C_1 is a Huffman or Shannon-Fano code. We

define terminology for that probability difference and then proceed to derive an upper bound for it, and describe the tightness of the bound.

For a given source, the *competitive advantage* of code C_1 over code C_2 is the quantity

$$\Delta = P(W) - P(L).$$

Thus, $|\Delta| \leq 1$, and C_1 competitively dominates C_2 if and only if $\Delta \geq 0$. Whereas competitive optimality indicates whether one code always dominates other codes, competitive advantage quantifies by how much one code dominates over another code. If the codes C_1 and C_2 are complete, then their Kraft sums each equal 1 by Lemma 6.1.2; if additionally C_1 and C_2 are monotone and differ in codeword lengths for at least one source symbol, then it is impossible for C_1 to have codewords that are shorter than or equal in length to those of C_2 for every source symbol, so $\Delta \neq 1$.

The example in [16] gives a non-competitively-optimal Huffman code with a competitive advantage of $0.5 - 0.4 = 0.1$ over a Huffman code. Under the game-playing description of source coding in [7] quoted above, the competitive advantage of Player A's code over Player B's code is equal to the total expected earnings of Player A.

Example 6.1.4 (Golf analogy)**.** One round of professional golf typically consists of a collection of players competing for 18 holes. Each player's score for a hole is the number of strokes it takes that player to get the ball in the hole. A player's total score for the round is the sum of the player's scores for all 18 holes. The player with the lowest total score for the 18 holes is the winner of that round. This scoring method is sometimes called "stroke play" or "medal play". Another form of golf scoring is known as "match play", where (perhaps two) players compete against each other, and each hole results in either a "win", a "loss", or a "tie" depending on which player had the lowest number of strokes for that hole. A player's total score for the round is then the total number of wins that player achieved. A player wins the round if the player has more wins than the other player. Stroke play and match play golf scoring are analogous to average codeword length and competitive advantage, respectively, in lossless coding, if one associates the golf players with prefix codes, golf holes with uniformly drawn source symbols, and player scores for a hole with codeword lengths produced by a prefix code.

The following questions have not previously been answered in the literature: (i) How likely or unlikely is it for a Huffman code to be competitively optimal, allowing for non-dyadic sources?; (ii) If a Huffman code is not competitively optimal for a particular source, how large can the competitive advantage Δ of another code be over the Huffman code?; (iii) Do Huffman codes always (perhaps, strictly) competitively dominate Shannon-Fano codes for non-dyadic sources, and how large can the competitive advantage Δ of another code be over the Shannon-Fano

code?

One approach we exploit to answer the first question is to choose a source "at random" and seek the probability that a resulting Huffman code is competitively optimal. For the second and third questions, one can seek the best possible upper bound on the competitive advantage over a Huffman code or over a Shannon-Fano code.

In this paper we address these questions by proving the following main results: (1) Competitively optimal codes can exist for a given source only if some Huffman code is competitively optimal for that source (Theorem 6.2.7); (2) The probability that a Huffman code for a rather generally chosen random source is competitively optimal converges to zero as the source size grows (Theorem 6.3.5), and therefore the probability that competitively optimal codes exist for such sources also converges to zero (Corollary 6.3.7); (3) For all non-dyadic sources, Huffman codes strictly competitively dominate Shannon-Fano codes (Theorem 6.5.1); (4) For all non-dyadic sources, the competitive advantage Δ of any code over a Huffman code is strictly less than $\frac{1}{3}$ (Theorem 6.6.6); (5) For each integer $n > 3$, there exists a non-dyadic source of size n and some prefix code whose competitive advantage Δ over a Huffman code is arbitrarily close to $\frac{1}{3}$ (Theorem 6.6.7); (6) For each positive integer n, there exists a non-dyadic source of size n and a prefix code for the source such that the competitive advantage Δ of the code over a Shannon-Fano code for the source becomes arbitrarily close to 1 as $n \longrightarrow \infty$, and the average length of the code becomes arbitrarily close to one bit less than the average length of the Shannon-Fano code as $n \longrightarrow \infty$ (Theorem 6.7.1).

We also analyze "small" sources and show that for all sources of size at most 3, Huffman codes are competitively optimal (Theorem 6.8.1), and sources of size 4 for which Huffman codes are competitively optimal can be characterized in terms of a certain convex polyhedral condition (Theorem 6.8.2).

Finally we conducted computer simulations that drew a million random sources from a flat Dirichlet distribution for each source size up to 34 source symbols and determined whether the resulting Huffman code satisfies a sufficient condition for competitively optimality. Our numerical observations are given in Section 6.9 and indicate that the convergence proven in Theorem 6.3.5 is very rapid for relatively small source sizes (Figure 6.2).

6.2 Existence of competitively optimal codes

In this section, we use a result of Yamamoto and Itoh to show that competitively optimal codes can exist for a given source only if some Huffman code is competitively optimal for that source (Theorem 6.2.7). This result is later used in Section 6.3 to show that competitively optimal codes usually do not exist for large randomly chosen sources (Corollary 6.3.7).

If a source is dyadic, then there can be only one Huffman code (up to length equivalence) and this code

is length equivalent to a Shannon-Fano code. However, for non-dyadic sources, somewhat unusual circumstances can arise. In what follows, Example 6.2.1 illustrates a source where one Huffman code is competitively optimal but another Huffman code for the same source is not, and Example 6.2.2 illustrates a source with two Huffman codes and a non-Huffman code that form a cycle of strict competitive domination, as well as another non-Huffman code that is expected length optimal.

Example 6.2.1 (Two Huffman codes)**.**
A source with symbols a, b, c, d and corresponding probabilities $\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6}$ has a Huffman code H_1 with codeword lengths $2, 2, 2, 2$, and another Huffman code H_2 with lengths $1, 2, 3, 3$. Each Huffman code has competitive advantage zero over the other. However, one can verify that H_1 is competitively optimal, whereas H_2 is not.

Example 6.2.2 (Two Huffman codes and two other codes)**.**
A source with symbols a, b, c, d, e, f and corresponding probabilities $\frac{1}{3}, \frac{1}{3}, \frac{1}{9}, \frac{1}{9}, \frac{1}{18}, \frac{1}{18}$ has a Huffman code H_1 with codeword lengths $1, 2, 3, 4, 5, 5$ and another Huffman code H_2 with lengths $2, 2, 3, 3, 3, 3$ (see Figure 6.1), each with average codeword length equal to $\frac{7}{3}$. Two other trees for codes C_1 and C_2 are shown, which are relabeled versions of the trees for H_2 and H_1, respectively. H_1 strictly competitively dominates H_2 since its competitive advantage is $P(a)-P(d, e, f) = \frac{1}{9}$, while C_1 strictly competitively dominates H_1 since its competitive advantage is $P(b, c) - P(a) = \frac{1}{9}$, and H_2 strictly competitively dominates C_1 since its competitive advantage is $P(a, d, e, f) - P(b, c) = \frac{1}{9}$. That is, H_1, H_2, and C_1 form a cycle, which illustrates the non-transitivity of strict competitive dominance. Also, C_2 is an expected length optimal code, but it is non-Huffman since nodes e and f were not merged.

As observed in the example, although Huffman codes are always expected length optimal for any given source, expected length optimal prefix codes need not be Huffman codes. But it turns out that any expected length optimal code is length equivalent to some Huffman code for the source.

Lemma 6.2.3 (e.g., [10, p. 670])**.** *For any source, if a prefix code is expected length optimal, then it is monotone.*

The following lemma may be hinted at in the proof of [7, Lemma 5.8.1, p. 123], and is also a special case of Lemma 6.4.3 in Section 6.4.

Lemma 6.2.4 ([5])**.** *For any source, every expected length optimal prefix code is length equivalent to some Huffman code.*

Lemma 6.2.5 (Yamamoto and Itoh [16, Theorem 3])**.** *For any source, every competitively optimal code is expected length optimal.*

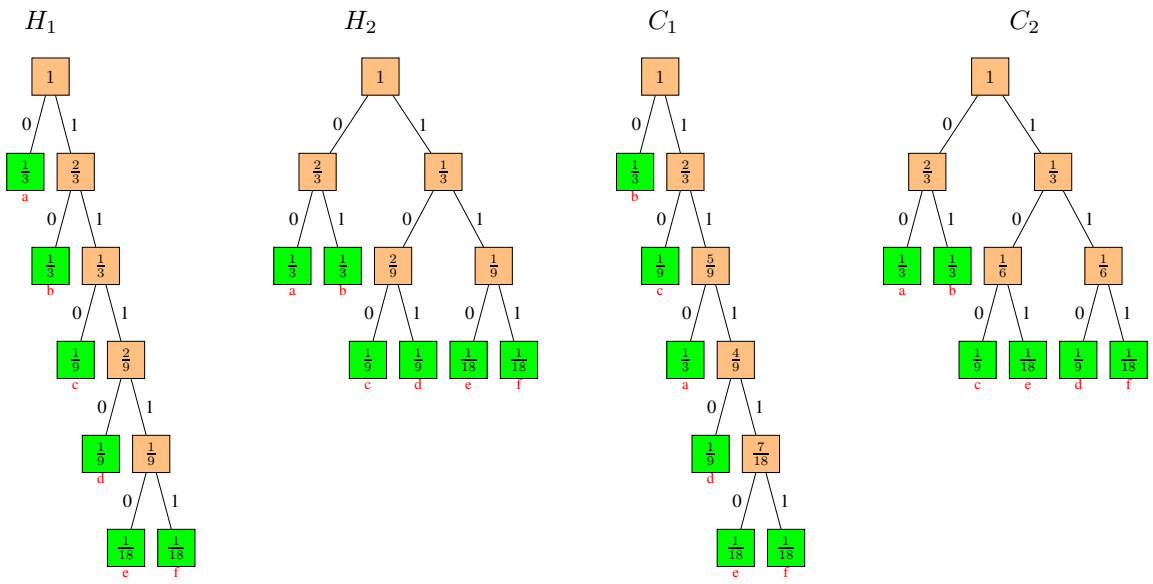


Figure 6.1. Code trees of four prefix codes for a source of size 6.

The following corollary follows immediately from Lemma 6.2.5 and Lemma 6.4.3 (in Section 6.4).

Corollary 6.2.6. *For any source, every competitively optimal code is length equivalent to some Huffman code.*

Note that the premises in Lemma 6.2.5 and Corollary 6.2.6 might be vacuous if no competitively optimal code exists for a particular source, in which case no conclusion can be drawn.

The following theorem shows that competitively optimal codes for a given source can exist only if some Huffman code is competitively optimal for the source.

Theorem 6.2.7. *For any source, if no Huffman code is competitively optimal, then no prefix code is competitively optimal.*

Proof. To prove the contrapositive, suppose some prefix code C is competitively optimal for a given source. Corollary 6.2.6 implies that some Huffman code maps source symbols to codewords of the same length as the codewords assigned by C. Thus, for any prefix code C', the competitive advantage of C over C' is nonnegative and is equal to the competitive advantage of the Huffman code over C', so the Huffman code also dominates C'. Thus, the Huffman code is competitively optimal. ■

6.3 Asymptotic converse to Cover's theorem on competitive optimality of Huffman codes

In this section, our goal is to analyze how likely it is that a source will have a competitively optimal Huffman code (and more generally any competitively optimal code). To address this question, we need to make

precise what "how likely" means in this context. There are many possible ways to choose a source "at random". By some means we wish to randomly obtain numbers $p_1, \ldots, p_n \in (0, 1)$ whose sum equals 1, interpret them as probabilities, and then determine whether a Huffman code constructed from these probabilities is competitively optimal.

One commonly used way to randomly obtain such probabilities is to sample from a "flat Dirichlet distribution". A *flat Dirichlet distribution* of size n has a uniform probability density on the $(n-1)$-dimensional simplex $\{(x_1, \ldots, x_n) \in [0, 1]^n : x_1 + \cdots + x_n = 1\}$ embedded in $\mathbb{R}^n$ (e.g., see [14]). For example, when $n = 3$, the point (p_1, p_2, p_3) is chosen uniformly from the (2-dimensional) triangle embedded in $\mathbb{R}^3$, whose vertices are $(0, 0, 1)$, $(0, 1, 0)$, and $(1, 0, 0)$. Choosing a random source from a flat Dirichlet distribution tends to be a natural approach since it treats all coordinates equally and uniformly. This method of random source selection was used, for example, to analyze average Huffman code rate redundancy in [11] and [15].

Another way to randomly create a source of size n is to choose n positive i.i.d. samples $X_1, \ldots, X_n$ according to some probability distribution on the positive reals, form their sum $S_n = X_1 + \cdots + X_n$, and then construct the normalized sequence $\frac{X_1}{S_n}, \ldots, \frac{X_n}{S_n}$. This technique specializes to a flat Dirichlet distribution when $X_1, \ldots, X_n$ are i.i.d. exponentials with mean one, as given in the following lemma.

Lemma 6.3.1 (e.g., [8, Chapter 11, Theorem 4.1])**.** *If $X_1, \ldots, X_n$ are i.i.d. exponential random variables with mean one and $S_n = X_1 + \cdots + X_n$, then the joint distribution of $\frac{X_1}{S_n}, \ldots, \frac{X_n}{S_n}$ is the same as that of a flat Dirichlet distribution of size n.*

Using this more general method for randomly generating a source of size n, we prove in this section that if the density of the i.i.d. sequence is positive on at least some interval $(0, \epsilon)$ with $\epsilon > 0$, then the probability a Huffman code is competitively optimal shrinks to 0 as n grows (Theorem 6.3.5). In other words, competitively optimal codes become rare for large sources. This result can be viewed as an asymptotic converse to Cover's theorem for dyadic sources (i.e., Theorem 6.1.3). It also indicates that Cover's result cannot be extended to many large sources beyond the dyadic ones.

We also examine an important special case of Theorem 6.3.5 when the random variables $X_1, \ldots, X_n$ are i.i.d. exponential.

Our Corollary 6.3.6 notes that the result of Theorem 6.3.5 is true when choosing a source at random from a flat Dirichlet distribution, which we use in Section 6.9 to gather experimental evidence of the convergence rate as $n \longrightarrow \infty$.

The next lemma shows that if one set of leaves of a Huffman code has a smaller Kraft sum than that of a second disjoint set of leaves, then one can find a prefix code that competitively wins against the Huffman code on

the set of smaller Kraft sum, loses on the set of larger Kraft sum, and ties on all other leaves.

Lemma 6.3.2. *For any source, if H is a Huffman code, and U and V are disjoint subsets of the source's alphabet S whose Huffman-Kraft sums satisfy $K(U) < K(V)$, then there exists a prefix code C such that*

$$U = \{x \in S : l_C(x) < l_H(x)\}$$
$$V = \{x \in S : l_C(x) > l_H(x)\}.$$

Proof. Let k be an integer such that

$$K(U) \leq (1 - 2^{-k})K(V) \tag{6.2}$$

and for each $x \in S$, define the following integer:

$$l(x) = \begin{cases} l_H(x) - 1 & \text{if } x \in U \\ l_H(x) + k & \text{if } x \in V \\ l_H(x) & \text{if } x \notin U \cup V. \end{cases}$$

Then

$$\begin{aligned} \sum_{x \in S} 2^{-l(x)} &= \sum_{x \in U} 2 \cdot 2^{-l_H(x)} + \sum_{x \in V} 2^{-k} \cdot 2^{-l_H(x)} + \sum_{x \notin U \cup V} 2^{-l_H(x)} \\ &= 2K(U) + 2^{-k}K(V) + (1 - K(U) - K(V)) \qquad (6.3) \\ &= 1 + K(U) - (1 - 2^{-k})K(V) \\ &\leq 1, \qquad (6.4) \end{aligned}$$

where (6.3) follows since the Kraft sum of all the Huffman codewords is 1, by Lemma 6.1.2; and (6.4) follows from (6.2). Therefore, the Kraft inequality (Lemma 6.1.1) implies that there exists a prefix code C whose codeword lengths are the values of $l(x)$ for all $x \in S$. That is, $l_C(x) = l_H(x) - 1$ for all $x \in U$; $l_C(x) = l_H(x) + k$ for all

$x \in V$; and $l_C(x) = l_H(x)$ for all $x \notin U \cup V$, and therefore

$$U = \{x \in S : l_C(x) < l_H(x)\}$$
$$V = \{x \in S : l_C(x) > l_H(x)\}.$$

■

Lemma 6.3.3. *For any source and any Huffman code for the source, if U and V are disjoint subsets of the source alphabet whose Huffman-Kraft sums satisfy $K(U) < K(V)$ and whose probabilities satisfy $P(U) > P(V)$, then the Huffman code is not competitively optimal for the source.*

Proof. Let S denote the source alphabet and let H be a Huffman code. By Lemma 6.3.2, since $K(U) < K(V)$, there exists a prefix code C such that

$$U = \{x \in S : l_C(x) < l_H(x)\}$$
$$V = \{x \in S : l_C(x) > l_H(x)\}.$$

The competitive advantage of C over the Huffman code H is thus $P(U) - P(V) > 0$, so the Huffman code is not competitively optimal. ■

The following lemma is essentially given in [16, equation (4)], but we give an alternate short proof here for completeness.

Lemma 6.3.4. *For any source, if y and y' are sibling nodes in a Huffman tree, z is a leaf node descendant of y, and $P(z) < P(y) - P(y')$, then the Huffman code is not competitively optimal for the source.*

Proof. We may assume $P(y) > P(y')$ since $P(z) > 0$. Let U and V be the sets of leaf node descendants of y and y', respectively. Let $U' = U - \{z\}$. Then, $K(U') < K(U) = K(V)$ (since siblings have the same Kraft sum) and $P(U') = P(U) - P(z) > P(U) - (P(U) - P(V)) = P(V)$. Then by Lemma 6.3.3, the Huffman code is not competitively optimal. ■

A hypothetical converse to Theorem 6.1.3 would say that Huffman codes for non-dyadic sources are never competitively optimal. This is not true (e.g., Theorem 6.8.1), but we are able to demonstrate that such a converse becomes probabilistically true as the source size grows.

Our next theorem, one of our main results, demonstrates an asymptotic converse to Theorem 6.1.3, by showing that competitively optimal Huffman codes become rare for randomly chosen sources as their size grows.

Theorem 6.3.5. *Let $\epsilon > 0$ and let $X_1, X_2, \ldots$ be an i.i.d. sequence of positive random variables with a density which is positive on at least $(0, \epsilon)$, and let $S_n = X_1 + \cdots + X_n$. Then the probability that a Huffman code for $\frac{X_1}{S_n}, \ldots, \frac{X_n}{S_n}$ is competitively optimal converges to zero as $n \longrightarrow \infty$.*

Proof. Let F denote the distribution function for each X_i. Let $\delta = \epsilon/24$. For each $k \in \{1, \ldots, 24\}$, define the interval

$$I_k = (\epsilon - k\delta, \epsilon - (k-1)\delta).$$

The intervals $I_1, \ldots, I_{24}$ are disjoint and their union lies in $(0, \epsilon)$.

Denote the indicator function for any $E \subseteq \mathbb{R}$ by $\mathbb{1}_E(x) = 1$ if $x \in E$ and $\mathbb{1}_E(x) = 0$ otherwise. For each $k \in \{1, \ldots, 24\}$, since the binary random variables $\mathbb{1}_{I_k}(X_i)$ are i.i.d. for all i, we have

$$\begin{aligned}
\frac{|\{i \in \{1, \ldots, n\} : X_i \in I_k\}|}{n} &= \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{I_k}(X_i) \\
&\xrightarrow{a.s.} E[\mathbb{1}_{I_k}(X_1)] \qquad (6.5) \\
&= P(X_1 \in I_k) \\
&= F(\epsilon - (k-1)\delta) - F(\epsilon - k\delta) \\
&> 0, \qquad (6.6)
\end{aligned}$$

where (6.5) follows from the strong law of large numbers (e.g., [4, Theorem 6.1]); and (6.6) follows since F is increasing on $(0, \epsilon)$. Let A be the event that all 24 of the convergences in (6.5) occur. The intersection of finitely many events with probability 1 has probability 1, so $P(A) = 1$.

We will show that for any outcome $\omega \in A$, there exists $N \geq 1$ such that for all $n \geq N$, any Huffman code for $\frac{X_1(\omega)}{S_n(\omega)}, \ldots, \frac{X_n(\omega)}{S_n(\omega)}$ is not competitively optimal. Suppose this has been done, and for each $n \geq 1$ let B_n be the event containing the outcomes ω such that a Huffman code for $\frac{X_1(\omega)}{S_n(\omega)}, \ldots, \frac{X_n(\omega)}{S_n(\omega)}$ is competitively optimal. If $\omega \in A$, then ω appears in only finitely many B_n, so $\omega \notin \limsup_{n\to\infty} B_n = \bigcap_{n\geq 1} \bigcup_{j \geq n} B_j$. Therefore $\limsup_{n\to\infty} B_n \subseteq A^c$, and so the theorem is proved using

$$\limsup_{n\to\infty} P(B_n) \leq P(\limsup_{n\to\infty} B_n) \leq P(A^c) = 0$$

where the first inequality follows from Fatou's Lemma (e.g., [4, Theorem 4.1]).

We will now prove what we promised. Let $\omega \in A$, and consider the sequence $X_1(\omega), X_2(\omega), \ldots$. Let

H be a Huffman tree constructed from the n probabilities $\frac{X_1(\omega)}{S_n(\omega)}, \ldots, \frac{X_n(\omega)}{S_n(\omega)}$. Since $\omega \in A$, the convergence in (6.5) holds for all $k \in \{1, \ldots, 24\}$, so for each such k there exists $N_k \geq 1$ such that for all $n \geq N_k$ we have $|\{i \in \{1, \ldots, n\} : X_i(\omega) \in I_k\}| \geq 1$.

Let $n \geq \max(N_1, \ldots, N_{24})$. Then for each $k \in \{1, \ldots, 24\}$, let Y_k equal $X_i(\omega)/S_n(\omega)$ for some $X_i(\omega) \in I_k$. That is, $Y_1 > Y_2 > \cdots > Y_{24}$ are probabilities corresponding to 24 of the n leaves in the Huffman tree H. Note that for all k, we have $Y_k S_n(\omega) \in I_k$, so

$$\frac{\epsilon - k\delta}{S_n(\omega)} < Y_k < \frac{\epsilon - (k-1)\delta}{S_n(\omega)}. \tag{6.7}$$

The sibling in H of the leaf for Y_{14} has probability at most Y_{13}, for otherwise the leaf for Y_{14} and its sibling would not have been nodes with two of the smallest available probabilities for merging as required by the Huffman construction. Then the probability $\widehat{Y}_{14}$ of the parent of the leaf for Y_{14} satisfies (using (6.7))

$$\widehat{Y}_{14} \leq Y_{14} + Y_{13} < \frac{\epsilon - 13\delta}{S_n(\omega)} + \frac{\epsilon - 12\delta}{S_n(\omega)} = \frac{\epsilon - \delta}{S_n(\omega)} < Y_1.$$

Then since the Huffman code H is monotone by Lemma 6.2.3, the leaf for Y_1 appears on a row in H that is at least as high as the row on which the parent of Y_{14} appears, so the leaf for Y_1 appears on a row strictly higher than the row on which the leaf for Y_{14} appears. Since $Y_1 > Y_2 > \cdots > Y_{14}$, monotonicity of Huffman codes shows the row numbers on which the leaves for these 14 probabilities appear are non-decreasing, so the conclusion from the previous sentence implies the leaf for some probability in the sequence $Y_1, Y_2, \ldots, Y_{14}$ appears on a row strictly higher in H than the leaf for the next probability in the sequence. Specifically, there exists $m \in \{1, \ldots, 13\}$ such that the leaf for Y_m appears on a row r strictly higher than the row r' on which the leaf for Y_{m+1} appears, i.e., $r < r'$.

Similarly, the sibling for the leaf for Y_{21} has probability at most Y_{20}, so the probability $\widehat{Y}_{21}$ of the parent of the leaf for Y_{21} satisfies (using (6.7))

$$\widehat{Y}_{21} \leq Y_{21} + Y_{20} < \frac{\epsilon - 20\delta}{S_n(\omega)} + \frac{\epsilon - 19\delta}{S_n(\omega)} = \frac{\epsilon - 15\delta}{S_n(\omega)} < Y_{14} \leq Y_{m+1}. \tag{6.8}$$

Then since Huffman codes are monotone, the leaf for Y_{m+1} appears on a row that is at least as high as the row on which the parent of Y_{21} appears, so the row r' on which the leaf for Y_{m+1} appears is strictly higher than the row r'' on which the leaf for Y_{21} appears, i.e., $r' < r''$.

Let U be the set consisting of the leaves for Y_{m+1} and Y_{21}, and let V be the set consisting of the leaf for

Y_m. Then

$$K(U) = 2^{-r'} + 2^{-r''} < 2^{-(r'-1)} \leq 2^{-r} = K(V),$$

and again using (6.7),

$$P(V) - P(U) = Y_m - (Y_{m+1} + Y_{21}) < \frac{\epsilon - (m-1)\delta}{S_n(\omega)} - \left(\frac{\epsilon - (m+1)\delta}{S_n(\omega)} + \frac{\epsilon - 21\delta}{S_n(\omega)} \right) = \frac{-\delta}{S_n(\omega)} < 0.$$

Then Lemma 6.3.3 implies the Huffman code for $\frac{X_1(\omega)}{S_n(\omega)}, \ldots, \frac{X_n(\omega)}{S_n(\omega)}$ is not competitively optimal, which is what we wanted to show. ■

The requirements for the density of the random variables chosen in the statement of Theorem 6.3.5 are not very restrictive and are satisfied by a wide range of random variables. Such densities include various versions of at least the following: beta, chi-square, Erlang, exponential, gamma, Gumbel, log-normal, logistic, Maxwell, Pareto, Rayleigh, uniform, and Weibull.

The following corollary follows from Lemma 6.3.1 and Theorem 6.3.5. It shows that competitively optimal Huffman codes become rare for large sources selected uniformly at random from a simplex.

Corollary 6.3.6. *Let $X_1, \ldots, X_n$ be the probabilities of a source chosen randomly from a flat Dirichlet distribution in $\mathbb{R}^n$. Then the probability that a Huffman code for this source is competitively optimal converges to zero as $n \longrightarrow \infty$.*

The next corollary follows immediately from Theorem 6.2.7 and Theorem 6.3.5. It shows that as the source size grows the existence of any competitively optimal codes becomes unlikely.

Corollary 6.3.7. *Let $\epsilon > 0$ and let $X_1, X_2, \ldots$ be an i.i.d. sequence of nonnegative random variables with a density which is positive on at least $(0, \epsilon)$, and let $S_n = X_1 + \cdots + X_n$. Then the probability that a competitively optimal code exists for the source $\frac{X_1}{S_n}, \ldots, \frac{X_n}{S_n}$ converges to zero as $n \longrightarrow \infty$.*

6.4 Lemmas for future sections

In this section, we give a number of lemmas that are used in the remainder of this paper.

Huffman codes are expected length optimal and Gallager [10, Theorem 1] characterized such codes in terms of a "sibling property", which says the code is complete and if the nodes in the code tree can be listed in order of non-increasing probability with each node being adjacent in the list to its sibling.

However, Huffman codes are not the only expected length optimal prefix codes. Expected length optimal prefix codes are characterized as those which are length equivalent to some Huffman code for the source.

Below we prove a second characterization of expected length optimal prefix codes, or equivalently prefix codes which are length equivalent to a Huffman code. Namely we show that such codes are strongly monotone. Strong monotonicity is a weaker condition than the sibling property, and thus a broader class of prefix codes are strongly monotone.

Let C be a prefix code a source has alphabet S. If $U \subset S$, then denote the Kraft sum of C's codeword lengths corresponding to the source symbols in U by

$$K_C(U) = \sum_{x \in U} 2^{-l_C(x)}.$$

Definition 6.4.1. Given a source with alphabet S, a prefix code C for the source is *strongly monotone* if for any subsets $A, B \subseteq S$, if there exist integers $i, j \geq 0$ such that $K_C(A) = 2^{-i} > 2^{-j} = K_C(B)$, then $P(A) \geq P(B)$.

The following lemma notes several properties that are preserved under length equivalence between prefix codes.

Lemma 6.4.2 ([5])**.** *If two prefix codes are length equivalent, then each of the following properties holds for one code if and only if it holds for the other code:*

- *completeness*
- *strong monotonicity*
- *expected length optimality.*

Lemma 6.4.3 ([5, Theorem 2.4])**.** *For any source and any prefix code C for the source, the following are equivalent:*

(1) C is complete and strongly monotone

(2) C is length equivalent to a Huffman code

(3) C is expected length optimal.

For a given source and a Huffman code, if A is a nonempty proper subset of the source alphabet whose Huffman-Kraft sum has binary expansion $K(A) = 0.b_1b_2\ldots$, then we define a *Huffman-Kraft partition* of A to be any sequence $A_1, A_2, \ldots$ of disjoint (possible empty) subsets of A whose union is A and such that $K(A_i) = b_i 2^{-i}$ for each i. If $b_k = 1$ and $b_i = 0$ for all $i \geq k+1$, then it suffices to specify the first k sets $A_1, \ldots, A_k$ in a Huffman-Kraft partition.

Lemma 6.4.4. *Every nonempty proper subset of a source's alphabet has a Huffman-Kraft partition.*

Proof. Let A be a nonempty proper subset of a source's alphabet and let $0.a_1a_2\ldots a_k$ be the binary expansion of the Huffman-Kraft sum $K(A)$, where $a_k = 1$.

We use induction on $N = a_1+\cdots+a_k$. Since A is nonempty, $K(A) > 0$, so $N \geq 1$. First, suppose $N = 1$. Then $a_i = 0$ for all $i \in \{1,\ldots,k-1\}$, so $K(A) = 2^{-k}$. Setting $A_k = A$ and $A_i = \varnothing$ for $i \in \{1,\ldots,k-1\}$ gives a Huffman-Kraft partition of A.

Now let $m \geq 2$ and suppose the lemma holds for $N = m-1$. We will next show the lemma must hold for $N = m$. Let $X = \{u \in A : K(u) \leq 2^{-k}\}$. Then $K(A-X)$ is an integer multiple of 2^{-k+1}, since for all $v \in A - X$ we have $K(v) = 2^{-k+i}$ for some $i \geq 1$. Thus, the binary expansion of $K(A-X)$ has 0s in all positions $i \geq k$. Therefore, since $a_k = 1$ and $K(A) = K(X) + K(A-X)$, the binary expansion of $K(X)$ has a 1 in position k, so $K(X) \geq 2^{-k}$.

Let Y be a subset of X whose Huffman-Kraft sum is minimum among all subsets of X with Huffman-Kraft sum at least 2^{-k}. We will show that $K(Y) = 2^{-k}$. Suppose to the contrary that $K(Y) > 2^{-k}$. Let $y \in Y$ be an element with minimum probability among the elements of Y. Since $y \in X$, we know $K(y) \leq 2^{-k}$, and also $K(y) = 2^{-i}$ for some integer i. Therefore, 2^{-k} is an integer multiple of $K(y)$. Also, $K(Y)$ is an integer multiple of $K(y)$, since for all $u \in Y$ there exists an $i \geq 0$ such that $K(u) = 2^iK(y)$. Therefore, $K(Y) \geq 2^{-k} + K(y)$ since $K(Y) > 2^{-k}$. But then $K(Y-\{y\}) = K(Y) - K(y) \geq 2^{-k}$ and $K(Y-\{y\}) = K(Y) - K(y) < K(Y)$, so the Huffman-Kraft sum of $Y - \{y\}$ is at least 2^{-k} but is smaller than that of Y, contradicting the minimality assumption on Y. Therefore, $K(Y) = 2^{-k}$.

Set $A_k = Y$. Since $m \geq 2$, the set $A - A_k$ is nonempty, and since A is a proper subset of the source's alphabet, so is $A - A_k$. Also, the Huffman-Kraft sum $K(A-A_k) = K(A) - K(A_k) = K(A) - 2^{-k}$ has exactly $m-1$ ones in its binary expansion. Thus, the induction hypothesis implies that $A - A_k$ has a Huffman-Kraft partition $A_1,\ldots,A_{k-1}$. Then $A_1,\ldots,A_k$ is a Huffman-Kraft partition of A. ■

Lemma 6.4.5. *Let A and B be subsets of a source alphabet whose Huffman-Kraft sums satisfy $K(A) = K(B) = 2^{-i}$ for some integer $i \geq 0$, and such that $|A| \geq 2$. Then $P(A) \leq 2P(B)$.*

Proof. By Lemma 6.4.3, Huffman codes are strongly monotone, so Huffman-Kraft sums obey the strong monotonicity condition.

Let $a \in A$ be an element of minimum Huffman-Kraft sum among the elements of A. Since $|A| \geq 2$, there exists $m \geq i+1$ such that $K(a) = 2^{-m}$, for otherwise $K(A) \geq 2 \cdot 2^{-i} > 2^{-i} = K(A)$, a contradiction. Therefore, the binary expansion of $K(A-\{a\}) = K(A) - K(a) = 2^{-i} - 2^{-m}$ has 1s in positions $i+1,\ldots,m$ and 0s in all other positions. By Lemma 6.4.4, there exists a Huffman-Kraft partition $A_1,\ldots,A_m$ of $A-\{a\}$. In particular, $K(A_{i+1}) = 2^{-(i+1)}$, and so $K(A-A_{i+1}) = K(A) - K(A_{i+1}) = 2^{-i} - 2^{-(i+1)} = 2^{-(i+1)}$. Then by

strong monotonicity $P(A_{i+1}), P(A - A_{i+1}) \leq P(B)$, and so $P(A) = P(A_{i+1}) + P(A - A_{i+1}) \leq 2P(B)$. ■

Lemma 6.4.6. *Given a source with alphabet S, suppose there is a subset $U \subseteq S$ whose Huffman-Kraft sum $K(U)$ is an integer multiple of 2^{-i} for some integer $i \geq 0$. If $A \subseteq U$ with $0 < K(A) < 2^{-j}$ for some integer $j \geq i$, then there exists a subset $B \subseteq U - A$ with $K(A \cup B) = 2^{-j}$.*

Proof. Since $K(A) > 0$ and $K(U - A) = K(U) - K(A) > 2^{-i} - 2^{-j} \geq 0$, Lemma 6.4.4 shows there exist Huffman-Kraft partitions $A_1, A_2, \ldots$ of A, and $B_1, B_2, \ldots$ of $U - A$. Let

$$B = \bigcup_{k>j} B_k.$$

Then $K(B) = K(B_{j+1}) + K(B_{j+2}) + \cdots \leq 2^{-(j+1)} + 2^{-(j+2)} + \cdots = 2^{-j}$, so $K(A \cup B) = K(A) + K(B) < 2^{-(j-1)}$. Since $K(U) = K(A \cup B) + K(B_1 \cup \cdots \cup B_j)$, and both $K(U)$ and $K(B_1 \cup \cdots \cup B_j)$ are integer multiples of 2^{-j}, it must be the case that $K(A \cup B)$ is an integer multiple of 2^{-j} as well. Then $0 < K(A \cup B) < 2^{-(j-1)}$ implies $K(A \cup B) = 2^{-j}$. ■

Lemma 6.4.7. *Given a source with alphabet S, suppose $A, B \subseteq S$ with Huffman-Kraft sums satisfying $2K(B) \leq 2^{-i} \leq K(A)$ for some integer i. Then $P(A) \geq P(B)$, with equality possible only if $K(A) = 2K(B)$.*

Proof. If B is empty, then $P(B) = 0$ and the result follows, so assume B is nonempty.

If $K(B) < 2^{-(i+1)}$, then by Lemma 6.4.6 (where A, B, U, i, j in Lemma 6.4.6 respectively correspond to B, B', S, 0, $i+1$ here) there exists a subset $B' \subseteq S - B$ with $K(B') = 2^{-(i+1)} - K(B)$. Alternatively, if $K(B) = 2^{-(i+1)}$ then we will let $B' = \varnothing$. In either case, $K(B \cup B') = 2^{-(i+1)}$.

By Lemma 6.4.4, there exists a Huffman-Kraft partition $A_1, A_2, \ldots$ of A. Let k be the smallest integer such that $A_k \neq \varnothing$. Since $K(A) \geq 2^{-i}$, we have $k \leq i$.

If $k < i$, then since $0 < K(B \cup B') = 2^{-(i+1)} < 2^{-(k+1)}$, Lemma 6.4.6 implies (where A, B, U, i, j in Lemma 6.4.6 respectively correspond to $B \cup B'$, E, S, 0, $i+1$ here) there exists a subset $E \subseteq S - (B \cup B')$ such that $K(E) = 2^{-(k+1)} - K(B \cup B')$. If $k = i$, then instead let $E = \varnothing$. In either case, $K(B \cup B' \cup E) = 2^{-(k+1)}$.

We have

$$\begin{aligned} P(A) &= P(A_k) + P(A - A_k) \\ &\geq P(A_k) && (6.9) \\ &\geq P(B \cup B' \cup E) && (6.10) \\ &= P(B) + P(B') + P(E) && (6.11) \\ &\geq P(B). && (6.12) \end{aligned}$$

where (6.10) follows the Huffman-Kraft sum equality $K(A_k) = 2^{-k} = 2K(B \cup B' \cup E)$ since Huffman codes are strongly monotone by Lemma 6.4.3; and (6.11) follows since B, B', and E are disjoint.

Now suppose $K(A) > 2K(B)$. Then either $K(B) < 2^{-(i+1)}$ or $K(A) > 2^{-i}$. In the first case, we have $P(B') > 0$ since $K(B \cup B') = 2^{-(i+1)}$, so the inequality in (6.12) becomes strict. Now consider two subcases of the second case. If $K(A) < 2^{-(i-1)}$, then $i - 1 < k \leq i$, so $k = i$ which implies $K(A - A_k) = K(A - A_i) = K(A) - K(A_i) = K(A) - 2^{-i} > 0$, and thus the inequality in (6.9) becomes strict. Otherwise, if $K(A) \geq 2^{-(i-1)}$, then $k < i$, and so $K(E) = 2^{-(k+1)} - K(B \cup B') = 2^{-(k+1)} - 2^{-(i+1)} > 0$. Thus $P(E) > 0$, and the inequality in (6.12) becomes strict. ■

6.5 Huffman codes competitively dominate Shannon-Fano codes

For dyadic sources, there can be only one Huffman code (up to length equivalence) and such a code is length equivalent to a Shannon-Fano code [7, Theorem 5.3.1], and thus the competitive advantage of either code over the other code is $\Delta = 0$. In this section, Theorem 6.5.1 shows that for all non-dyadic sources, every Huffman code always strictly competitively dominates every Shannon-Fano code.

This result shows that Shannon-Fano codes are never competitively optimal for non-dyadic sources. Actually, as the source size grows, Huffman codes themselves are usually not competitively optimal either, as shown in Section 6.3.

The following theorem shows that when a source is not dyadic, every Huffman code always has a positive competitive advantage over the Shannon-Fano code.

Theorem 6.5.1. *Huffman codes strictly competitively dominate Shannon-Fano codes if and only if the source is not dyadic.*

Proof. Suppose a non-dyadic source has alphabet S and a Huffman code H. By Lemma 6.4.3, Huffman codes are strongly monotone, so Huffman-Kraft sums obey the strong monotonicity condition.

Denote the Huffman and Shannon-Fano codeword lengths for each $y \in S$ by $l_{\text{H}}(y)$ and $l_{\text{SF}}(y)$, respectively. Let $W = \{i \in S : l_{\text{SF}}(y) < l_{\text{H}}(y)\}$ and $L = \{i \in S : l_{\text{SF}}(y) > l_{\text{H}}(y)\}$ and $T = S - (W \cup L)$, as in (6.1). It suffices to show $P(W) < P(L)$.

If $y \in W$ then

$$\log_2 \frac{1}{P(y)} \leq \left\lceil \log_2 \frac{1}{P(y)} \right\rceil = l_{\text{SF}}(y) \leq l_{\text{H}}(y) - 1,$$

and if $y \in L$ then

$$\log_2 \frac{1}{P(y)} > \left\lceil \log_2 \frac{1}{P(y)} \right\rceil - 1 = l_{\text{SF}}(y) - 1 \geq l_{\text{H}}(y).$$

Therefore, probabilities and Huffman-Kraft sums of wins, losses, and ties are bounded as

$$\begin{aligned}
P(y) &\geq 2 \cdot 2^{-l_{\text{H}}(y)} = 2K(y) && \text{if } y \in W \\
P(y) &< 2^{-l_{\text{H}}(y)} = K(y) && \text{if } y \in L \\
K(y) = 2^{-l_{\text{H}}(y)} \leq P(y) &< 2 \cdot 2^{-l_{\text{H}}(y)} = 2K(y) && \text{if } y \in T.
\end{aligned}$$

Thus, for any nonempty subset $A \subseteq S$,

$$P(A) \geq 2K(A) \qquad \text{if } A \subseteq W \tag{6.13}$$

$$P(A) < K(A) \qquad \text{if } A \subseteq L \tag{6.14}$$

$$K(A) \leq P(A) < 2K(A) \quad \text{if } A \subseteq T, \tag{6.15}$$

since $P(A) = \sum_{y \in A} P(y)$ and $K(A) = \sum_{y \in A} K(y)$ for any subset $A \subseteq S$.

Suppose that $L = \varnothing$. Then from (6.13) and (6.15) we have $P(y) \geq K(y)$ for all $y \in S$. Since $K(y)$ is an integer power of $1/2$ for all $y \in S$, there exists at least one element $y \in S$ with $P(y) > K(y)$, or else the source would be dyadic. But then

$$1 = \sum_{y \in S} P(y) > \sum_{y \in S} K(y) = 1,$$

which is a contradiction. Thus, in fact $L \neq \varnothing$, and therefore $P(L) > 0$. This implies $P(W) < 1$.

If $W = \varnothing$ then $P(W) - P(L) < 0$, and we are done. Suppose $W \neq \varnothing$. By Lemma 6.4.4, there exist Huffman-Kraft partitions $W_1, W_2, \ldots$ and $L_1, L_2, \ldots$ of W and L, respectively. Let k be the smallest integer such

that $W_k \neq \varnothing$.

If $k = 1$, then $W_1 \neq \varnothing$, so $K(W_1) = \frac{1}{2}$ and therefore we get the contradiction that $1 > P(W_1) \geq 2K(W_1) = 1$ by (6.13). Thus $k \geq 2$.

We will use the following fact in the remainder of the proof. If $A \subseteq S$ and $K(A) = 2^{-(k-1)}$, then

$$P(A) \geq P(W_k) \tag{6.16}$$

$$\geq 2K(W_k) \tag{6.17}$$

$$= 2^{-(k-1)} \tag{6.18}$$

$$= K(A) \tag{6.19}$$

where (6.16) follows from strong monotonicity; (6.17) follows from (6.13); and (6.18) follows from $P(W_k) = 2^{-k}$ since $W_k \neq \varnothing$. Also, $L_{k-1} = \varnothing$, for otherwise $K(L_{k-1}) = 2^{-(k-1)}$, which by (6.19) would imply $P(L_{k-1}) \geq K(L_{k-1})$, contradicting $P(L_{k-1}) < K(L_{k-1})$ by (6.14), as $L_{k-1} \subseteq L$.

Suppose $K(L) < 2^{-(k-2)}$. Then since $L_{k-1} = \varnothing$, at least the first $k-1$ bits in the binary expansion of $K(L)$ are zero, so $K(L) < 2^{-(k-1)}$. By Lemma 6.4.6, (where i, j, U, A, B, in Lemma 6.4.6 respectively correspond to 0, $k-2$, S, $W_k \cup L$, A here), since $0 < K(W_k \cup L) < 2^{-k} + 2^{-(k-1)} < 2^{-(k-2)} \leq 1 = K(S)$, there exists a subset $A \subseteq S - (W_k \cup L)$ such that $K(A) = 2^{-(k-2)} - K(W_k \cup L)$. Then

$$\begin{aligned} K(L \cup A) &= K(W_k \cup L \cup A) - K(W_k) \\ &= K(W_k \cup L) + K(A) - K(W_k) \\ &= 2^{-(k-2)} - K(W_k) \\ &= 2^{-(k-2)} - 2^{-k} \\ &= 2^{-(k-1)} + 2^{-k} \end{aligned}$$

so the Huffman-Kraft partition of $L \cup A$ provided by Lemma 6.4.4 consists of 2 disjoint subsets, E and F, of $L \cup A$ such that $E \cup F = L \cup A$, along with $K(E) = 2^{-k}$ and $K(F) = 2^{-(k-1)}$. Then $P(E) > 0$, and $P(F) \geq K(F)$ by (6.19).

Since $L \subseteq L \cup A = E \cup F$, we have $S - (W_k \cup E \cup F) \subseteq S - L \subseteq W \cup T$. Therefore, $P(S - (W_k \cup$

$E \cup F)) \geq K(S - (W_k \cup E \cup F)) = 1 - 2^{-(k-2)}$ by (6.13) and (6.15). But we also have

$$\begin{aligned} P(S - (W_k \cup E \cup F)) &= 1 - P(W_k) - P(E) - P(F) \\ &< 1 - P(W_k) - P(F) \\ &\leq 1 - 2K(W_k) - K(F) \qquad (6.20) \\ &= 1 - 2^{-(k-1)} - 2^{-(k-1)} \\ &= 1 - 2^{-(k-2)} \\ &= K(S - (W_k \cup E \cup F)), \end{aligned}$$

which is a contradiction, where (6.20) follows from (6.13). Therefore, our assumption was false that $K(L) < 2^{-(k-2)}$, so in fact $K(L) \geq 2^{-(k-2)}$. Thus, $k \neq 2$, for otherwise $K(L) \geq 1$ which contradicts $W \neq \varnothing$. Therefore, $k \geq 3$. Since $W_1 = \cdots = W_{k-1} = \varnothing$, we have $2K(W) < 2^{-(k-2)} \leq K(L)$, and so Lemma 6.4.7 shows $P(W) < P(L)$. ■

Theorem 6.5.1 guarantees that for each n, and for all non-dyadic sources of size n, Huffman codes always strictly competitively dominate Shannon-Fano codes. On the other hand, we saw in Section 6.3 that as the source size grows, an increasingly large fraction of non-dyadic sources have prefix codes that strictly competitively dominate Huffman codes. Said more casually, Huffman codes usually are dominated by another code but always dominate Shannon-Fano codes.

6.6 Bound on competitive advantage over Huffman codes

In this section, we derive an upper bound on the competitive advantage of an arbitrary prefix code over a Huffman code for a given source, and show that for every source of size at least four the upper bound can be approached arbitrarily closely by some sources. We show that no prefix code can have a competitive advantage of $\frac{1}{3}$ or higher over any Huffman code (Theorem 6.6.6), and in fact this upper bound is tight in that it can be approached arbitrarily closely from below for all source sizes, by at least some sources (Theorem 6.6.7).

If A is a subset of a source's alphabet, then at least one of the following three Huffman-Kraft sum conditions is satisfied: (i) A is empty and $K(A) = 0$; (ii) A is the entire alphabet and $K(A) = 1$; or (iii) $K(A)$ is a finite sum of negative integer powers of 2, and has a binary expansion of the form $K(A) = 0.b_1 b_2 \ldots$, with $b_i \in \{0, 1\}$ for all i, and where the number of nonzero bits is at least one and is finite.

If A is a subset of a source alphabet, then the probability $P(A)$ and the Huffman-Kraft sum $K(A)$ are related. When A is empty, $P(A) = K(A) = 0$, and when A is the entire source alphabet, $P(A) = K(A) = 1$. If

the source is dyadic, then $P(A) = K(A)$ is always true. For non-dyadic sources, the relationship between $P(A)$ and $K(A)$ is more complicated. We next establish some lemmas that are used to prove Theorem 6.6.6. Some of these lemmas relate the probabilities and the Huffman-Kraft sums of source alphabet subsets.

In this section, for any given source, if C is a prefix code that competes against a Huffman code for the same source, then define the events W (i.e., C "wins"), L (i.e., C "loses"), and T (i.e., C "ties"), as in (6.1) (taking $C_1 = C$, and C_2 as the Huffman code).

In what follows, the definitions of W, L, and T from (6.1) (taking C_2 as the Huffman code) are used in Lemma 6.6.1, Lemma 6.6.2, and Theorem 6.6.6.

Lemma 6.6.1. *For any source, if a prefix code C has a positive competitive advantage over a Huffman code, then for at least one source symbol, C produces a longer codeword than the Huffman codeword.*

Proof. It suffices to show that L is nonempty. Since C has a positive competitive advantage over a Huffman code H, we have $P(W) > P(L)$. Then,

$$0 \leq E[l_C(X)] - E[l_H(X)] \tag{6.21}$$

$$= \sum_{y \in W} (l_C(y) - l_H(y))\, P(y) + \sum_{y \in L} (l_C(y) - l_H(y))\, P(y) \tag{6.22}$$

$$< \sum_{y \in L} (l_C(y) - l_H(y))\, P(y) \tag{6.23}$$

where (6.21) follows since a prefix code C cannot have a lower expected length than the Huffman code for a given source; (6.22) follows since $l_C(y) - l_H(y) = 0$ for all $y \in T$; and (6.23) follows since $l_C(y) - l_H(y) < 0$ for all $y \in W$ and $P(W) > P(L) \geq 0$, so $W \neq \varnothing$. If $L = \varnothing$, then (6.23) would yield a contradiction. ■

Lemma 6.6.2. *For any source, if a prefix code C has a positive competitive advantage over a Huffman code, then the Huffman-Kraft sums of the set W of wins and set L of losses of C satisfy $K(W) < K(L)$.*

Proof. Let H denote the Huffman code and suppose, to the contrary, that

$$K(W) \geq K(L). \tag{6.24}$$

Let S be the source's alphabet and let $T = S - (W \cup L)$ be the set of ties. Then

$$
\begin{aligned}
1 &\geq \sum_{x \in S} 2^{-l_C(x)} && (6.25)\\
&= \sum_{x \in W} 2^{-l_C(x)} + \sum_{x \in T} 2^{-l_C(x)} + \sum_{x \in L} 2^{-l_C(x)} \\
&> \sum_{x \in W} 2^{-l_C(x)} + \sum_{x \in T} 2^{-l_C(x)} && (6.26)\\
&= \sum_{x \in W} 2^{-l_C(x)} + \sum_{x \in T} 2^{-l_H(x)} && (6.27)\\
&\geq \sum_{x \in W} 2^{-l_H(x)+1} + \sum_{x \in T} 2^{-l_H(x)} && (6.28)\\
&= 2K(W) + K(T) \\
&\geq K(W) + K(T) + K(L) && (6.29)\\
&= 1, && (6.30)
\end{aligned}
$$

a contradiction, where (6.25) is the Kraft inequality (6.1.1) applied to the prefix code C; (6.26) follows from $L \neq \varnothing$ by Lemma 6.6.1; (6.27) follows from $l_C(x) = l_H(x)$ when $x \in T$; (6.28) follows from $l_C(x) \leq l_H(x) - 1$ when $x \in W$; (6.29) follows from (6.24); and (6.30) follows since the Huffman tree is complete. ■

Lemma 6.6.3. *If $b > a$, then $\frac{x+a}{x+b}$ is monotonically increasing in x for all $x \neq -b$.*

Proof. $\frac{d}{dx}\left(\frac{x+a}{x+b}\right) = \frac{b-a}{(x+b)^2} > 0.$ ■

Lemma 6.6.4. *Given a source with alphabet S, let U and V be disjoint subsets of S with Huffman-Kraft sums satisfying $K(U) < 2^{-i} \leq K(V)$ for some integer $i \geq 0$. Then $P(U) < 2P(V)$.*

Proof. Since $K(V) \geq 2^{-i}$, we have $P(V) > 0$. If $K(U) = 0$ then $P(U) = 0 < 2P(V)$, so we may assume $K(U) > 0$.

By Lemma 6.4.4, there exists a Huffman-Kraft partition $V_1, V_2, \ldots$ of V. Let $k \geq 0$ be the smallest integer such that $V_k \neq \varnothing$. Since $K(V) \geq 2^{-i}$, we have $k \leq i$. By Lemma 6.4.6 (taking $B = U'$, $U = S$, $A = U$, and $j = k$), there exists a subset $U' \subseteq S - U$ such that $K(U') = 2^{-k} - K(U) > 0$. Then $|U \cup U'| \geq 2$ since both U and U' are nonempty, and also $K(U \cup U') = 2^{-k} = K(V_k)$, so $P(U \cup U') \leq 2P(V_k)$ by Lemma 6.4.5 (taking $A = U \cup U'$, $B = V_k$, and $i = k$). Therefore

$$P(U) \leq 2P(V_k) - P(U') < 2P(V_k) \leq 2P(V).$$

■

For any Huffman-Kraft partition $A_1, A_2, \ldots$ of a subset A of a source alphabet and for any integer $k \geq 1$ we will define the notation $A_{<k} = A_1 \cup \cdots \cup A_{k-1}$ and $A_{\leq k} = A_1 \cup \cdots \cup A_k$, as well as $A_{>k} = A_{k+1} \cup A_{k+2} \ldots$ and $A_{\geq k} = A_k \cup A_{k+1} \ldots$.

Lemma 6.6.5. *Given a source with alphabet S, suppose $U, V \subseteq S$ are disjoint subsets with Huffman-Kraft sums satisfying $K(U) < K(V)$. Then $P(U) - P(V) < \frac{1}{3}$.*

Proof. If $K(U) = 0$ then $P(U) - P(V) \leq 0 < \frac{1}{3}$, so suppose $K(U) > 0$. By Lemma 6.4.4, there exist Huffman-Kraft partitions $U_1, U_2, \ldots$ of U and $V_1, V_2, \ldots$ of V. Let $m \geq 0$ be the smallest integer such that $V_m \neq \varnothing$ and $U_m = \varnothing$; such an integer exists because $K(U) < K(V)$. Since $P(U) - P(V) \leq P(U) - P(V_{\leq m})$, and U and $V_{\leq m}$ are disjoint, without loss of generality we will assume $V_i = \varnothing$ for all $i > m$.

We now assert that for certain nonnegative integers $k \leq m-1$, there exists $A \subseteq S$ satisfying the following three conditions:

$$K(A) = 2^{-k} \tag{6.31}$$

$$U_{\geq k+1} \cup V_{\geq k+1} \subseteq A \tag{6.32}$$

$$P(U_{\geq k+1}) - P(V_{\geq k+1}) < \frac{1}{3}P(A). \tag{6.33}$$

Specifically, we will first show that this assertion is true when $k = m - 1$. Then, we will show inductively that whenever the assertion is true for some positive k it must also be true for some smaller nonnegative k. We then will conclude that the assertion must be true for $k = 0$.

Once we have established the assertion is true for $k = 0$, we can further infer that

$$P(U) - P(V) = P(U_{\geq 1}) - P(V_{\geq 1}) < \frac{1}{3}P(A) = \frac{1}{3},$$

where $P(A) = 1$ since $K(A) = 1$, thus proving the lemma.

Base Step: $k = m - 1$. Since $K(S - (U_{<m} \cup V_{<m})) = 1 - K(U_{<m} \cup V_{<m})$ is an integer multiple of $2^{-(m-1)}$, and

$$0 < K(V_m) \leq K(U_{\geq m} \cup V_{\geq m}) = K(U_{>m}) + K(V_m) < 2^{-(m-1)},$$

Lemma 6.4.6 shows (taking $A \leftarrow U_{\geq m} \cup V_{\geq m}$, $B \leftarrow U'$, $U \leftarrow S - (U_{<m} \cup V_{<m})$, $i = j \leftarrow m - 1$) there exists

$U' \in= S - (U \cup V)$ such that $K(U' \cup U_{\geq m} \cup V_{\geq m}) = 2^{-(m-1)}$. Then setting

$$A = U' \cup U_{\geq m} \cup V_{\geq m} \tag{6.34}$$

gives

$$K(A) = 2^{-(m-1)}. \tag{6.35}$$

Also, since $K(U_{\geq m}) = K(U_{>m}) < 2^{-m} = K(V_m) = K(V_{\geq m})$, Lemma 6.6.4 shows $P(U_{\geq m}) < 2P(V_{\geq m})$. Therefore,

$$\frac{P(U_{\geq m}) - P(V_{\geq m})}{P(A)} < \frac{P(U_{\geq m}) - P(V_{\geq m})}{P(U_{\geq m}) + P(V_{\geq m})} \tag{6.36}$$

$$< \frac{P(V_{\geq m})}{3P(V_{\geq m})} \tag{6.37}$$

$$= \frac{1}{3}, \tag{6.38}$$

where (6.36) follows from $P(U') > 0$ since $K(U') = K(A) - K(U_{\geq m} \cup V_{\geq m}) > 2^{-(m-1)} - 2^{-(m-1)} = 0$; and (6.37) follows from $P(U_{\geq m}) < 2P(V_{\geq m})$ and Lemma 6.6.3.

By (6.35), (6.34), and (6.38), these conditions hold for $k = m - 1$.

Inductive Step: $1 \leq k \leq m - 1$. Notice that if $m = 1$, then the base case of $k = m - 1$ automatically proves the assertion for $k = 0$, so we may assume $m \geq 2$.

Assume that (6.31) – (6.33) hold for some positive integer $k \leq m - 1$ and for some $A \subseteq S$. We will show that there exists $A' \subseteq S$ that satisfies the three conditions above for some index $j \in \{0, \ldots, k-1\}$.

From the definition of m, we have $K(U_i) = K(V_i)$ for all $i < m$. In particular, $K(U_k) = K(V_k)$ since $k \leq m - 1$. Then $K(U_k) + K(V_k) \in \{0, 2^{-(k-1)}\}$, so $K(U_{\leq k} \cup V_{\leq k}) = K(U_{<k} \cup V_{<k}) + K(U_k \cup V_k)$ is an integer multiple of $2^{-(k-1)}$. Therefore, $K(S - U_{\leq k} \cup V_{\leq k}) = 1 - K(U_{\leq k} \cup V_{\leq k})$ is an integer multiple of $2^{-(k-1)}$. Since $0 < K(A) = 2^{-k} < 2^{-(k-1)}$, and $A \subseteq S - U_{\leq k} \cup V_{\leq k}$, Lemma 6.4.6 shows (taking $A \leftarrow A$, $B \leftarrow B$, $U \leftarrow S - (U_{\leq m} \cup V_{\leq m})$, $i = j \leftarrow k - 1$) there exists $B \subseteq S - (A \cup U_{\leq k} \cup V_{\leq k})$ with $K(B) = 2^{-(k-1)} - K(A) = 2^{-k}$.

Case 1: $K(U_k) = K(V_k) = 0$. Let $A' = A \cup B$. Then $K(A') = K(A) + K(B) = 2^{-(k-1)}$ and

$$U_{\geq k} \cup V_{\geq k} = U_{\geq k+1} \cup V_{\geq k+1} \subseteq A \subseteq A'$$

from (6.32). Also,

$$P(U_{\geq k}) - P(V_{\geq k}) = P(U_{\geq k+1}) - P(V_{\geq k+1}) < \frac{1}{3}P(A) < \frac{1}{3}P(A')$$

from (6.33) and $K(A) = K(A') - K(B) < K(A')$. Thus A' satisfies conditions (6.31) – (6.33) except the index k has been reduced to $j = k - 1 \geq 0$.

Case 2: $K(U_k) = K(V_k) = 2^{-k}$. Let $j \leq k$ be the smallest integer such that $K(U_i) = 2^{-i} = K(V_i)$ for all $i \in \{j, \ldots, k\}$. Then $j \geq 2$, since otherwise, $j = 1$ would imply $1 \geq K(U)+K(V) > 2K(U) \geq 2K(U_1) = 1$, a contradiction. Also, from the definition of j, we have $K(U_{j-1}) = K(V_{j-1}) = 0$. Let

$$A' = A \cup B \cup (U_{\geq j} - U_{>k}) \cup (V_{\geq j} - V_{>k}). \tag{6.39}$$

Then

$$\begin{aligned} U_{\geq j-1} \cup V_{\geq j-1} &= U_{\geq j} \cup V_{\geq j} \\ &= (U_{\geq j} - U_{>k}) \cup (V_{\geq j} - V_{>k}) \cup U_{\geq k+1} \cup V_{\geq k+1} \\ &\subseteq (U_{\geq j} - U_{>k}) \cup (V_{\geq j} - V_{>k}) \cup A \qquad (6.40) \\ &\subseteq A' \qquad (6.41) \end{aligned}$$

where (6.40) follows from (6.32); and

$$\begin{aligned} K(A') &= K(A) + K(B) + K(U_{\geq j} - U_{>k}) + K(V_{\geq j} - V_{>j}) \\ &= 2^{-k} + 2^{-k} + 2\sum_{i=j}^{k} 2^{-i} \\ &= 2^{-(j-2)}. \qquad (6.42) \end{aligned}$$

Now let $E \in \{A, B\}$ such that $P(E) = \min(P(A), P(B))$. Note $K(E) = 2^{-k}$, since $K(A) = K(B) = 2^{-k}$. Since

$$K((V_{\geq j} - V_{>k}) \cup E) = \sum_{i=j}^{k} 2^{-i} + K(E) = 2^{-(j-1)} - 2^{-k} + 2^{-k} = 2^{-(j-1)} > 2^{-j} = K(U_j),$$

strong monotonicity of Huffman-Kraft sums and Lemma 6.4.3 imply $P(U_j) \leq P(V_{\geq j} - V_{>k}) + P(E)$.

Suppose $j < k$. Then for all $i \in \{j + 1, \ldots, k\}$, we have $P(U_i) \leq P(V_{i-1})$ by strong monotonicity of

Huffman-Kraft sums, since $K(U_i) = 2^{-i} < 2^{-(i-1)} = K(V_{i-1})$. Thus

$$\begin{aligned} P(U_{\geq j+1} - U_{>k}) &= P(U_{j+1}) + \cdots + P(U_k) \\ &\leq P(V_j) + \cdots + P(V_{k-1}) \\ &= P(V_{\geq j} - V_{>k-1}). \end{aligned}$$

Therefore

$$\begin{aligned} P(U_{\geq j} - U_{>k}) &= P(U_j) + P(U_{\geq j+1} - U_{>k}) \\ &\leq P(V_{\geq j} - V_{>k}) + P(E) + P(V_{\geq j} - V_{>k-1}) \\ &= 2P(V_{\geq j} - V_{>k-1}) + P(V_k) + P(E). \end{aligned} \tag{6.43}$$

On the other hand, suppose $j = k$. Then $V_{\geq j} - V_{>k-1} = \varnothing$, so we also have

$$\begin{aligned} P(U_{\geq j} - U_{>k}) &= P(U_j) \\ &\leq P(V_{\geq j} - V_{>k}) + P(E) \\ &= 2P(V_{\geq j} - V_{>k-1}) + P(V_k) + P(E). \end{aligned} \tag{6.44}$$

Now we combine both the cases $j < k$ and $j = k$. Since $V_m \subseteq A$ and $K(V_m) = 2^{-m} < 2^{-k} = K(A)$, neither V_m nor $A - V_m$ is empty, so $|A| = |V_m| + |A - V_m| \geq 2$. Then since $K(A) = 2^{-k} = K(V_k)$, Lemma 6.4.5 shows $P(A) \leq 2P(V_k)$. Thus

$$P(E) = \min(P(A), P(B)) \leq \min(2P(V_k), P(B)),$$

and so

$$2P(V_k) + P(E) + P(B) \geq 3P(E). \tag{6.45}$$

Finally, we apply Lemma 6.6.3 by using the values

$$\begin{aligned} x &= P(U_{\geq j} - U_{>k}) \\ a &= -P(V_{\geq j} - V_{>k}) \\ b &= P(V_{\geq j} - V_{>k}) + P(B) \\ x' &= 2P(V_{\geq j} - V_{>k-1}) + P(V_k) + P(E). \end{aligned}$$

It is clear that $a < b$, and we have $x \leq x'$ from (6.43) (for $j < k$) and (6.44) (for $j = k$), Thus,

$$\begin{aligned} \frac{x+a}{x+b} &\leq \frac{x'+a}{x'+b} \\ \frac{P(U_{\geq j} - U_{>k}) - P(V_{\geq j} - V_{>k})}{P(U_{\geq j} - U_{>k}) + P(V_{\geq j} - V_{>k}) + P(B)} &\leq \frac{P(V_{\geq j} - V_{>k-1}) + P(E)}{3P(V_{\geq j} - V_{>k-1}) + 2P(V_k) + P(E) + P(B)} \\ &\leq \frac{P(V_{\geq j} - V_{>k-1}) + P(E)}{3P(V_{\geq j} - V_{>k-1}) + 3P(E)} \qquad (6.46) \\ &= \frac{1}{3}, \qquad (6.47) \end{aligned}$$

where (6.46) follows from (6.45). Therefore,

$$\begin{aligned} P(U_{\geq j-1}) - P(V_{\geq j-1}) &= P(U_{\geq j}) - P(V_{\geq j}) \\ &= P(U_{\geq j} - U_{>k}) - P(V_{\geq j} - V_{>k}) + P(U_{\geq k+1}) - P(V_{\geq k+1}) \\ &< \frac{1}{3}(P(U_{\geq j} - U_{>k}) + P(V_{\geq j} - V_{>k}) + P(B) + P(A)) \qquad (6.48) \\ &= \frac{1}{3}P(A'), \qquad (6.49) \end{aligned}$$

where (6.48) follows from (6.47) and (6.33); and (6.49) follows from (6.39). Thus A' satisfies the three conditions (6.31) – (6.32) by way of (6.42), (6.41), and (6.49), except the index k has been reduced to $j - 2 \in \{0, \ldots, k - 2\}$. ■

Theorem 6.6.6. *For any source, the competitive advantage of any prefix code over a Huffman code is less than* $\frac{1}{3}$.

Proof. Let C denote an arbitrary prefix code for the source. Let W and L denote the sets of wins and losses, respectively, of C over the Huffman code. It suffices to assume the competitive advantage of C over the Huffman code is positive, so $W \neq \varnothing$. Then Lemma 6.6.1 implies $L \neq \varnothing$, and Lemma 6.6.2 implies $K(W) < K(L)$. Therefore, $P(W) - P(L) < \frac{1}{3}$ by Lemma 6.6.5. ■

The following theorem shows that for any size at least four, sources can be found whose competitive

advantages over Huffman codes are arbitrarily close to $1/3$ and whose average lengths are arbitrarily close to that of a Huffman code.

Theorem 6.6.7. *For every $n \geq 4$, there exists a source of size n and a prefix code that has a competitive advantage over a Huffman code arbitrarily close to $\frac{1}{3}$ and the code's average length is arbitrarily close to that of the Huffman code.*

Proof. Let $n \geq 4$ and $\epsilon > 0$, and define $\alpha = \frac{\epsilon/2}{1-2^{-n+3}}$. Let the source be of size n and with symbol probabilities:

$$\begin{aligned} p_1 &= \frac{1}{3} + \epsilon \\ p_2 &= \frac{1}{3} \\ p_3 &= \frac{1}{3} - 2\epsilon \\ p_k &= \alpha 2^{4-k} \qquad (4 \leq k \leq n). \end{aligned}$$

One can verify that $p_1 + \cdots + p_n = 1$ and for each $k \in \{2, \ldots, n\}$ we have $p_k > p_{k+1} + \cdots + p_n$, so the Huffman code for the source assigns a word of length k to p_k for $k = 1, \ldots, n-1$, and also a word of length $n-1$ to p_n.

Define a prefix code C which is identical to the Huffman code, except that it reassigns p_1, p_2, and p_3 to codewords of lengths 3, 1, and 2, respectively. The code C will produce a shorter codeword than that of the Huffman code with probability $p_2 + p_3$ and will produce a longer codeword with probability p_1. Thus, the competitive advantage of C over the Huffman code is $\Delta = p_2 + p_3 - p_1 = \frac{1}{3} - 3\epsilon$.

Denote the codeword lengths of the Huffman code by l_i. The average length of the Huffman code is

$$1 \cdot \left(\frac{1}{3} + \epsilon\right) + 2 \cdot \left(\frac{1}{3}\right) + 3 \cdot \left(\frac{1}{3} - 2\epsilon\right) + \sum_{k=4}^{n} p_i l_i$$

and the average length of C is

$$1 \cdot \left(\frac{1}{3}\right) + 2 \cdot \left(\frac{1}{3} - 2\epsilon\right) + 3 \cdot \left(\frac{1}{3} + \epsilon\right) + \sum_{k=4}^{n} p_i l_i$$

so their difference is $-\frac{14}{3}\epsilon$.

In summary, the code C achieves a competitive advantage over the Huffman code of $\frac{1}{3} - 3\epsilon$ and has an average length at most $\frac{14}{3}\epsilon$ greater than that of the Huffman code. Taking ϵ arbitrarily small makes the competitive advantage approach $\frac{1}{3}$ and the average length difference approach zero. ■

6.7 Bound on competitive advantage over Shannon-Fano codes

On one hand, Shannon-Fano codes are efficient, since they suffice in proving Shannon's source coding theorem that says the average length of optimal block codes arbitrarily approaches from above the entropy of a source, as the block size grows. The proof uses the fact that the average length of a Shannon-Fano code is always within one bit of the source entropy, and so the average length per symbol of a Shannon-Fano code for a source block of size n is within $\frac{1}{n}$ bit of the source entropy.

One the other hand, Huffman codes are strictly better than Shannon-Fano codes in an average length sense for non-dyadic sources, and perform equally well for dyadic sources. Similarly, in a competitive sense, Theorem 6.5.1 showed that Huffman codes strictly competitively dominate Shannon-Fano codes if and only if the source is not dyadic.

The competitive advantage of one code over a Shannon-Fano code (or, actually, any other code) is trivially upper bounded by one, and the average length of a code can be at most one bit less than that of a Shannon-Fano code. The following theorem shows that there exist increasingly large sources with prefix codes that can approach both of these extremes over Shannon-Fano codes simultaneously.

Theorem 6.7.1. *For every positive integer n, there exists a source of size n and a prefix code that has a competitive advantage of at least $1 - 2^{-n+2}$ over a Shannon-Fano code for the source, and the code's average length is at least $1 - 2^{-n+2}$ less than the average length of the Shannon-Fano code.*

Proof. Let $\epsilon \in (0, 4^{-n})$ and let X be a source of size n whose probabilities are

$$p_k = \begin{cases} 2^{-k} - \epsilon & \text{if } 1 \leq k \leq n-1 \\ 2^{-n+1} + (n-1)\epsilon & \text{if } k = n. \end{cases}$$

Since $\left\lceil \log_2 \frac{1}{p} \right\rceil = m$ if and only if $2^{-m} \leq p < 2^{-m+1}$, a Shannon-Fano code for this distribution has codeword lengths

$$l_k = \left\lceil \log_2 \frac{1}{p_k} \right\rceil = \begin{cases} k+1 & \text{if } 1 \leq k \leq n-1 \\ n-1 & \text{if } k = n. \end{cases}$$

Note that l_n was determined from the fact that for all $n \geq 1$,

$$(n-1)4^{-n} < 2^{-n+1}. \tag{6.50}$$

Let C be a prefix code that assigns the word $1^{k-1}0$ to the outcomes that have probability p_k when $k < n$, and assigns the word 1^{n-1} to the outcome with probability p_n. This prefix code produces a shorter codeword than a Shannon-Fano code whenever $1 \leq k \leq n-1$, and ties when $k = n$, so its competitive advantage over a Shannon-Fano code is lower bounded as

$$\Delta = 1 - p_n = 1 - \frac{1}{2^{n-1}} - (n-1)\epsilon \geq 1 - \frac{1}{2^{n-1}} - \frac{n-1}{4^n} > 1 - \frac{1}{2^{n-2}}. \tag{6.51}$$

where (6.51) follows from (6.50).

The difference between the average lengths of the Shannon-Fano code and the code C is

$$(n-1)p_n + \sum_{k=1}^{n-1}(k+1)p_k - (n-1)p_n - \sum_{k=1}^{n-1} kp_k = \sum_{k=1}^{n-1} p_k = 1 - p_n, \tag{6.52}$$

the same quantity as the competitive advantage previously computed in (6.51).

■

In the preceding proof, the average length of the code C is at most 2^{-n+2} more than the source entropy, since

$$\begin{aligned} E[l_C(X))] &< E[l_{C_{SF}}(X))] - 1 + \frac{1}{2^{n-2}} && (6.53)\\ &< H(X) + 1 - 1 + \frac{1}{2^{n-2}} && (6.54)\\ &= H(X) + \frac{1}{2^{n-2}}. \end{aligned}$$

where (6.53) follows from (6.51) and (6.52); and (6.54) follows from Shannon's source coding theorem. We also note that the term 2^{-n+2} that occurs in the bounds of the theorem can be sharpened to be arbitrarily close to 2^{-n+1} but we chose to keep the proof simple instead.

6.8 Small codes

In this section, we analyze which sources of size at most 4 have competitively optimal Huffman codes.

Theorem 6.8.1. *Huffman codes are competitively optimal for all sources of size at most* 3*.*

Proof. If the source is of size 1 or 2, the result is trivial, so suppose the size is 3. Denote the source symbols by $1, 2, 3$ such that $P(1) \geq P(2) \geq P(3) > 0$. The word lengths of a Huffman code H are $l_H(1) = 1$ and $l_H(2) = l_H(3) = 2$. Let C denote any other prefix code, and use the notation W and L, as in (6.1). It is not

possible for $1 \in W$, since $l_C(1) \geq 1 = l_H(1)$. If $2 \in W$, then $l_C(2) = 1$ and therefore $l_C(1), l_C(3) \geq 2$, so $1 \in L$ and $3 \notin W$, which implies the competitive advantage of C over the Huffman code is $\Delta = P(W) - P(L) = P(2) - P(L) \leq P(2) - P(1) \leq 0$. Alternatively, if $3 \in W$, then we similarly conclude $\Delta \leq 0$. Finally, if $2, 3 \notin W$, then $P(W) = 0$, so $\Delta \leq 0$. ■

Theorem 6.8.2. *Let Q be the hexahedron with vertices $(\frac{1}{2}, \frac{1}{2}, 0)$, $(\frac{2}{5}, \frac{1}{5}, \frac{1}{5})$, $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, $(\frac{1}{3}, \frac{1}{3}, \frac{1}{6})$, $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$. For every source of size 4 with probabilities $p_1 \geq p_2 \geq p_3 \geq p_4 > 0$, a Huffman code is competitively optimal if the triple (p_1, p_2, p_3) lies in the exterior of Q, and is not competitively optimal if the triple lies in the interior of Q.*

Proof. Denote the source symbols by $1, 2, 3, 4$ and their probabilities by p_1, p_2, p_3, p_4, respectively. We will determine conditions on $p_1, \ldots, p_4$ such that there exists a prefix code with a positive competitive advantage over a Huffman code. It suffices to consider complete prefix codes, since any non-complete prefix code contains at least one codeword that could be shortened without decreasing its competitive advantage. The only possible codeword length distributions for such size-4 codes are $1, 2, 3, 3$ and $2, 2, 2, 2$. In either case, the Huffman algorithm merges the source symbols 3 and 4 to form a new symbol with probability $p_3 + p_4$.

Suppose $p_3 + p_4 > p_1$. Then the Huffman algorithm merges 1 and 2 and then the $(3, 4)$ symbol is merged with the $(1, 2)$ symbol to get a balanced tree with codeword lengths $2, 2, 2, 2$. If a size-4 prefix code achieves a positive competitive advantage over this Huffman code, then it must have codeword lengths $1, 2, 3, 3$, for otherwise only ties would occur. In this case, the competitive advantage would be the probability of the new code's length-1 word minus the sum of the probabilities of its two length-3 words, which equals $p_1 - (p_3 + p_4) < 0$, so in fact the new code would be strictly competitively dominated by the Huffman code. The competitive advantage would still not be positive even if $p_3 + p_4 = p_1$ and the Huffman algorithm created codewords with lengths $2, 2, 2, 2$.

Alternatively, assume $p_3 + p_4 \leq p_1$ with the Huffman algorithm merging the $(3, 4)$ symbol with 2, and then merging the resulting $(2, (3, 4))$ symbol with 1. The resulting codeword lengths are $1, 2, 3, 3$. The competitive advantage of any depth-2 balanced tree over the Huffman code would be $p_3 + p_4 - p_1 \leq 0$, so such codes are competitively dominated by the Huffman code. Thus, any code C with a positive competitive advantage Δ over the Huffman code must have lengths $1, 2, 3, 3$, and hence C just permutes the Huffman code's assignment of codeword lengths to source symbols.

Suppose $l_C(1) = 1$. If $l_C(2) = 2$, then $\Delta = 0$. If $l_C(2) = 3$ and $l_C(3) = 2$, then $\Delta = p_3 - p_2 \leq 0$. If $l_C(2) = 3$ and $l_C(4) = 2$, then $\Delta = p_4 - p_2 \leq 0$.

Alternatively, suppose $l_C(1) \neq 1$. There are 9 possible cases for $(l_C(1), l_C(2), l_C(3), l_C(4))$:

(2,1,3,3): $\Delta = p_2 - p_1 \leq 0$

(2,3,1,3): $\Delta = p_3 - p_1 - p_2 \leq 0$

(2,3,3,1): $\Delta = p_4 - p_1 - p_2 \leq 0$

(3,2,1,3): $\Delta = p_3 - p_1 \leq 0$

(3,2,3,1): $\Delta = p_4 - p_1 \leq 0$

(3,3,1,2): $\Delta = p_3 + p_4 - p_1 - p_2 \leq 0$

(3,3,2,1): $\Delta = p_3 + p_4 - p_1 - p_2 \leq 0$

(3,1,2,3): $\Delta = p_2 + p_3 - p_1$

(3,1,3,2): $\Delta = p_2 + p_4 - p_1$.

So the only codes C that can yield $\Delta > 0$ are the cases $(3, 1, 2, 3)$ and $(3, 1, 3, 2)$.

Let us denote the following inequalities:

$$(I1): \ p_1 \geq p_2$$

$$(I2): \ p_2 \geq p_3$$

$$(I3): \ p_3 \geq p_4$$

$$(I4): \ p_4 > 0$$

$$(I5): \ p_3 + p_4 \leq p_1$$

$$(I6): \ p_2 + p_3 > p_1$$

$$(I7): \ p_2 + p_4 > p_1.$$

Inequalities (I1) – (I6) determine a set in $\mathbb{R}^3$ whose interior is a hexahedron specified by the 5 vertices $(\frac{1}{2}, \frac{1}{2}, 0)$, $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, $(\frac{2}{5}, \frac{1}{5}, \frac{1}{5})$, $(\frac{1}{3}, \frac{1}{3}, \frac{1}{6})$, $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$. The first 3 vertices satisfy $\sim$(I7) with equality, the 4^{th} vertex satisfies (I7), and 5^{th} vertex satisfies $\sim$(I7). Therefore, the hexahedron is cut into two tetrahedra by (I7) and is known as a triangular dipyramid.

The Huffman code is competitively optimal in the exterior of this hexahedron, is not competitively optimal in the interior of this hexahedron, and is sometimes competitively optimal on the boundary. ■

Corollary 6.8.3. *If a source of size 4 is chosen uniformly at random from a flat Dirichlet distribution, then the probability its Huffman code is competitively optimal is $2/3$.*

Proof. The hexahedron in Theorem 6.8.2 is a union of two tetrahedra, whose volumes are computed using deter-

minants as (e.g., [13])

$$\frac{1}{6}\cdot\begin{vmatrix} 1/2 & 1/2 & 0 & 1 \\ 2/5 & 1/5 & 1/5 & 1 \\ 1/3 & 1/3 & 1/3 & 1 \\ 1/3 & 1/3 & 1/6 & 1 \end{vmatrix} = \frac{1}{6}\cdot\frac{1}{180} \qquad \frac{1}{6}\cdot\begin{vmatrix} 1/2 & 1/2 & 0 & 1 \\ 2/5 & 1/5 & 1/5 & 1 \\ 1/3 & 1/3 & 1/3 & 1 \\ 2/5 & 1/5 & 1/5 & 1 \end{vmatrix} = \frac{1}{6}\cdot\frac{1}{120}\ .$$

The set of all p_1, p_2, p_3, p_4 satisfying $p_1 \geq p_2 \geq p_3 \geq p_4 > 0$ and $p_1 + p_2 + p_3 + p_4 = 1$ is determined by the 4 inequalities

$$\begin{aligned} p_1 &\geq p_2 \\ p_2 &\geq p_3 \\ p_1 + p_2 + 2p_3 &\geq 1 \\ p_1 + p_2 + p_3 &< 1. \end{aligned}$$

These form a tetrahedron with vertices $(1,0,0)$, $(\frac{1}{2},\frac{1}{2},0)$, $(\frac{1}{3},\frac{1}{3},\frac{1}{3})$, $(\frac{1}{4},\frac{1}{4},\frac{1}{4})$ whose volume is

$$\frac{1}{6}\cdot\begin{vmatrix} 1 & 0 & 0 & 1 \\ 1/2 & 1/2 & 0 & 1 \\ 1/3 & 1/3 & 1/3 & 1 \\ 1/4 & 1/4 & 1/4 & 1 \end{vmatrix} = \frac{1}{6}\cdot\frac{1}{24}\ .$$

Thus the probability of randomly selecting a source from a flat Dirichlet distribution whose Huffman code is not competitively optimal is $(\frac{1}{180} + \frac{1}{120})/\frac{1}{24} = \frac{1}{3}$. So the probability the Huffman code is competitively optimal is $\frac{2}{3}$. ■

6.9 Experimental evidence

We demonstrate numerically that if a source is chosen at random, then as the source size grows, the probability becomes nearly zero that a Huffman code will be competitively optimal. Experimentally, this probability is less than 1% when the source size is at least 20. That is, with near certainty each Huffman code will be competitively dominated by some other prefix code as the source size increases. This indicates that for most sources, from a competitive advantage point of view, there really is no "best" code to use. Each code can be strictly competitively dominated by another in never-ending cycles of code sequences.

One way to generate source probabilities $p_1, \ldots, p_n$ chosen according to a flat Dirichlet distribution is to choose n points independently and uniformly on a circle of circumference 1 and then use the n arc lengths between neighboring points as the desired probabilities. Such a procedure treats all sources equally and indeed yields interesting results.

For any source of size n, exhaustively checking whether each complete prefix code competitively dominates the Huffman code appears to become a computationally infeasible task as n grows, since the number of such prefix codes grows quickly. However, Lemma 6.3.4 gives a sufficient condition for a Huffman tree to not be competitively optimal, which allows us to obtain a lower bound on the probability that a Huffman code is not competitively optimal for a given source. Thus we can randomly select many sources and determine if such a condition holds, in which case we can then declare the Huffman code not competitively optimal. This suboptimal condition turns out to be overwhelmingly sufficient to observe that the probability is practically zero that the Huffman code of a randomly chosen source is competitively optimal even for relatively small source sizes.

For each source size $n \in \{3, \ldots, 34\}$, we generated 10^6 sources from a flat Dirichlet distribution, i.e., chosen uniformly at random on the $(n-1)$-dimensional simplex embedded in $\mathbb{R}^n$. For each such source we determined whether the sufficient condition of Lemma 6.3.4 was satisfied. Fig 6.2 plots for each n the fraction of the randomly generated sources that satisfied the sufficient condition. That is, the true fraction of the randomly generated sources for which a Huffman code was not competitively optimal lies above the plotted curve. The observed lower bound curve quickly tends toward 1, so the true fraction of the randomly generated sources with competitively non-optimal Huffman codes tends toward 1 as well.

For the case of $n = 3$, Theorem 6.8.1 guarantees that 100% of the randomly chosen sources will have competitively optimal Huffman codes, which is exactly what was observed experimentally.

For the case $n = 4$, Corollary 6.8.3 gives a $2/3$ probability of a randomly chosen source to have a competitively optimal Huffman code. The experimentally observed upper bound was 66.6992%.

For $n \geq 5$, one can see that the probability a randomly chosen source has a competitively optimal Huffman code rapidly decreases towards 0, and in fact no such competitively optimal Huffman codes were observed out of the million chosen for each $n \geq 31$.

Chapter 6 is a reprint of the material as it appears in: S. Congero and K. Zeger, "Competitive advantage of Huffman and Shannon-Fano codes", submitted to *IEEE Transactions on Information Theory*, November 13, 2023.

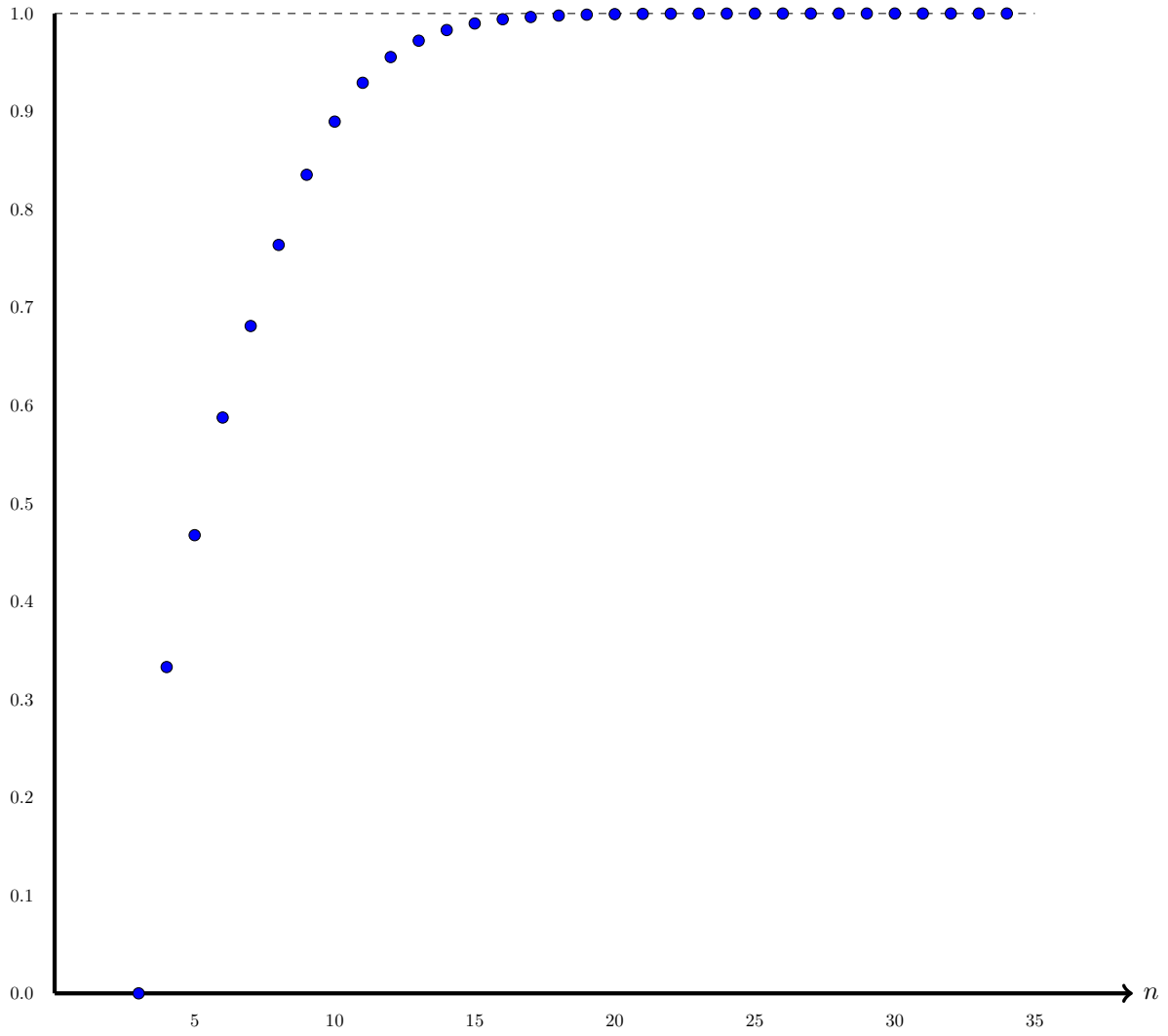


Figure 6.2. Lower bound on the fraction of 10^6 randomly chosen sources whose Huffman code is not competitively optimal, as a function of the source size n. For $n = 15$ Huffman codewords, about 99% of randomly selected sources did not have competitively optimal Huffman codes. For $n \geq 31$, all 10^6 randomly chosen sources had Huffman codes that were not competitively optimal.

References

[1] R. Bell and T. M. Cover, "Competitive optimality of logarithmic investment", *Mathematical Operations Research*, vol. 5, no. 2, pp. 161 – 166, May 1980.

[2] J. Berstel, D. Perrin, and C. Reutenauer, *Codes and Automata*, Encyclopedia of Mathematics and its Applications, Cambridge University Press, 2009.

[3] J.R. Bhatnagar, "Competitive optimality: A novel application in evaluating practical AI Systems", *Engineering Applications of Artificial Intelligence*, vol. 102, article 104241, June 2021.

[4] P. Billingsley, *Probability and Measure*, John Wiley & Sons, New York, 1986.

[5] S. Congero and K. Zeger, "Characterizations of minimal expected length codes", *IEEE Transactions on Information Theory*, (submitted on November 13, 2023). Also available at: arXiv:2311.07007 [cs.IT].

[6] T. M. Cover, "On the competitive optimality of Huffman codes", *IEEE Transactions on Information Theory*, vol. 37, no. , pp. 172 – 174, January 1991.

[7] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd edition, New Jersey, Wiley-Interscience, 2006.

[8] L. Devroye, *Non-Uniform Random Variate Generation*, Springer-Verlag, New York, 1986.

[9] M. Feder, "A note on the competitive optimality of Huffman codes", *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 436 – 439, March 1992.

[10] R. G. Gallager, "Variations on a theme by Huffman", *IEEE Transactions on Information Theory*, vol. 24, no. 6, pp. 668 – 8674, November 1978.

[11] M. Khosravifard, H. Saidi, M. Esmaeili, and T. A. Gulliver, "The minimum average code for finite memoryless monotone sources", *IEEE Transactions on Information Theory*, vol. 53, no. 3, pp. 955 – 975, March 2007.

[12] T. Linder, V. Tarokh, K. Zeger, "Existence of optimal prefix codes for infinite source alphabets", *IEEE Transactions on Information Theory*, vol. 43, no. 6, pp. 2026 – 2028, November 1997.

[13] H. B. Newson, "Volume of a polyhedron", *Annals of Mathematics*, vol. 1, no. 1/4, pp. 108 – 110, September 1899.

[14] K. W. Ng, G.-L. Tian, and M.-L. Tang, *Dirichlet and Related Distributions: Theory, Methods and Applications*, John Wiley & Sons, 2011.

[15] P. Rastegari, M. Khosravifard, H. Narimani, and T. A. Gulliver, "On the structure of the minimum average redundancy code for monotone sources", *IEEE Communications Letters*, vol. 18, no. 4, pp. 664 – 667, April 2014.

[16] H. Yamamoto and T. Itoh, "Competitive optimality of source codes", *IEEE Transactions on Information Theory*, vol. 41, no. 6, pp. 2015 – 2019, November 1995.

[17] H. Yamamoto and H. Yokoo, "Average-sense optimality and competitive optimality for almost instantaneous VF codes", *IEEE Transactions on Information Theory*, vol. 47, no. 6, pp. 2174 – 2184, September 2001.