國立交通大學

電信工程研究所

碩士論文

唯一可解一對一編碼之分析與實作

Analysis and Practice of Uniquely Decodable One-to-One Code

研 究 生：劉勁甫

指導教授：陳伯寧 教授

共同指導教授：陸曉峰 教授

中 華 民 國 一0二 年 七 月

唯一可解一對一編碼之分析與實作
Analysis and Practice of Uniquely Decodable One-to-One Code

研 究 生：劉勁甫　　Student：Chin-Fu Liu

指導教授：陳伯寧　　Advisor：Po-Ning Chen

共同指導教授：陸曉峰　　Co-Advisor : Hsiao-feng (Francis) Lu

國 立 交 通 大 學

電信工程研究所

碩 士 論 文

A Thesis

Submitted to Institute of Computer and Information Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of Master

in

Communication Engineering

July 2013

Hsinchu, Taiwan, Republic of China

中華民國一O二年七月

唯一可解一對一編碼之分析與實作

學生：劉勁甫　　指導教授：陳伯寧 陸曉峰

國立交通大學電信工程研究所碩士班

摘　　要

在這篇論文中，我們提出可藉由在連續的一對一字碼(One-to-One Code)之間安插特定字元(Unique Word)來分開一對一字碼，使得連續的一對一字碼符合唯一可解(Uniquely Decodable)性質，藉此產生一種新的編碼方式，我們稱為「唯一可解一對一碼」(Uniquely Decodable One-to-One Code)。我們接著分析唯一可解一對一碼的組合特性與壓縮能力，並由此推導出給定長度 n 的唯一可解一對一碼的個數公式。經由此公式，我們即可計算給定統計特性之信息源的唯一可解一對一碼的最佳平均碼長。我們另外推導出三個平均碼長的上限公式，與兩個平均碼長的極限值的上限公式，據此，我們證明當特定字元(Unique Word)的長度與信息源字元的整合個數皆趨近無限大時，唯一可解一對一碼的平均碼長的極限值可以達到信息源的理論極限熵值(Entropy)。論文接著提供在不同條件下的快速編碼與快速解碼的演算法。我們的數值實驗結果顯示，相對於赫夫曼碼(Huffman Code)，在相同的解碼複雜度的前提下，唯一可解一對一碼具有可與其相比的低壓縮率。相對於藍柏吉夫碼(Lembel-Ziv Code)，唯一可解一對一碼則具有較佳的平均碼長，此數值結果說明了唯一可解一對一碼在實際應用的可行性。

Analysis and Practice of Uniquely Decodable One-to-One Code

Student: Chin-Fu Liu Advisors: Po-Ning Chen, Hsiao-feng (Francis) Lu

Institute of Communications Engineering
National Chiao Tung University

ABSTRACT

In this thesis, we consider the so-called *uniquely decodable one-to-one code* (UDOOC) that is formed by inserting a "comma" indicator, termed the *unique word* (UW), between consecutive one-to-one codewords for separation. Along this research direction, we first investigate the general combinatorial properties of UDOOCs, in particular the enumeration of UDOOC codewords for any (finite) codeword length. Based on the obtained formula on the number of length-n codewords for a given UW, the average codeword length of the optimal UDOOC for a given source statistics can be computed. Upper bounds on the average codeword length of UDOOCs are next established. The analysis on bounds of the average codeword length then leads to two asymptotic bounds on ultimate per-letter average codeword length, one of which is achievable and hence tight for a certain source statistics and UW, and the other of which proves the achievability of source entropy rate of UDOOCs when both the block size of source letters for UDOOC compression and UW length go to infinity. Efficient encoding and decoding algorithms for UDOOCs are subsequently given. Numerical results show that when grouping three English letters as a block, the UDOOCs with UW = 0001, 0000, 000001 and 000000 can respectively reach the compression rates of 3.531, 4.089, 4.115 and 4.709 bits per English letter (with the lengths of UWs included), where the source stream to be compressed is the book titled *Alice's Adventures in Wonderland*. In comparison with the first-order Huffman code, the second-order Huffman code, the third-order Huffman code and the Lembel-Ziv code, which respectively achieve the compression of 3.940, 3.585, 3.226 and 6.028 bits per single English letter, the proposed UDOOCs can potentially in comparable compression rate to the Huffman code under similar decoding complexity and yield a better average codeword length than that of the Lempel-Ziv code, thereby confirming the practicability of the simple idea of separating OOC codewords by UWs.

Acknowledgements

First and foremost, I would like to express my sincerest gratitude to my advisor Professor Po-Ning Chen and co-advisor Professor Francis Lu, who provided me with invaluable and resourceful guidance in research. Professor Chen always looked after me not only in my master program but also when I studied in my bachelor program. He always kindly and patiently assists me and my entire research with his inspirational and enlightening instruction. On the other hand, Professor Francis Lu shows me a favorable and superior model as a scholar. His keen and vigorous academic observation illuminates me in this thesis. It is no exaggeration to say that without their exceedingly impressive and wonderful kindness and patience, I could not have completed my thesis.

I shall extend my thanks to all the members in NTL lab and all my friends for their friendship in the past two years, especially, to Mr. Ting-Yi Wu, who taught me how to use LaTeX and gave me a multitude of assistances when I joined NTL lab.

Last but not least, I would like to thank my senior high school chemistry teacher and her husband, who provided me with spiritual support. They were always concerned about my health and happiness even after I graduated from senior high school.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Investigation of lossless source coding can be roughly classified into two categories, one for the compression of a sequence of source letters and the other for a single "one shot" source symbol [6]. A well-known representative for the former is the Huffman code, while the latter is usually referred to as the one-to-one code (OOC).

The Huffman code is an optimal entropy code that can achieve the minimum average codeword length for a given statistics of source letters. It obeys the rule of unique decodability and hence concatenation of Huffman codewords can be uniquely recovered by the decoder. Although optimal in principle, it may encounter several obstacles in implementation. For example, the rare codewords are exceedingly long in length, thereby hampering the efficiency of decoding. Other practical obstacles include *i*) the codebook needs to be *a priori* stored for encoding and decoding, which may demand a large space for sources with moderately large alphabet size, *ii*) for a sequence of codewords, the decoding must be sequential, not in parallel, and *iii*) decoding error will propagate across subsequent codewords.

In contrast to unique decodability, the OOC only requires to assign a distinct codeword to each source symbol. It has been studied since 1970s [32] and has been shown to achieve an average codeword length smaller than the source entropy minus a nontrivial amount of

quantity called *anti-redundancy* [28]. Various research works over the years have shown that the anti-redundancy can be as big as the logarithm of the source entropy [1, 4, 5, 12, 20, 23, 25, 26, 28, 29, 30]. In comparison with an entropy coding like Huffman, the codewords of an OOC can be sequenced alphabetically and hence the practice of an OOC is generally considered more computationally convenient.

A question that may arise from the above discussion is whether we could add a "comma" indicator, termed *Unique Word* (UW) in this thesis, in-between consecutive OOC codewords, and use the OOC for lossless compression of a sequence of source letters. A direct merit of such a structure is that the alphabetically sequenced OOC codeword can be manipulated without *a priori* stored codebook at both the encoding and decoding ends. This is however achieved at a price of an additional constraint that the "comma" indicator must not appear as an internal subword[1] in the concatenation of either an OOC codeword with a comma indicator, or a comma indicator with an OOC codeword.

On the one hand, this additional constraint facilitates the fast identification of OOC codewords in a coded bit-stream and makes feasible the subsequent parallel decoding of them. On the other hand, the achievable average codeword length of an UW-forbidden OOC may increase significantly for a bad choice of UWs. Therefore, it is of theoretical importance to investigate the minimum average codeword length of a UW-forbidden OOC, in particular the selection of a proper UW that could minimize this quantity. Since the resultant UW-forbidden OOC coding system satisfies unique decodability (UD), we will refer to it conveniently as the UDOOC in the sequel.

We would like to point out that the conception of inserting UWs among consecutive words might not be new in existing practical applications. For example, in certain situations, an

[1]We say $\boldsymbol{a} = a_1 \dots a_m$ is not an *internal subword* of $\boldsymbol{b} = b_1 \dots b_n$ if there does not exist i such that $b_i \dots b_{i+m-1} = \boldsymbol{a}$ for all $1 < i < n - m + 1$. When the same condition holds for all $1 \leq i \leq n - m + 1$ (i.e., with two equalities), we say $\boldsymbol{a}$ is not a *subword* of $\boldsymbol{b}$.

entity similar to UWs has been inserted into a bit-stream as a boundary indicator for a frame, or as a synchronization support, or as a part of error control mechanism. However, a complete theoretical study of the UDOOC conception remains undone, which is the main target of this thesis.

We now give a formal definition of binary UDOOCs.

Definition 1. *Given UW* $\boldsymbol{k} = k_1 k_2 \dots k_L \in \mathbb{F} \times \cdots \times \mathbb{F} = \mathbb{F}^L$, *where* $\mathbb{F} = \{0, 1\}$, *we say* $\mathcal{C}_{\boldsymbol{k}}(n)$ *is a UDOOC of length* $n \geq 1$ *associated with* $\boldsymbol{k}$ *if it contains all binary length-*n *tuples* $\boldsymbol{b} = b_1 \dots b_n$ *such that* $\boldsymbol{k}$ *is not an internal subword of the concatenated bit-stream* $\boldsymbol{kbk}$. *As a special case, we set* $\mathcal{C}_{\boldsymbol{k}}(0) := \{null\}$.[2] *The overall UDOOC associated with* $\boldsymbol{k}$, *denoted by* $\mathcal{C}_{\boldsymbol{k}}$, *is given by*

$$\mathcal{C}_{\boldsymbol{k}} := \bigcup_{n \geq 0} \mathcal{C}_{\boldsymbol{k}}(n).$$

For a better comprehension of Definition 1, we next give an example to illustrate how a UDOOC is generated and how it is used in encoding and decoding. The way to count the number of length-n UDOOC codewords will follow.

Example 1. *Suppose a UW* $\boldsymbol{k} = 00$ *is chosen. In order to prohibit an alternate concatenation of UWs and UDOOC codewords from containing 00 as an internal subword, the following constraints must be satisfied.*

- *Type-I constraints: The UW cannot be a subword of any UDOOC codeword. This means that within any codeword of length* $n \geq 1$*:*

  (C1) "0" can only be followed by "1".

  (C2) "1" can be followed by either "0" or "1".

[2] In our binary UDOOC, it is allowable to place two UWs side-by-side with nothing in-between in order to produce a *null* codeword.

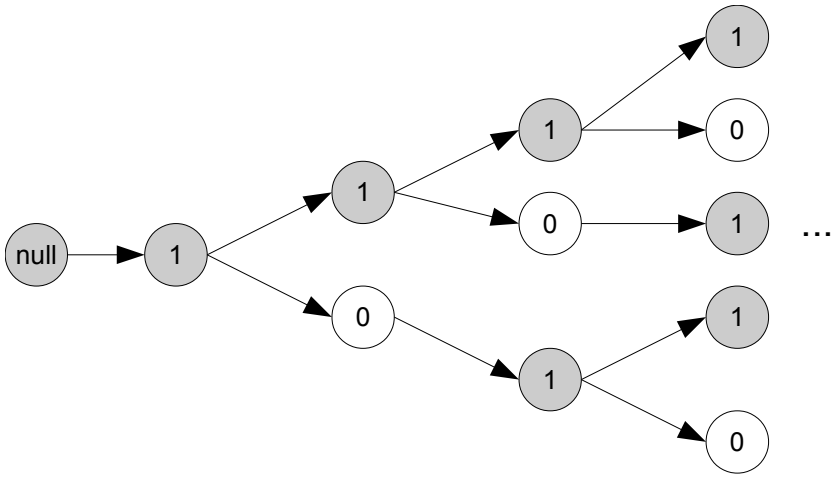

Figure 1.1: UDOOC code tree for UW = 00.

- *Type-II constraints: Besides the type-I constraints, the UW cannot be an internal subword of the concatenation of UWs and UDOOC codewords, regardless of the ordering. This implies that except for the "null" codeword:*

  (C3) The first bit of a codeword cannot be "0".

  (C4) The last bit of a codeword cannot be "0".

By Constraints (C1)-(C4), we can place the UDOOC codewords on a code tree as shown in Fig. 1.1, in which a path starting from the root node and ending at a gray-shaded node corresponds to a codeword. Thus, the codewords for UW = 00 include null, 1, 11, 101, 111, 1011, 1101, 1111, etc. It should be noted that we only show the codewords of length up to four, while the code tree actually can grow indefinitely in depth.

At the decoding stage, suppose the received bit-stream is 00100110010100111100, where we add UWs both at the beginning and at the end to indicate the margins of the bit-stream. This may facilitate, for example, its noncoherent transmission. Then, the decoder first locates UWs and parses the bit-stream into separate codewords as 1, 11, 101 and 1111, after which the four codewords can be decoded separately (possibly in parallel) to their respective source

symbols.

With the code tree representation, the number of length-n codewords in a UDOOC code tree can be straightforwardly calculated. Let the "null"-node be located at level 0*. For* $n \geq 1$*, denote by* a_n *and* b_n *the numbers of "1"-nodes and "0"-nodes at the nth level of the code tree, respectively. By the two type-I constraints, the following recursions should hold:*

$$\begin{cases} a_n &= a_{n-1} + b_{n-1} \\ b_n &= a_{n-1} \end{cases} \quad \text{for } n \geq 2.$$

With the initial values of $a_1 = 1$ *and* $a_2 = 1$*, it follows that* $\{a_n\}_{n=1}^{\infty}$ *is the renowned Fibonacci sequence [13], i.e.,* $a_n = a_{n-1} + a_{n-2}$ *for* $n \geq 3$*. This result, together with the two type-II constraints, implies that the number of length-n codewords is* $|\mathcal{C}_{00}(n)| = a_n$ *for* $n \geq 1$*, which according to the Fibonacci recursion is given by:*

$$a_n = \frac{\varphi^n - \bar{\varphi}^n}{\sqrt{5}},$$

where $\varphi = \frac{1+\sqrt{5}}{2}$ *is the Golden ratio and* $\bar{\varphi} = \frac{1-\sqrt{5}}{2}$ *is the Galois conjugate of* φ *in number field* $\mathbb{Q}(\sqrt{5})$*. Thus,* $|\mathcal{C}_{00}(n)|$ *grows exponentially with base* $\varphi \approx 1.618$*.* □

We can similarly examine the choice of UW = 01 and draw the respective code tree as shown in Fig. 1.2, where its type-I constraints become:

(C1) "0" can only be followed by "0".

(C2) "1" can be followed by either "0" or "1".

and no type-II constraints are resulted. We then obtain

$$\begin{cases} a_n &= a_{n-1} \\ b_n &= a_{n-1} + b_{n-1} \end{cases} \quad \text{for } n \geq 2,$$

and $|\mathcal{C}_{01}(n)| = a_n + b_n = n + 1$. It is obvious that although from Figs. 1.1 and 1.2, taking UW = 01 seems to provide more codewords than taking UW = 00 at small n, the linear

growth rate of $|\mathcal{C}_{01}(n)|$ with respect to the codeword length suggests that such choice would give no code rate asymptotically and will be outperformed by the choice of UW $= 00$ when n is only moderately large.

The above two exemplified UWs point to an important fact that the best UW, which minimizes the average codeword length, depends on the code size required. Thus, investigation of the efficiency of a UW may need to consider the transient superiority in addition to claiming the asymptotic winner.

In this thesis, we provide efficient encoding and decoding algorithms for UDOOCs, and investigate their general combinatorial properties, in particular the enumeration of codewords for any (finite) codeword length. Based on the obtained formula on $|\mathcal{C}_{\boldsymbol{k}}(n)|$, i.e., the number of length-n codewords for a given UW $\boldsymbol{k}$, the average codeword length of the optimal UDOOC for a given source statistics can be computed. Classifications of UWs are followed, where two types of equivalences are specified, which are (exact) *equivalence* and *asymptotic equivalence*. UWs that are equivalent in the former sense requires to have exactly the same minimum average codeword length for every source statistics, while asymptotic equivalence only dictates the UWs to result in the same ultimate code rate as codeword length approaches infinity. Enumeration of the number of asymptotic equivalent UW classes are then studied with the help of the methodologies in [16] and [24]. Besides, three upper bounds on the average codeword length of UDOOCs are established. The first one is a general upper bound when only the largest probability of source symbols is given. The second upper bound refines the first one under the premise that the source entropy is additionally known. When both the largest and second largest probabilities of source symbols are present apart from the source entropy, the third upper bound can be used. Since these bounds are derived in terms of different techniques, actually none of the three bounds dominates the other two for all statistics. Comparison of these bounds for an English text with statistics from [35] and

that with statistics from *Alice's Adventures in Wonderland* will be accordingly provided. The analysis on bounds of the average codeword length is ended up with two asymptotic bounds on ultimate per-letter average codeword length, one of which is tight for a certain choices of source statistics and UW, and the other of which leads to the achievability of the ultimate per-letter average codeword length to the source entropy rate when both the source block length for compression and UW length tend to infinity.

It may be of interest to note that the numeration of codewords, i.e., $c_{\boldsymbol{k},n} = |\mathcal{C}_{\boldsymbol{k}}(n)|$, is actually obtained indirectly via the determination of an auxiliary quantity $s_{\boldsymbol{k},n}$, which is the number of words that satisfy the type-I constraints but not necessarily the type-II constraints. By utilizing the Goulden-Jackson cluster method [15, 19, 21, 22, 31], the explicit formula for $s_{\boldsymbol{k},n}$ can be established. The desired enumeration formula for the number of length-n UDOOC codewords is then obtained by proving that both the so-called linear constant coefficient difference equation (LCCDE) and the asymptotic growth rate of $s_{\boldsymbol{k},n}$ and $c_{\boldsymbol{k},n}$ are identical. We conclude from the obtained formulas that the all-zero UW has the largest asymptotic growth rate among all UWs of the same length, while the UW with the smallest growth rate is $00\ldots01$. Interestingly, the all-zero UW is often the one that has the worst $c_{\boldsymbol{k},n}$ for small n, as contrary to UW $00\ldots01$, whose $c_{\boldsymbol{k},n}$ tops all other UWs when n is small. We then demonstrate by using these two special UWs that the general encoding and decoding algorithms can be considerably simplified when further taking into consideration the structure of particular UWs. A side result from the enumeration of $c_{\boldsymbol{k},n}$ is that for all UWs, the codeword growth rate of UDOOCs will tend to $|\mathbb{F}| = 2$ as the length of the UW goes to infinity.

Numerical results show that when grouping three English letters as a block and separating the consecutive blocks by UWs, the UDOOCs with UW = 0001, 0000, 000001 and 000000 can respectively reach the compression rates of 3.531, 4.089, 4.115, 4.709 bits per English

letter (with the length of UWs included), where the source stream to be compressed is the book titled *Alice's Adventures in Wonderland*. In comparison with the first-order Huffman code, the second-order Huffman code, the third-order Huffman code[3] and the Lempel-Ziv code, which respectively achieve the compression rates of 3.940, 3.585, 3.226 and 6.028 bits per English letter, the proposed UDOOCs can potentially result in comparable compression rate to the Huffman code under similar decoding complexity and yield a better average codeword length than that of the Lempel-Ziv code, thereby confirming the practicability of the simple idea of separating OOC codewords by UWs.

In the literature, there are a number of publications working on enumeration of words in a set that forbids the appearance of a specific pattern [7, 8, 9, 10, 11]. For example, Doroslova investigated the number of binary length-n words, in which a specific subword like $1010\ldots10$ is not allowed [9]. He then extended the result to non-binary alphabet and forbidden subwords of length 3 [8, 11], and forbidden subwords of length 4 [10], as well as the so-called "good" forbidden subwords [7]. The analyses in [7, 8, 9, 10, 11] however depend on the specific structure of forbidden subwords considered and no asymptotic examination is performed. On the other hand, algorithmic approaches have been devoted to a problem of similar (but not the same) kind, one of which is called the Goulden-Jackson cluster method [15, 21, 22, 19, 31].

Instead of enumerating the number of words internally without a forbidden pattern, some researches investigate the inherent characteristic of such patterns. In this literature, Rivals and Rahmann [24] provide an algorithm to account for the number of overlaps[4] for a given

[3] A kth-order Huffman code maps a block of k source letters onto a variable-length codeword.

[4] In [16] and [24], the authors actually use a different name "autocorrelation" for "overlap" originated from [15]. Specifically, they define the *autocorrelation* $\boldsymbol{v} = v_1 \cdots v_L$ of a binary length-L string $\boldsymbol{u} = u_1 \cdots u_L$ as a binary zero-one bit-stream of length L such that $v_i = 1$ if i is a period of $\boldsymbol{u}$, where i is said to be a period of $\boldsymbol{u}$ when $u_j = u_{i+j}$ for every $1 \leq j \leq L - i$. Since the term *autocorrelation* is extensively used in other literature ilke digital communications to illustrate similar but different conception, we adopt the name of "overlap" in this thesis.

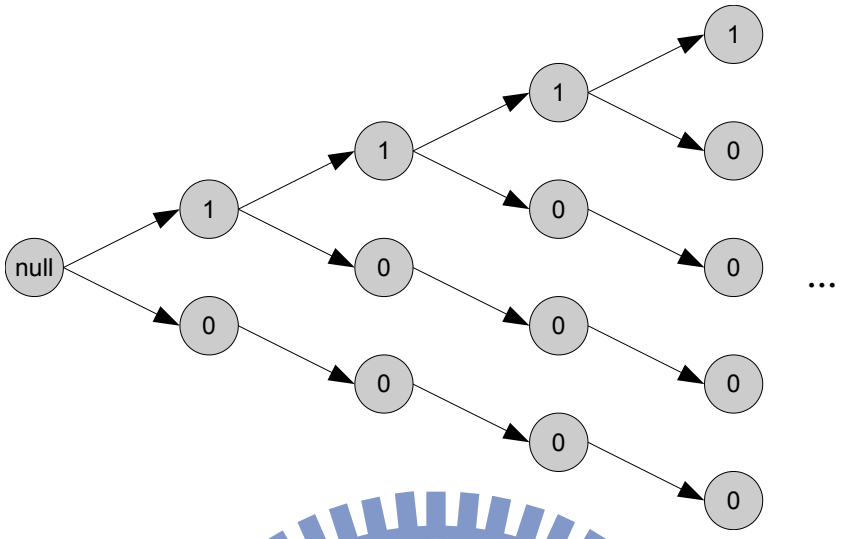

Figure 1.2: UDOOC code tree for UW = 01.

set of patterns, for which the definition will be later given in Definition 4 for completeness. Different from the algorithmic approach in [24], Guibas and Odlyzko established upper and lower bounds for the number of overlaps when the length of the concerned pattern goes to infinity [16].

The rest of the thesis is organized as follows. In Chapter 2, construction of general UDOOCs is introduced. In Chapter 3, combinatorial properties of UDOOCs, including the enumeration of the number of codewords, are derived. In Chapter 4, the encoding and decoding algorithms as well as bounds on average codeword length for general UDOOCs are provided and remarked. In Chapter 5, numerical results on UDOOCs are presented. Conclusion is drawn in Chapter 6.

Chapter 2

Construction of UDOOCs

In the previous chapter, we have seen that the code tree of a UDOOC with UW = 00 (or UW = 01) is by far a useful tool for devising its properties. Along this line, we will provide a systematic construction of code tree for general UDOOC in this chapter. Specifically, a digraph [2] with directional edges that meet the type-I and type-II constraints[1] from the UW will be first introduced. By the digraph, the construction of a general UDOOC code tree as well as the determination of the growth rate of UDOOC codewords with respect to the codeword length will follow.

2.1 Digraphs for UDOOCs

Let $\boldsymbol{k} = k_1 \dots k_L$ be the chosen UW of length L. Denote by $G_{\boldsymbol{k}} = (V, E_{\boldsymbol{k}})$ the digraph for the UDOOC with UW = $\boldsymbol{k}$, where $V = \mathbb{F} \times \cdots \times \mathbb{F} = \mathbb{F}^{L-1}$ is the set of all binary length-$(L-1)$ tuples, and $E_{\boldsymbol{k}}$ is the set of directional edges given by

$$E_{\boldsymbol{k}} := \left\{ (\boldsymbol{i}, \boldsymbol{j}) \in V^2 : i_2^{L-1} = j_1^{L-2} \text{ and } i_1 \boldsymbol{j} \neq \boldsymbol{k} \right\}. \tag{2.1}$$

[1]For clarity of its explanation, we introduce the so-called type-I and type-II constraints in Example 1. Listing these constraints for a general UW however may be tedious and less comprehensive. As will be seen from this chapter, these constraints can actually be absorbed into the construction of the digraph (See specifically Eq. (2.1)); hence, explicitly listing of constraints becomes of secondary necessity.

Here, we use the conventional shorthand $i_s^t = i_s i_{s+1} \dots i_t$ to denote a binary string from index s to index t, and the elements in V are interchangeably denoted by either i_1^{L-1} or $\boldsymbol{i} = i_1 \dots i_{L-1}$, depending on whichever is more convenient.

Define the 2^{L-1}-by-2^{L-1} adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$ for the digraph $G_{\boldsymbol{k}}$ by putting its $(i+1, j+1)$th entry as

$$(\mathtt{A}_{\boldsymbol{k}})_{i+1,j+1} = \begin{cases} 1, & \text{if } (\boldsymbol{i}, \boldsymbol{j}) \in E_{\boldsymbol{k}}, \\ 0, & \text{otherwise}, \end{cases} \tag{2.2}$$

where we abuse the notation by using i (resp. j) to be the integer corresponding to binary representation of $\boldsymbol{i} = i_1 \dots i_{L-1}$ (resp. $\boldsymbol{j} = j_1 \dots j_{L-1}$) with the leftmost bit being the most significant bit. As an example, for $\boldsymbol{k} = 010$, we have $V = \{00, 01, 10, 11\}$,

$$E_{010} = \{(00, 00), (00, 01), (01, 11), (10, 00), (10, 01), (11, 10), (11, 11)\},$$

$G_{010} = (V, E_{101})$ in Fig. 2.1, and

$$\mathtt{A}_{010} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

It should be noted that the adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$ will be used in enumerating the number of UDOOC codewords in Chapter 3.

2.2 Code Trees for UDOOCs

Equipped with digraph $G_{\boldsymbol{k}}$, construction of the code tree for the UDOOC with UW $= \boldsymbol{k}$ becomes straightforward.

Recall that a UDOOC codeword of length n is a binary n-tuple $\boldsymbol{b} = b_1 \dots b_n$, satisfying that $\boldsymbol{k}$ is not an internal subword of the concatenated bit-stream $\boldsymbol{kbk}$. As such, the traversal of the digraph for the construction of a UDOOC code tree should start from the vertex

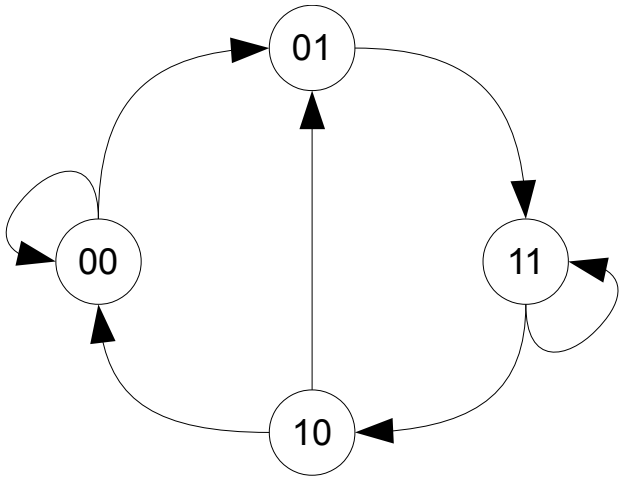

Figure 2.1: Digraph G_{010} for UW $\boldsymbol{k} = 010$.

$k_2^L \in V$, which corresponds to the initial "null"-node in the code tree. Next, a "0"-node at level 1 is generated if both $(k_2^L, j_1^{L-1}) \in E_{\boldsymbol{k}}$ and $j_{L-1} = 0$ are satisfied. By the same rule, the "null"-node is followed by a "1"-node at level 1 if $(k_2^L, j_1^{L-1}) \in E_{\boldsymbol{k}}$ and $j_{L-1} = 1$. We then move the current vertex to j_1^{L-1} and draw a branch from "j_{L-1}"-node at level 1 to a followup "0"-node (resp. "1"-node) at level 2 in the code tree if $(j_1^{L-1}, \ell_1^{L-1}) \in E_{\boldsymbol{k}}$ and $\ell_{L-1} = 0$ (resp. $\ell_{L-1} = 1$). We move the current vertex again to ℓ_1^{L-1} and re-do the above procedure to generate the nodes in the next level. Repeating this process will complete the exploration of the nodes in the entire code tree.

Determination of the gray-shaded nodes that end a codeword can be done as follows. Since $\boldsymbol{k}$ cannot be an internal subword of $\boldsymbol{kbk}$, a node should be gray-shaded if it is immediately followed by a sequence of offspring nodes with their binary marks equal to $k_1 \dots k_{L-1}$. The construction of the UDOOC code tree is accordingly finished.

As an example, we continue from the exemplified UW $\boldsymbol{k} = 010$ with digraph $G_{\boldsymbol{k}}$ in Fig. 2.1 and explore its respective UDOOC code tree in Fig. 2.2 by following the previously mentioned procedure. By starting from the vertex $k_2^3 = 10$ that corresponds to the "null"-node, two

succeeding nodes are generated since both $(10, 00)$ and $(10, 01)$ are in E_{010} (cf. Fig. 2.2). Now from vertex 00 that corresponds to the "0"-node at level 1, we can reach either vertex 00 or vertex 01 in one transition; hence, both "0"-node and "1"-node are the succeeding nodes to the "0"-node at level 1. However, since vertex 01 can only walk to vertex 11 in one transition, the "1"-node at level 1 has only one succeeding node with mark "1." Continuing this process then exhausts all the nodes in the code tree in Fig. 2.2. Next, all nodes that are followed by $k_1k_2 = 01$ in sequence in the code tree are gray-shaded. The code tree for the UDOOC with UW $\boldsymbol{k} = 010$ is then completed.

We end this chapter by giving the type-I and type-II constraints for the exemplified code tree as follows.

- Type-I constraints:

  (C1) "0" can be followed by either "0" or "1".

  (C2) "1" can be followed by "0" only when the node prior to this "1"-node is not a "0"-node.

- Type-II constraints:

  (C3) The first two bits of a UDOOC codeword cannot be "10."

  (C4) The last two bits of a UDOOC codeword cannot be "01."

Note that with these constraints (in particular (C4)), one can also perform the node-shading step by first gray-shading all the nodes in the code tree, and then unshade those that end with "01" (in addition to the "1"-node at level 1 for this specific UW). Nevertheless, it may be tedious to perform the node-unshading for a general UW. For example, when UW $\boldsymbol{k} = 01001$, all nodes that end a codeword b_1^n, satisfying either $b_{n-3}b_{n-2}b_{n-1}b_nk_1 = \boldsymbol{k}$ or $b_{n-2}b_{n-1}b_nk_1k_2 = \boldsymbol{k}$, should be unshaded. This confirms the superiority of constructing the

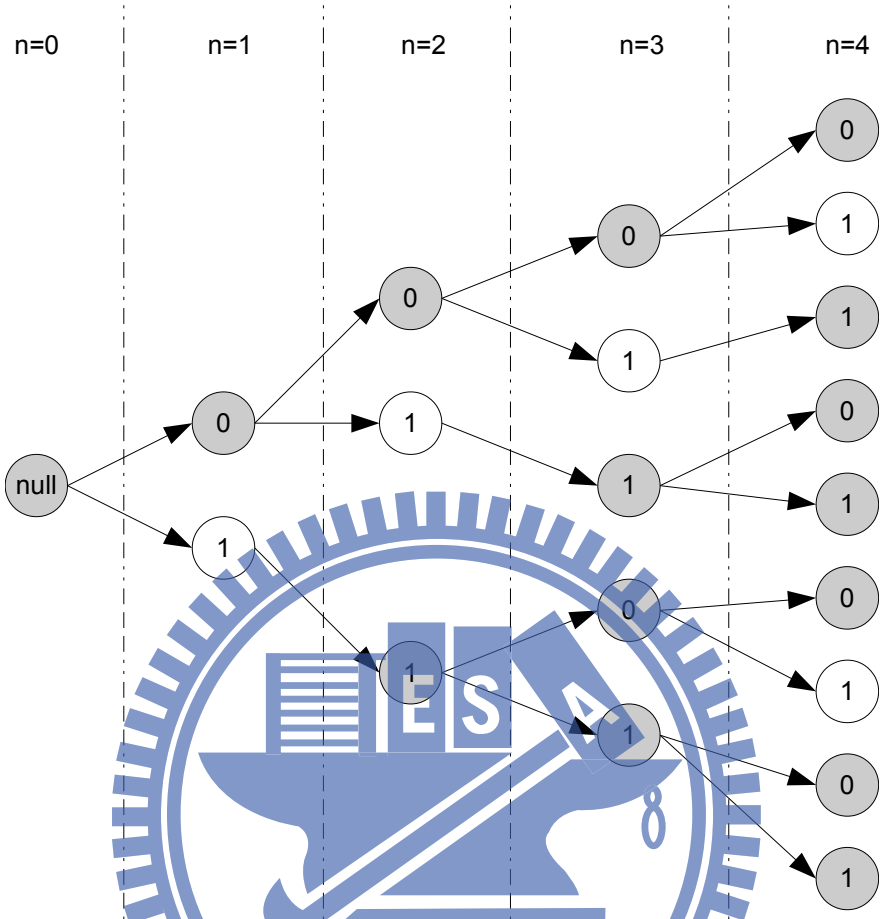

Figure 2.2: Code tree for the UDOOC with UW $\boldsymbol{k} = 010$.

UDOOC code tree in terms of the digraph over analyzing the explicit listing of constraints from the adopted UW that are perhaps convenient only for some special UWs.

Chapter 3

Combinatorial Properties of UDOOCs

3.1 Enumeration of Codewords in $\mathcal{C}_{\boldsymbol{k}}(n)$

In this section, we will see that the conception of digraph $G_{\boldsymbol{k}}$, in particular its respective adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$, can lead to the enumeration of codewords, i.e., $c_{\boldsymbol{k},n} = |\mathcal{C}_{\boldsymbol{k}}(n)|$.

In accordance with the fact that the traversal of the digraph for the construction of a UDOOC code tree should start from vertex $k_2^L \in V$, we define a length-2^{L-1} initial vector as $(\underline{x}_{\boldsymbol{k}})_{j+1} = 1$ if integer j has the binary representation k_2^L, and $(\underline{x}_{\boldsymbol{k}})_{j+1} = 0$, otherwise. It then follows that the $(\ell+1)$th entry of row vector $\underline{x}_{\boldsymbol{k}}^\top \mathtt{A}_{\boldsymbol{k}}^n$ gives the number of length-n walks that end at vertex $\boldsymbol{\ell}$ on digraph $G_{\boldsymbol{k}}$, where "$^\top$" denotes the vector/matrix transpose operation, and $\boldsymbol{\ell} = \ell_1 \ldots \ell_{L-1}$ is the binary representation of integer index ℓ.

However, not every length-n walk produces a codeword. Notably, some nodes on the code tree will be gray-shaded and some will not. Recall that $\boldsymbol{k}$ cannot be an internal subword of $\boldsymbol{kbk}$ if $\boldsymbol{b} = b_1 b_2 \ldots b_n$ is a codeword. This implies that $\boldsymbol{b}$ is a length-n codeword if, and only if, the vertex sequence k_2^L, $k_3^L b_1$, $k_4^L b_1^2$, …, $b_n k_1^{L-2}$, k_1^{L-1} is a valid walk of length $n+L-1$ on digraph $G_{\boldsymbol{k}}$. As a result, the number of length-n codewords equals the number of length-$(n+L-1)$ walks from vertex k_2^L to vertex k_1^{L-1} on digraph $G_{\boldsymbol{k}}$. Following the above discussion, we define a length-2^{L-1} *ending vector* $\underline{y}_{\boldsymbol{k}}$ as $(\underline{y}_{\boldsymbol{k}})_{j+1} = 1$ if integer j has

the binary representation k_1^{L-1}, and $(\underline{y}_{\boldsymbol{k}})_{j+1} = 0$, otherwise. Then, the number of length-n codewords is given by

$$c_{\boldsymbol{k},n} := |\mathcal{C}_{\boldsymbol{k}}(n)| \;=\; \underline{x}_{\boldsymbol{k}}^{\top} \mathtt{A}_{\boldsymbol{k}}^{n+L-1} \underline{y}_{\boldsymbol{k}}. \tag{3.1}$$

3.2 Equivalence among UWs

Two UWs that result in the same minimum average codeword length for every source statistics should be considered *equivalent* because the average codeword length is a key performance measure for lossless compression of a sequence of source letters. We then define the equivalence of two UWs as follows.

Definition 2. *Two UWs $\boldsymbol{k}$ and $\boldsymbol{k}'$ are said to be* equivalent, *denoted by $\boldsymbol{k} \equiv \boldsymbol{k}'$, if the numbers of their length-n codewords in the corresponding UDOOCs are the same for all n, i.e.,*

$$c_{\boldsymbol{k},n} \;=\; c_{\boldsymbol{k}',n} \quad \text{for all } n \geq 0. \tag{3.2}$$

□

By this definition, UDOOCs associated with equivalent UWs have the same number of codewords in every code tree level; hence they achieve the same minimum average codeword length in lossless compression of a sequence of source letters. This equivalence relation allows us to focus only on one UW in every equivalent class. It is however hard to exhaust and identify all equivalent classes of UWs of arbitrary length. Instead, we will introduce a less restrictive notion of *asymptotic equivalence* when the asymptotic compression rate of UDOOCs is concerned, and derive the number of all asymptotically equivalent classes of UWs in the next chapter.

Some properties about the (exact) equivalence of UWs are given below.

Proposition 1 (Equivalence in order reversing)**.** *UW* $\boldsymbol{k}' = k_L \dots k_1$ *is equivalent to UW* $\boldsymbol{k} = k_1 k_2 \dots k_L$.

Proof. It follows simply from that $\boldsymbol{b} = b_1 b_2 \dots b_n \in \mathcal{C}_{\boldsymbol{k}}$ if, and only if, $\boldsymbol{b}' = b_n b_{n-1} \dots b_1 \in \mathcal{C}_{\boldsymbol{k}'}$. □

Proposition 2 (Equivalence in binary complement)**.** *If* $\bar{\boldsymbol{k}}$ *is the bit-wise binary complement of* $\boldsymbol{k}$*, then* $\bar{\boldsymbol{k}} \equiv \boldsymbol{k}$.

Proof. It is a consequence of the fact that the concatenated bit-stream $\boldsymbol{kbk}$ contains $\boldsymbol{k}$ as an internal subword if, and only if, the binary complement $\overline{\boldsymbol{kbk}}$ of $\boldsymbol{kbk}$ contains $\bar{\boldsymbol{k}}$ as an internal subword. □

From Propositions 1 and 2, it can be verified that there are at most four equivalent classes for UWs of length $L = 4$. Representative UWs for these four equivalent classes are 0000, 0001, 0100 and 0101, respectively.

3.3 Growth Rates of UDOOCs

In this section, we investigate the asymptotic growth rate of UDOOCs, of which the definition is first given.

Definition 3. *Given UW* $\boldsymbol{k}$*, the asymptotic growth rate of the resulting UDOOC is defined as*

$$g_{\boldsymbol{k}} := \lim_{n \to \infty} \frac{c_{\boldsymbol{k},n+1}}{c_{\boldsymbol{k},n}}. \tag{3.3}$$

By its definition, the asymptotic growth rate of a UDOOC indicates how fast the number of codewords grows as n increases.

It is obvious that $g_{\boldsymbol{k}} \leq 2$ for all UWs because the upper bound of 2 is the growth rate for unconstrained binary sequences of length n. In addition, the limit in (3.3) must exist since it can be inferred from enumerative combinatorics [27], and also from algebraic graph theory [3], that $g_{\boldsymbol{k}}$ is the largest eigenvalue of adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$. In the next proposition, we show that the largest eigenvalue of adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$ is unique for all UWs but $\boldsymbol{k} = 01$.

Proposition 3 (Uniqueness of the largest eigenvalue of $\mathtt{A}_{\boldsymbol{k}}$)**.** *For any UW* $\boldsymbol{k}$ *of length* $L \geq 2$ *except* $\boldsymbol{k} = 01$*, the largest eigenvalue of adjacency matrix* $\mathtt{A}_{\boldsymbol{k}}$ *is unique.*

Proof. By Perron-Frobenius theorem [14][18], the largest engenvalue of adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$ is unique with algebraic multiplicity equal to 1 if $G_{\boldsymbol{k}}$ is a strongly connected diagrph. Thus, we only need to show that $G_{\boldsymbol{k}}$ is a strongly connected digraph except for $\boldsymbol{k} = 01$.

We then argue that $G_{\boldsymbol{k}}$ is a strongly connected diagrph when $L \geq 3$ as follows. According to the definition of $E_{\boldsymbol{k}}$ in (2.1), the only situation that a vertex may not be strongly connected to other vertex is when $j = k_2 k_3 \cdots k_L$. This however cannot happen when $L \geq 3$ because vertex $\overline{k}_1 k_2 \cdots k_{L-1}$ will connect strongly to $k_2 k_3 \cdots k_L$. The proof is completed by verifying the two cases for $L = 2$, i.e., G_{00} is strongly connected but G_{01} is not. □

The digraph for $\boldsymbol{k} = 01$ is plotted in Fig. 3.1. It clearly indicates that there is no directed path from vertex 0 to vertex 1. As expected, the algebraic multiplicity of the largest eigenvalue 1 of $\mathtt{A}_{01}$ is 2.

By the standard technique of using an indeterminate z in enumerative combinatorics, we

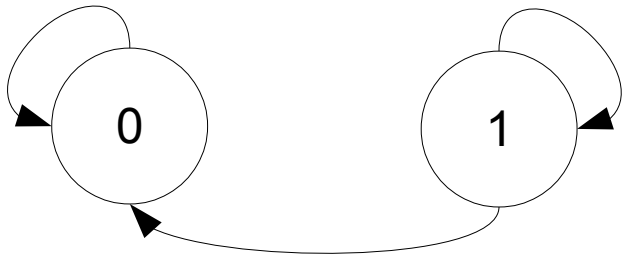

Figure 3.1: Digraph $G_{\boldsymbol{k}}$ for UW $\boldsymbol{k} = 01$.

derive

$$\begin{aligned}\sum_{n=0}^{\infty} c_{\boldsymbol{k},n} z^n &= \sum_{n=0}^{\infty} \underline{x}_{\boldsymbol{k}}^{\top} \mathtt{A}_{\boldsymbol{k}}^{n+L-1} \underline{y}_{\boldsymbol{k}} z^n \\ &= \underline{x}_{\boldsymbol{k}}^{\top} \left(\sum_{n=0}^{\infty} \mathtt{A}_{\boldsymbol{k}}^{n} z^n \right) \mathtt{A}_{\boldsymbol{k}}^{L-1} \underline{y}_{\boldsymbol{k}} \\ &= \underline{x}_{\boldsymbol{k}}^{\top} \left(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z \right)^{-1} \mathtt{A}_{\boldsymbol{k}}^{L-1} \underline{y}_{\boldsymbol{k}} \\ &= \frac{\underline{x}_{\boldsymbol{k}}^{\top} \mathrm{adj} \left(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z \right) \mathtt{A}_{\boldsymbol{k}}^{L-1} \underline{y}_{\boldsymbol{k}}}{\det \left(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z \right)}, \end{aligned} \tag{3.4}$$

where the first equality follows from (3.1) and $\mathtt{I}$ denotes the identity matrix of proper size. Equation (3.4) then implies that $\det\left(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z\right)$ can give a linear recursion of $c_{\boldsymbol{k},n}$ in the form of a linear constant coefficient difference equation (LCCDE).

Now let $\lambda_1, \ldots, \lambda_m$ be the distinct nonzero eigenvalues of adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$ with algebraic multiplicities $e_1, \ldots, e_m$, respectively, where we assume with no loss of generality that $|\lambda_1| \geq \cdots \geq |\lambda_m|$. In terms of the technique of standard partial fraction for polynomial ring, we can rewrite (3.4) as

$$\frac{\underline{x}_{\boldsymbol{k}}^{\top} \mathrm{adj} \left(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z \right) \mathtt{A}_{\boldsymbol{k}}^{L-1} \underline{y}_{\boldsymbol{k}}}{\det \left(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z \right)} = \sum_{i=1}^{m} \frac{p_i(z)}{(1 - \lambda_i z)^{e_m}} \tag{3.5}$$

for some polynomials $p_i(z)$. The next step is expectantly to rewrite the righ-hand-side (RHS) of (3.5) as a power series of indeterminate z in order to recover the actual values of $c_{\boldsymbol{k},n}$ for

all n. As an example, this can be done by the typical approach in enumerative combinatorics through

$$\frac{1}{(1-\lambda_i z)^{e_m}} = \sum_{n=0}^{\infty} \binom{n+e_m-1}{n} \lambda_i^n z^n,$$

which holds for all $|z| < \min_{1\le i\le m} \frac{1}{|\lambda_i|}$.

Although the asymptotic growth rate $g_{\boldsymbol{k}}$ equals exactly the largest eigenvalue of adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$, it is in general difficult to find a closed-form expression for this value without a proper reshaping of adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$. To this end, we consider the following set for $n \ge L$,

$$\mathcal{S}_{\boldsymbol{k}}(n) := \{\boldsymbol{b} \in \mathbb{F}^n : \boldsymbol{k} \text{ is not a subword of } \boldsymbol{b}\}, \tag{3.6}$$

which, in a way, defines the set of distinct length-n walks on digraph $G_{\boldsymbol{k}}$. Denoting $s_{\boldsymbol{k},n} := |\mathcal{S}_{\boldsymbol{k}}(n)|$ and by an argument similar to (3.4), one can easily show that

$$\sum_{n=0}^{\infty} s_{\boldsymbol{k},n} z^n = \sum_{n=0}^{L-1} 2^n z^n + z^L \frac{\underline{\mathbf{1}}^\top \operatorname{adj}(\mathtt{I}-\mathtt{A}_{\boldsymbol{k}} z)\underline{\mathbf{1}}}{\det(\mathtt{I}-\mathtt{A}_{\boldsymbol{k}} z)}, \tag{3.7}$$

where $\underline{\mathbf{1}}$ is the all-one vector of appropriate length. Equation (3.7) then implies that the enumeration of $s_{\boldsymbol{k},n}$ also depends upon the polynomial $\det(\mathtt{I}-\mathtt{A}_{\boldsymbol{k}} z)$ as $c_{\boldsymbol{k},n}$ does. Based on this observation, we can infer and prove that $s_{\boldsymbol{k},n}$ has the same same asymptotic growth rate as $c_{\boldsymbol{k},n}$. We summarize this important inference in the proposition below, of which the proof will be relegated to the next section.

Proposition 4. *For any UW $\boldsymbol{k}$, sequences $\{c_{\boldsymbol{k},n}\}_{n=0}^{\infty}$ and $\{s_{\boldsymbol{k},n}\}_{n=0}^{\infty}$ have the same asymptotic growth rate, i.e.,*

$$g_{\boldsymbol{k}} = \mathfrak{g}_{\boldsymbol{k}},$$

where

$$\mathfrak{g}_{\boldsymbol{k}} := \lim_{n\to\infty} \frac{s_{\boldsymbol{k},n+1}}{s_{\boldsymbol{k},n}}.$$

Notably, in order to distinguish the asymptotic growth rate of $s_{\boldsymbol{k},n}$ from that of $c_{\boldsymbol{k},n}$, a different font $\mathfrak{g}_{\boldsymbol{k}}$ is used to denote the asymptotic growth rate of $s_{\boldsymbol{k},n}$.

3.4 Enumeration of $s_{\boldsymbol{k},n}$

Enumerating $s_{\boldsymbol{k},n}$ turns out to be easier than enumerating $c_{\boldsymbol{k},n}$ due to that there is lesser number of constraints on the sequences in $\mathcal{S}_{\boldsymbol{k},n}$. It can be done by an approach similar to the *Goulden-Jackson clustering method* [22]. Before the presentation of the main theorems, we define the *overlap function* and *overlap vector* of a binary stream $\boldsymbol{k}$ as follows.

Definition 4. *For a given $\boldsymbol{k}$ of length L, its overlap function is defined as*

$$r_{\boldsymbol{k}}(i) := \begin{cases} 1 & \text{if } k_{i+1}^{L} = k_1^{L-i} \text{ and } 0 \le i \le L-1 \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

Furthermore, we define its length-L overlap vector as $(\underline{r}_{\boldsymbol{k}})_j = r_{\boldsymbol{k}}(j-1)$ for $j = 1 \dots L$.

Theorem 1. *For a length-L UW $\boldsymbol{k}$ with overlap function $r_{\boldsymbol{k}}(i)$,*

$$\sum_{n\ge 0} s_{\boldsymbol{k},n} z^n = \frac{1 + \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) z^i}{(1-2z)(1 + \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) z^i) + z^L}. \tag{3.9}$$

Proof. The result follows from the Goulden-Jackson clustering method [22]. For completeness, a simplified proof to this theorem is provided in Appendix A. □

The next example illustrates the usage of the above theorem to the target result of Proposition 4.

Example 2. *Consider the case of $\boldsymbol{k} = 000$; then by following (3.8), the corresponding overlap function is*

$$r_{000}(i) = \begin{cases} 1, & i = 0, 1, 2, \\ 0, & \text{otherwise.} \end{cases}$$

Substituting the above into (3.9), we obtain

$$\sum_{n\ge 0} s_{000,n} z^n = \frac{1 + z + z^2}{1 - z - z^2 - z^3}.$$

By regarding the above as an LCCDE, we conclude that the sequence $s_{000,n}$ satisfies the following recursion for all $n \geq 0$:

$$s_{000,n} = s_{000,n-1} + s_{000,n-2} + s_{000,n-3} + \delta_n + \delta_{n-1} + \delta_{n-2},$$

where

$$\delta_n = \begin{cases} 1, & n = 0 \\ 0, & \textit{otherwise} \end{cases}$$

is the Kronecker delta function. □

Equipped with Theorem 1, we are now ready to prove Proposition 4.

Proof of Proposition 4. Let $h_{\boldsymbol{k}}(z)$ denote the denominator of (3.9), i.e.,

$$h_{\boldsymbol{k}}(z) = (1-2z)\left(1 + \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) z^i\right) + z^L. \tag{3.10}$$

As the RHS of (3.9) is an irreducible rational function with denominator $h_{\boldsymbol{k}}(z)$, it follows that

$$\mathfrak{g}_{\boldsymbol{k}} = \max\{u^{-1} \ : \ h_{\boldsymbol{k}}(u) = 0\}.$$

Since the RHSs of (3.4) and (3.9) share the same denominator, for which a proof will be given later in Corollary 1, we get

$$g_{\boldsymbol{k}} \leq \max\{u^{-1} \ : \ h_{\boldsymbol{k}}(u) = 0\},$$

where it becomes an inequality because the rational function in (3.4) could be reducible. This completes the converse proof: $g_{\boldsymbol{k}} \leq \mathfrak{g}_{\boldsymbol{k}}$.

To prove $g_{\boldsymbol{k}} \geq \mathfrak{g}_{\boldsymbol{k}}$ (which then implies $g_{\boldsymbol{k}} = \mathfrak{g}_{\boldsymbol{k}}$), it suffices to show that $c_{\boldsymbol{k},n+2} \geq s_{\boldsymbol{k},n}$ for $n \geq L$. This can be done by substantiating that for any $\boldsymbol{b} \in \mathcal{S}_{\boldsymbol{k}}(n)$, there exist a prefix bit p and a suffix bit q, where $p, q \in \mathbb{F}$, such that $p\boldsymbol{b}q \in \mathcal{C}_{\boldsymbol{k}}(n+2)$.

Using the prove-by-contradiction argument, we first claim that $\boldsymbol{k}$ is an internal subword of both $\boldsymbol{k}p\boldsymbol{b}$ and $\boldsymbol{k}\bar{p}\boldsymbol{b}$, where $\bar{p} = 1 - p$. This claim, together with $\boldsymbol{b} \in \mathcal{S}_{\boldsymbol{k}}(n)$, implies the existence of indices $1 < i < L + 2$ and $1 < j < L + 2$ such that

$$\underbrace{k_i \cdots k_L p b_1 \cdots b_{i-2}}_{=\boldsymbol{a}} = \underbrace{k_j \cdots k_L \bar{p} b_1 \cdots b_{j-2}}_{=\tilde{\boldsymbol{a}}} = \boldsymbol{k}, \tag{3.11}$$

where we abuse the notations to let

$$\boldsymbol{a} = \begin{cases} k_i \cdots k_L p, & \text{if } i = 2 \\ k_i \cdots k_L p b_1 \cdots b_{i-2}, & \text{if } 2 < i < L + 1 \\ p b_1 \cdots b_{i-2}, & \text{if } i = L + 1 \end{cases} \tag{3.12}$$

and similar notational abuse is applied to $\tilde{\boldsymbol{a}}$ and j. Assume without loss of generality that $i < j$. Then, the sums of the last $(j - 1)$ bits of $\boldsymbol{a}$ and $\tilde{\boldsymbol{a}}$ must equal, i.e.,

$$k_{L-(j-i-1)} + \cdots + k_L + p + b_1 + \cdots + b_{i-2} = \bar{p} + b_1 + \cdots + b_{j-2}.$$

Canceling out common terms at both sides gives

$$k_{L-(j-i-1)} + \cdots + k_L + p = \bar{p} + b_{i-1} + \cdots + b_{j-2}. \tag{3.13}$$

Note again that $\tilde{\boldsymbol{a}} = \boldsymbol{k}$; hence, substituting $b_{(j-2)-\ell}$ by $k_{L-\ell}$ for $\ell = 0, 1, \cdots, j - i - 1$ in (3.13) gives $p = \bar{p}$, which contradicts the assumption that $\bar{p} = 1 - p$.

For the suffix bit q, we again assume to the contrary that there exist indices i and j, satisfying $n + 1 - L < i < j < n + 2$, such that

$$\underbrace{b_i \cdots b_n q k_1 \cdots k_{L-n+i-2}}_{=\boldsymbol{d}} = \underbrace{b_j \cdots b_n \bar{q} k_1 \cdots k_{L-n+j-2}}_{=\tilde{\boldsymbol{d}}} = \boldsymbol{k} \tag{3.14}$$

After canceling out common terms in the respective sums of the first $(n + 2 - i)$ bits of $\boldsymbol{d}$ and $\tilde{\boldsymbol{d}}$, we obtain

$$b_i + \cdots + b_{j-1} + q = \bar{q} + k_1 + \cdots + k_{j-i}.$$

Since $\boldsymbol{d} = \boldsymbol{k}$, the above then implies $q = \bar{q}$, which again leads to a contradiction. □

As having been mentioned, the LCCDE of $c_{\boldsymbol{k},n}$ can be used to provide a recursion formula for $c_{\boldsymbol{k},n}$. A claim that can lead to this desired LCCDE of $c_{\boldsymbol{k},n}$ is that $h_{\boldsymbol{k}}(z) = \det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)$. We then summarize both in the next corollary.

Corollary 1. *$h_{\boldsymbol{k}}(z)$ defined in* (3.10) *equals the denominator in* (3.4)*, i.e.,*

$$h_{\boldsymbol{k}}(z) = \det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z). \tag{3.15}$$

In addition, (3.4) *implies for $n \geq L$,*

$$c_{\boldsymbol{k},n} = \left[\sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i)\left(2c_{\boldsymbol{k},n-i-1} - c_{\boldsymbol{k},n-i}\right)\right] + 2c_{\boldsymbol{k},n-1} - c_{\boldsymbol{k},n-L}. \tag{3.16}$$

Proof. Combining (3.7) and (3.9) gives

$$\frac{1 + \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) z^i}{(1-2z)(1 + \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) z^i) + z^L} = \frac{f(z)}{\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)},$$

for some polynomial $f(z)$. Notice that the left-hand-side (LHS) is an irreducible rational function in z. Furthermore, Proposition 10 in Appendix B shows $\deg \det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z) = L$. It then implies that

$$\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z) \;=\; h_{\boldsymbol{k}}(z)$$

and

$$f(z) \;=\; 1 + \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) z^i,$$

and (3.15) is thus established. To prove (3.16), note that the characteristic polynomial for $\mathtt{A}_{\boldsymbol{k}}$ is given by

$$\begin{aligned} \chi_{\mathtt{A}_{\boldsymbol{k}}}(z) &= \det(z\mathtt{I} - \mathtt{A}_{\boldsymbol{k}}) \\ &= z^{2^{L-1}} h_{\boldsymbol{k}}(z^{-1}) \\ &= z^{2^{L-1}-L}(z^L h_{\boldsymbol{k}}(z^{-1})) \end{aligned}$$

where $z^L h_{\boldsymbol{k}}(z^{-1})$ is a polynomial with degree

$$\deg h_{\boldsymbol{k}}(z) = \deg \det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z) = L.$$

Denote

$$m \;=\; \min\{p > 0 \;:\; \text{Nullity}(\mathtt{A}_{\boldsymbol{k}}^p) = 2^{L-1} - L\}. \tag{3.17}$$

By Cayley-Hamilton Theorem [17], the following polynomial

$$\begin{aligned}\mu_{\boldsymbol{k}}(z) &= z^m (z^L h_{\boldsymbol{k}}(z^{-1})) \\ &= z^m \left(z^L - 2z^{L-1} + \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) z^{L-i-1}(z-2) + 1 \right)\end{aligned} \tag{3.18}$$

is an annihilating polynomial for $\mathtt{A}_{\boldsymbol{k}}$. We shall remark that $\mu_{\boldsymbol{k}}(z)$ needs not be the minimal polynomial for $\mathtt{A}_{\boldsymbol{k}}$. Plugging (3.18) into (3.1) yields that for $n-1 \geq \max\{m, L-1\}$, we have

$$\begin{aligned} c_{\boldsymbol{k},n} &= \underline{x}_{\boldsymbol{k}}^\top \mathtt{A}_{\boldsymbol{k}}^{n+L-1} \underline{y}_{\boldsymbol{k}} \\ &= \underline{x}_{\boldsymbol{k}}^\top \mathtt{A}_{\boldsymbol{k}}^{n-1-m} \mathtt{A}_{\boldsymbol{k}}^{m+L} \underline{y}_{\boldsymbol{k}} \end{aligned} \tag{3.19}$$

$$\begin{aligned} &= \underline{x}_{\boldsymbol{k}}^\top \mathtt{A}_{\boldsymbol{k}}^{n-1} \left[2\mathtt{A}_{\boldsymbol{k}}^{L-1} + \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i)(2\mathtt{A}_{\boldsymbol{k}}^{L-i-1} - \mathtt{A}_{\boldsymbol{k}}^{L-i}) - \mathtt{I} \right] \underline{y}_{\boldsymbol{k}} \\ &= \left[\sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) \left(2c_{\boldsymbol{k},n-i-1} - c_{\boldsymbol{k},n-i}\right) \right] + 2c_{\boldsymbol{k},n-1} - c_{\boldsymbol{k},n-L} \end{aligned} \tag{3.20}$$

where the condition of $n-1 \geq \max\{m, L-1\}$ follows from (i) $n-1-m \geq 0$ such that (3.19) holds, and (ii) $n-1 \geq L-1$ such that the last term of the RHS of (3.20) represents $c_{\boldsymbol{k},n-L}$.

Finally, since $\text{rank}(\mathtt{A}_{\boldsymbol{k}}^p) \leq L$ for $p = L-1$ (see Proposition 10), we have $m \leq L-1$, which immediately gives $\max\{m, L-1\} = L-1$, and the proof is completed. □

From Proposition 4, we learn that $c_{\boldsymbol{k},n}$ and $s_{\boldsymbol{k},n}$ have the same asymptotic growth rate, and both of their enumerations depend on $\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)$. Below we will use $s_{\boldsymbol{k},n}$ to determine

the asymptotic growth rates corresponding to two specific UWs, $\boldsymbol{a} = 0\ldots00$ and $\boldsymbol{b} = 0\ldots01$. We then proceed to show that $\boldsymbol{a}$ has the largest growth rate among all UWs of the same length, while the smallest growth rate is resulted when UW $= \boldsymbol{b}$.

Theorem 2. *Among all UWs of the same length, the all-zero UW has the largest growth rate, while UW* $0\ldots01$ *achieves the smallest.*

Proof. For notational convenience, we set $\boldsymbol{a} = 0\ldots0$ and $\boldsymbol{b} = 0\ldots01$. For UW $= \boldsymbol{a}$, it can be verified from (3.8) and (3.9) that

$$h_{\boldsymbol{a}}(z) = 1 - \sum_{i=1}^{L} z^i, \tag{3.21}$$

and hence the sequence of $\{s_{\boldsymbol{a},n}\}_{n=-1}^{\infty}$ satisfies the following recursion:

$$s_{\boldsymbol{a},n} = \sum_{i=1}^{L} s_{\boldsymbol{a},n-i} \quad \text{for } n \geq L.$$

Similarly, we have $h_{\boldsymbol{b}}(z) = 1 - 2z + z^L$, and therefore,

$$s_{\boldsymbol{b},n} = 2s_{\boldsymbol{b},n-1} - s_{\boldsymbol{b},n-L} \quad \text{for } n \geq L.$$

For general UW $\boldsymbol{k}$ of length L, (3.9) gives the following recursion:

$$s_{\boldsymbol{k},n} = \sum_{i=1}^{L-1} (2s_{\boldsymbol{k},n-i-1} - s_{\boldsymbol{k},n-i})\, r_{\boldsymbol{k}}(i) + 2s_{\boldsymbol{k},n-1} - s_{\boldsymbol{k},n-L}. \tag{3.22}$$

Note that $r_{\boldsymbol{k}}(i) \in \{0, 1\}$ by definition, and $2s_{\boldsymbol{k},m-1} \geq s_{\boldsymbol{k},m}$ for all m. From (3.22), the following bounds hold for any UW $\boldsymbol{k}$ with $n \geq L$:

$$2s_{\boldsymbol{k},n-1} - s_{\boldsymbol{k},n-L} \;\leq\; s_{\boldsymbol{k},n} \;\leq\; \sum_{i=1}^{L} s_{\boldsymbol{k},n-i}, \tag{3.23}$$

where the lower bound is obtained by replacing all $r_{\boldsymbol{k}}(i)$ by 0, and the upper bound is obtained by replacing all $r_{\boldsymbol{k}}(i)$ by 1. In particular, $s_{\boldsymbol{k},n}$ equals the upper bound in (3.23)

when $\boldsymbol{k} = \boldsymbol{a} = 00\cdots0$, and the lower bound is achieved when $\boldsymbol{k}$ is $\boldsymbol{b} = 00\cdots01$. By dividing all terms in (3.23) by $s_{\boldsymbol{k},n-1}$ and taking $n \to \infty$, we obtain

$$2 - g_{\boldsymbol{k}}^{-(L-1)} \;\le\; g_{\boldsymbol{k}} \;\le 1 + g_{\boldsymbol{k}}^{-1} + \cdots + g_{\boldsymbol{k}}^{-L+1}. \tag{3.24}$$

To prove our claim that $g_{\boldsymbol{a}}$ is the largest and $g_{\boldsymbol{b}}$ is the smallest among all $g_{\boldsymbol{k}}$, we first assume to the contrary that there exists $\hat{\boldsymbol{k}}$ with $g_{\hat{\boldsymbol{k}}} > g_{\boldsymbol{a}}$. Substituting this into (3.24) leads to the following contradiction

$$g_{\hat{\boldsymbol{k}}} \overset{(i)}{<} \sum_{i=1}^{L} g_{\boldsymbol{a}}^{-i+1} \overset{(ii)}{=} g_{\boldsymbol{a}},$$

where (i) holds because $g_{\hat{\boldsymbol{k}}}^{-1} < g_{\boldsymbol{a}}^{-1}$ by assumption and (ii) is valid because $g_{\boldsymbol{a}}^{-1}$ is a zero of $h_{\boldsymbol{a}}(z)$ given in (3.21).

To show $g_{\boldsymbol{b}}$ achieves the minimum, again assume to the contrary that there exists $\hat{\boldsymbol{k}}$ such that $g_{\hat{\boldsymbol{k}}} < g_{\boldsymbol{b}}$. Note from (3.24) that

$$0 \le g_{\hat{\boldsymbol{k}}} - 2 + g_{\hat{\boldsymbol{k}}}^{-(L-1)} = (1 - g_{\hat{\boldsymbol{k}}}) \left(g_{\hat{\boldsymbol{k}}}^{-L+1} + \cdots + g_{\hat{\boldsymbol{k}}}^{-1} - 1 \right). \tag{3.25}$$

Although $g_{\boldsymbol{k}} \ge 1$ in general, we claim in this case $g_{\hat{\boldsymbol{k}}} > 1$. For otherwise, that $h_{\hat{\boldsymbol{k}}}(z = g_{\hat{\boldsymbol{k}}}^{-1} = 1) = 0$ according to (3.10) implies that $r_{\hat{\boldsymbol{k}}}(i) = 0$ for all i. Hence, $h_{\hat{\boldsymbol{k}}}(z) = h_{\boldsymbol{b}}(z)$ and $g_{\hat{\boldsymbol{k}}} = g_{\boldsymbol{b}}$, a contradiction. Now with $1 < g_{\hat{\boldsymbol{k}}} < g_{\boldsymbol{b}}$, the following series of inequalities lead to the desired contradiction:

$$g_{\boldsymbol{b}} \overset{(i)}{=} g_{\boldsymbol{b}}^{-L+2} + \cdots + 1 \overset{(ii)}{<} g_{\hat{\boldsymbol{k}}}^{-L+2} + \cdots + 1 \overset{(iii)}{\ge} g_{\hat{\boldsymbol{k}}},$$

where (i) follows from $h_{\boldsymbol{b}}(z = g_{\boldsymbol{b}}^{-1}) = 0$ and $g_{\boldsymbol{b}} > 1$, (ii) holds because $g_{\boldsymbol{b}}^{-1} < g_{\hat{\boldsymbol{k}}}^{-1}$, and (iii) is due to (3.25) and $g_{\hat{\boldsymbol{k}}} > 1$. □

Using similar technique in the proof of Theorem 2, we can further devise a general upper bound and a general lower bound for $g_{\boldsymbol{k}}$ that hold for any $\boldsymbol{k}$.

Theorem 3. *For any UW* $\boldsymbol{k}$ *of length* $L \geq 2$*, the asymptotic growth rate* $g_{\boldsymbol{k}}$ *satisfies*

$$2 - 2^{-(L-2)} \leq g_{\boldsymbol{k}} \leq 2 - 2^{-L}. \tag{3.26}$$

Proof. It is straightforward to see $s_{\boldsymbol{k},n-1} \leq s_{\boldsymbol{k},n} \leq 2s_{\boldsymbol{k},n-1}$ and hence $1 \leq g_{\boldsymbol{k}} \leq 2$.

To prove the upper bound, we assume without loss of generality that $g_{\boldsymbol{k}} > 1$ since the upper bound trivially hold when $g_{\boldsymbol{k}} = 1$. We then derive

$$g_{\boldsymbol{k}} - 1 \;=\; g_{\boldsymbol{k}}(1 - g_{\boldsymbol{k}}^{-1}) \overset{(i)}{\leq} 1 - g_{\boldsymbol{k}}^{-L} \overset{(ii)}{\leq} 1 - 2^{-L},$$

where (i) follows from multiplying both sides of the second inequality in (3.24) by $(1 - g_{\boldsymbol{k}}^{-1})$ with the fact $g_{\boldsymbol{k}} > 1$, and (ii) holds since $g_{\boldsymbol{k}} \leq 2$.

To establish the lower bound, we use the following series of inequalities:

$$\begin{aligned} g_{\boldsymbol{k}}(1 - g_{\boldsymbol{k}}^{-1}) &= g_{\boldsymbol{k}} - 1 \\ &\overset{(i)}{\geq} 1 - g_{\boldsymbol{k}}^{-(L-1)} \\ &= (1 - g_{\boldsymbol{k}}^{-1})\left(1 + g_{\boldsymbol{k}}^{-1} + g_{\boldsymbol{k}}^{-2} + \cdots + g_{\boldsymbol{k}}^{-(L-2)}\right) \\ &\overset{(ii)}{\geq} (1 - g_{\boldsymbol{k}}^{-1})\left(1 + 2^{-1} + 2^{-2} + \cdots + 2^{-(L-2)}\right) \\ &= (1 - g_{\boldsymbol{k}}^{-1})\left(2 - 2^{-(L-2)}\right), \end{aligned} \tag{3.27}$$

where (i) is from the first inequality in (3.24), and (ii) holds because $g_{\boldsymbol{k}} \leq 2$. Equipped with (3.27), we next distinguish two cases to complete the proof.

1. When $L = 2$, the lower bound is trivially valid, and is actually achieved by taking UW $= 01$ as g_{01} equals the multiplicative inverse of the smallest zero of polynomial $h_{01}(z) = 1 - 2z + z^2 = (1 - z)^2$.

2. For $L > 2$, it suffices to show $g_{\boldsymbol{k}} > 1$. Assume to the contrary that there exists $\boldsymbol{k}$ of length $L > 2$ such that $g_{\boldsymbol{k}} = 1$. By $h_{\boldsymbol{k}}(z = g_{\boldsymbol{k}}^{-1} = 1) = 0$ and (3.10), we have $r_{\boldsymbol{k}}(i) = 0$ for all i and hence $h_{\boldsymbol{k}}(z) = 1 - 2z + z^L$. Since $g_{\boldsymbol{k}}$ is the multiplicative inverse of the smallest zero of $h_{\boldsymbol{k}}(z)$, the absolute values of all the remaining zeros of $h_{\boldsymbol{k}}(z)$,

say $\lambda_1, \ldots, \lambda_{L-1}$, must be strictly larger than 1. It then follows that in the splitting of $h_{\boldsymbol{k}}(z)$, i.e.,

$$h_{\boldsymbol{k}}(z) \;=\; (z-1)\prod_{i=1}^{L-1}(z-\lambda_i),$$

the constant term $\prod_{i=1}^{L-1}|\lambda_i|$ must also be larger than 1, contradicting to the fact that the constant term in polynomial $h_{\boldsymbol{k}}(z) = 1 - 2z + z^L$ is 1.

□

Theorem 3 provides concrete explicit expressions for both upper and lower bounds on $g_{\boldsymbol{k}}$. Although the bounds are asymptotically tight and well approximate the true $g_{\boldsymbol{k}}$ for moderately large L, they are not sharp in general. We can actually refine them using Theorem 2 and obtain that $g_{\boldsymbol{b}} \le g_{\boldsymbol{k}} \le g_{\boldsymbol{a}}$, where from the proof of Theorem 2, we have

$$g_{\boldsymbol{a}} = \max\{t : h_{\boldsymbol{a}}(z = t^{-1}) = 0\}$$

and

$$g_{\boldsymbol{b}} = \max\{t : h_{\boldsymbol{b}}(z = t^{-1}) = 0\}.$$

The determination of $g_{\boldsymbol{a}}$ and $g_{\boldsymbol{b}}$ can be done via finding the largest zeros of $\bar{h}_{\boldsymbol{a}}(z) = z^L h_{\boldsymbol{a}}(z^{-1})$ and $\bar{h}_{\boldsymbol{b}}(z) = z^L h_{\boldsymbol{b}}(z^{-1})$, respectively. By noting that

$$\begin{aligned}(z-1)\bar{h}_{\boldsymbol{a}}(z) &= (z-1)(z^L - z^{L-1} - \cdots - 1)\\ &= z^{L+1} - 2z^L + 1\end{aligned}$$

and

$$\bar{h}_{\boldsymbol{b}}(z) = z^L - 2z^{L-1} + 1,$$

we conclude the following corollary.

Corollary 2. *Let* $\boldsymbol{a} = 0\ldots0$ *and* $\boldsymbol{b} = 0\ldots01$ *be binary streams of length* L*. Then for any* $\boldsymbol{k}$ *of the same length to* $\boldsymbol{a}$ *and* $\boldsymbol{b}$*,*

$$g_{\boldsymbol{b}} \leq g_{\boldsymbol{k}} \leq g_{\boldsymbol{a}}.$$

In addition, $g_{\boldsymbol{a}} = \alpha_{L+1}$ *and* $g_{\boldsymbol{b}} = \alpha_L$*, where* α_L *is the largest zero of the polynomial* $z^L - 2z^{L-1} + 1$*. In particular, we have* $\alpha_L \approx 2 - 2^{-L+1}$ *for large* L*.* □

Based on Theorem 3, the following corollary is immediate by taking L to infinity.

Corollary 3. *As* $L \to \infty$*, the asymptotic growth rate approaches* 2 *for all UW* $\boldsymbol{k}$*, i.e.,*

$$\lim_{L\to\infty} g_{\boldsymbol{k}} = 2.$$

In Table 3.1, we illustrate the asymptotic growth rates of UDOOCS for UWs $\boldsymbol{a}$ and $\boldsymbol{b}$ with length up to 8. Also shown are the bounds in Theorem 3. It is seen that for moderately large L, all UDOOCs have roughly the same asymptotic growth rate, and hence are about the same good in terms of compressing sources of large size. Furthermore, having $g_{\boldsymbol{k}} \to 2$ as $L \to \infty$ means that for very large L, UDOOCs can have asymptotic growth rates comparable to the unconstrained OOC, whose asymptotic growth rate equals 2.

Table 3.1: The asymptotic growth rates for UWs $\boldsymbol{a}$ and $\boldsymbol{b}$ and the bounds in Theorem 3 with various L

L	2	3	4	5	6	7	8
$2-2^{-L}$	1.75	1.875	1.938	1.969	1.984	1.992	1.996
$g_{\boldsymbol{a}}$	1.618	1.839	1.928	1.966	1.984	1.992	1.996
$g_{\boldsymbol{b}}$	1	1.618	1.839	1.928	1.966	1.984	1.992
$2-2^{-(L-2)}$	1	1.5	1.75	1.875	1.938	1.969	1.984

3.5 Asymptotic Equivalence

After presenting the results on asymptotic growth rates, we proceed to define the asymptotic equivalence for UWs and show that the number of asymptotic UW classes is equal to the number of different overlap vectors in Definition 4.

Definition 5. *Two UWs* $\boldsymbol{k}$ *and* $\boldsymbol{k}'$ *are said to be* asymptotically equivalent, *denoted by* $\boldsymbol{k} \overset{a.e.}{\equiv} \boldsymbol{k}'$, *if they have the same growth rate, i.e.,* $g_{\boldsymbol{k}} = g_{\boldsymbol{k}'}$. □

Following the definition, we have the next proposition.

Proposition 5. *Fix the length* L *of UWs, and denote by* κ_L *the number of all possible overlap vectors of length* L. *Then, the number of asymptotically equivalent UW classes is upper-bounded by* κ_L.

Proof. Since the reciprocal of the growth rate of $s_{\boldsymbol{k},n}$ must be the smallest root of the denominator $h_{\boldsymbol{k}}(z)$ in (3.9), the respective overlap vector $\underline{r}_{\boldsymbol{k}}$ determines the growth rate of $s_{\boldsymbol{k},n}$. Because two different polynomials $h_{\boldsymbol{k}}(z)$ and $h_{\boldsymbol{k}}(z)$, resulting respectively from two different overlap vectors, may have the same smallest root, the number of asymptotic growth rates of $s_{\boldsymbol{k},n}$ must be upper-bounded by κ_L. The proof is then completed by the result from Proposition 4 that $s_{\boldsymbol{k},n}$ and $c_{\boldsymbol{k},n}$ have the same growth rate. □

One may find the number of asymptotically equivalent UW classes by a brutal force algorithm when L is small. With the help of Proposition 5, an efficient algorithm for its upper bound κ_L is available in [24], in which $\boldsymbol{r}_{\boldsymbol{k}}$ is regarded as *(auto)correlations* of a string. Some κ_L values are accordingly listed in Table 3.2. This table shows the trend, as being

Table 3.2: Numbers of asymptotically equivalent UW classes, κ_L. It is stated in [24] that the lower asymptotic bound $1/(2\ln(2)) \approx 0.72$ only holds for very large L; hence, this lower bound is not valid for $L \leq 13$ in this table .

L	1	2	3	4	5	6	7	8	9	10	11	12	13
κ_L	1	2	3	4	6	8	10	13	17	21	27	30	37
$\frac{\ln \kappa_L}{(\ln L)^2}$	–	1.44	1.91	.72	.69	.65	.61	.59	.59	.57	.57	.55	.55

pointed out in [16], that $\ln \kappa_L$ grows at the speed of $(\ln(L))^2$, or specifically,

$$\frac{1}{2\ln 2} \leq \liminf_{L\to\infty} \frac{\ln \kappa_L}{(\ln L)^2} \leq \limsup_{L\to\infty} \frac{\ln \kappa_L}{(\ln L)^2} \leq \frac{1}{2\ln \frac{3}{2}}. \tag{3.28}$$

Chapter 4

Encoding and Decoding Algorithms of UDOOCs

In this chapter, the encoding and decoding algorithms of UDOOCs are presented. Also provided are upper bounds for the averaged codeword length of the resulting UDOOC.

Denote by $\mathcal{U} = \{u_1, u_2, \cdots, u_M\}$ the source alphabet of size M to be encoded. Assume without loss of generality that $p_1 \geq p_2 \geq \cdots \geq p_M$, where p_i is the probability of occurrence for source symbol u_i.

Then, an optimal source coding scheme for UDOOCs should assign codewords of shorter lengths to messages with higher probabilities and reserve longer codewords for unlikely messages. By following this principle, the encoding mapping ϕ from $\mathcal{U}$ to $\mathcal{C}_{\boldsymbol{k}}$ should satisfy $\ell(\phi(u_i)) \leq \ell(\phi(u_j))$ whenever $i \leq j$, where $\ell(\phi(u_i))$ denotes the length of bit stream $\phi(u_i)$. The coding system thus requires a computation-based ordering of the words in $\mathcal{C}_{\boldsymbol{k}}$ according to their lengths. This can be achieved in terms of the recursions regarding $c_{\boldsymbol{k},n}$ (for example, (3.20)). As such, $\phi(u_1)$ must be the null word and the mapping ϕ must always form a bijection mapping between $\{u_i : F_{\boldsymbol{k},n-1} < i \leq F_{\boldsymbol{k},n}\}$ and $\mathcal{C}_{\boldsymbol{k}}(n)$ for every integer $n \geq 1$, where

$$F_{\boldsymbol{k},n} := \begin{cases} \sum_{i=0}^{n} c_{\boldsymbol{k},i}, & \text{if } n \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

This optimal assignment results in the average codeword length:

$$L_{\boldsymbol{k}} \;=\; \ell(\boldsymbol{k}) + \sum_{i=1}^{M} p_i \cdot \ell(\phi(u_i)), \tag{4.2}$$

where the first term $\ell(\boldsymbol{k})$ accounts for the insertion of UW $\boldsymbol{k}$ to separate adjacent codewords.

4.1 Upper Bounds on Average Codeword Length of UDOOCs

The average codeword length $L_{\boldsymbol{k}}$ is clearly a function of the source distributions and does not in general exhibit a close-form formula. In order to understand the general performance of UDOOCs, three upper bounds on $L_{\boldsymbol{k}}$ are established in this section. The first upper bound is applicable to the situation when the largest probability p_1 of source symbols is given. Other than p_1, the second upper bound additionally requires the knowledge of the source entropy. When both the largest and second largest probabilities (i.e., p_1 and p_2) of source symbols are present apart from the source entropy, the third upper bound can be used. Note that since the third upper bound holds for all UWs and requires no knowledge about $\boldsymbol{k}$, one may predict that the third upper bound could be worse than the other two. Experiments using English text from *Alice's Adventures in Wonderland* however indicate that such a prediction may not be always valid. Nevertheless, the second upper bound is better than the first one in most cases we examine. Details are given below.

Proposition 6 (The first upper bound on $L_{\boldsymbol{k}}$)**.** *For UW $\boldsymbol{k}$ of length L, the average codeword length $L_{\boldsymbol{k}}$ is upper-bounded as follows:*

$$L_{\boldsymbol{k}} \leq L + (1 - p_1)N_{\boldsymbol{k}} \tag{4.3}$$

where $N_{\boldsymbol{k}}$ is the smallest integer such that $\sum_{n=0}^{N_{\boldsymbol{k}}} c_{\boldsymbol{k},n} \geq M$.

Proof. It can be derived from (4.2) and $\ell(\phi(u_1)) = 0$ that

$$\begin{aligned} L_{\boldsymbol{k}} &= L + \sum_{i=2}^{M} p_i \ell(\phi(u_i)) \\ &\leq L + \sum_{i=2}^{M} p_i \max_{2 \leq i \leq M} \ell(\phi(u_i)) \\ &= L + (1 - p_1) N_{\boldsymbol{k}}. \end{aligned}$$

□

Proposition 7 (The second upper bound on $L_{\boldsymbol{k}}$). *Suppose $g_{\boldsymbol{k}} > 1$. Then*

$$L_{\boldsymbol{k}} \leq L + \frac{H(\mathcal{U}) + p_1 \log_2(p_1)}{\log_2(g_{\boldsymbol{k}})} + (1 - p_1)(1 - \log_{g_{\boldsymbol{k}}}(K_{\boldsymbol{k}}))$$

where $H(\mathcal{U}) = \sum_{m=1}^{M} p_m \log_2(1/p_m)$ is the source entropy with units in bits, $K_{\boldsymbol{k}}$ is a constant given by

$$K_{\boldsymbol{k}} = \min \left\{ g_{\boldsymbol{k}}^{1-n_m} \sum_{n=0}^{n_m-1} c_{\boldsymbol{k},n} \ : \ m = 2, \cdots, M \right\}, \tag{4.4}$$

and n_m is the smallest integer satisfying $F_{\boldsymbol{k},n_m} \geq \frac{1}{p_m}$.

Proof. By definition of n_m,

$$\sum_{n=0}^{n_m-1} c_{\boldsymbol{k},n} < \frac{1}{p_m} \leq \sum_{n=0}^{n_m} c_{\boldsymbol{k},n}.$$

It thus follows that

$$K_{\boldsymbol{k}} \ g_{\boldsymbol{k}}^{n_m-1} \leq \sum_{n=0}^{n_m-1} c_{\boldsymbol{k},n} < \frac{1}{p_m},$$

which by $g_{\boldsymbol{k}} > 1$ implies

$$n_m \leq 1 - \log_{g_{\boldsymbol{k}}}(K_{\boldsymbol{k}} p_m) = 1 - \log_{g_{\boldsymbol{k}}}(p_m) - \log_{g_{\boldsymbol{k}}}(K_{\boldsymbol{k}}).$$

On the other hand, that $p_1 \geq p_2 \geq \cdots \geq p_M$ implies that $p_m \leq \frac{1}{m}$ for $1 \leq m \leq M$, and hence $\sum_{n=0}^{n_m} c_{\boldsymbol{k},n} \geq \frac{1}{p_m} \geq m$ and $\ell(\phi(u_m)) \leq n_m$. Consequently,

$$\begin{aligned}
L_{\boldsymbol{k}} &= L + \sum_{m=2}^{M} p_m \ell(\phi(u_m)) \\
&\leq L + \sum_{m=2}^{M} p_m n_m \\
&\leq L + \sum_{m=2}^{M} p_m \left(1 - \log_{g_{\boldsymbol{k}}}(p_m) - \log_{g_{\boldsymbol{k}}}(K_{\boldsymbol{k}})\right) \\
&= L - \sum_{m=2}^{M} p_m \log_{g_{\boldsymbol{k}}}(p_m) + \sum_{m=2}^{M} p_m \left(1 - \log_{g_{\boldsymbol{k}}}(K_{\boldsymbol{k}})\right) \\
&= L + \frac{\mathrm{H}(\mathcal{U}) + p_1 \log_{g_{\boldsymbol{k}}}(p_1)}{\log_2(g_{\boldsymbol{k}})} + (1 - p_1)(1 - \log_{g_{\boldsymbol{k}}} K_{\boldsymbol{k}}).
\end{aligned}$$

□

The previous two upper bounds require the computations of either $N_{\boldsymbol{k}}$, or $g_{\boldsymbol{k}}$ and $K_{\boldsymbol{k}}$; hence, they are functions of UW $\boldsymbol{k}$. Next we provide a simple third upper bound that holds universally for all UWs.

Proposition 8 (The third upper bound on $L_{\boldsymbol{k}}$)**.** *For UW $\boldsymbol{k}$ of length $L > 2$,*

$$L_{\boldsymbol{k}} \leq L + \frac{H(\mathcal{U}) + p_1 \log_2(p_1) + p_2 \log_2(p_2)}{\log_2(2 - 2^{2-L})} + (2 - 2p_1 - p_2).$$

Proof. First, we claim that

$$c_{\boldsymbol{k},n} \geq 2^{n-2} \quad \text{for } 2 \leq n \leq L + 1. \tag{4.5}$$

This claim can be substantiated by proving that for any binary sequence $\boldsymbol{b} \in \mathbb{F}^m$, where $0 \leq m = n - 2 \leq L - 1$, there exist a prefix bit p and a suffix bit q, where p,$q \in \mathbb{F}$, such that $\boldsymbol{k}$ is not an internal subword of $\boldsymbol{k}p\boldsymbol{b}q\boldsymbol{k}$. The proof of (4.5) can thus be equivalently done in two steps: *i*) there exists q such that $\boldsymbol{k}$ is not a subword of $\boldsymbol{u} := \boldsymbol{b}qk_1^{L-1}$, and *ii*) there exists p such that $\boldsymbol{k}$ is not an internal subword of $\boldsymbol{k}p\boldsymbol{u}$.

Because i) trivially holds when $m = 0$, we only needs to prove it under $m > 0$. Utilizing the prove-by-contradiction argument, we suppose that $\boldsymbol{k}$ is a subword of both $\boldsymbol{b}qk_1^{L-1}$ and $\boldsymbol{b}\bar{q}k_1^{L-1}$, where $\bar{q} = 1 - q$. This implies the existence of indices $1 \le i < j \le m + 1$ such that

$$\underbrace{b_i \cdots b_m q k_1 \cdots k_{L-m+i-2}}_{=\boldsymbol{d}} = \underbrace{b_j \cdots b_m \bar{q} k_1 \cdots k_{L-m+j-2}}_{=\tilde{\boldsymbol{d}}} = \boldsymbol{k}$$

where we abuse the notations to let

$$\boldsymbol{d} = \begin{cases} \boldsymbol{b}qk_1 \cdots k_{L-m-1}, & \text{if } i = 1 \\ b_i \cdots b_m q k_1 \cdots k_{L-m+i-2}, & \text{if } 1 < i < m+1 \\ qk_1 \cdots k_{L-1}, & \text{if } i = m+1 \end{cases}$$

and similar notational abuse is applied to $\tilde{\boldsymbol{d}}$ and j. After canceling out common terms in the respective sums of the first $(m + 2 - i)$ bits of $\boldsymbol{d}$ and $\tilde{\boldsymbol{d}}$, we obtain

$$b_i + \cdots + b_{j-1} + q = \bar{q} + k_1 + \cdots + k_{j-i}.$$

Since $\boldsymbol{d} = \boldsymbol{k}$, the above then implies $q = \bar{q}$, which leads to a contradiction. The validity of i) is verified.

After verifying $\boldsymbol{u} = \boldsymbol{b}qk_1^{L-1} \in \mathcal{S}_{\boldsymbol{k}}(m + L)$, we can follow the proof of Proposition 4 to confirm ii) (See the paragraph regarding (3.11) and (3.12)). The claim in (4.5) is thus validated. Note that the equality in (4.5) holds if, and only if, $\boldsymbol{k}$ is all-zero or all-one.

Next, we note also from the proof of Proposition 4 that $c_{\boldsymbol{k},n+2} \ge s_{\boldsymbol{k},n}$ for $n \ge L$. Since $s_{\boldsymbol{k},L} = 2^L - 1$, we immediately have $c_{\boldsymbol{k},L+2} \ge 2^L - 1$. On the other hand, we can obtain by (3.23) that[1]

$$\frac{s_{\boldsymbol{k},n}}{s_{\boldsymbol{k},n-1}} \ge 2 - 2^{2-L} \text{ for } n \ge L. \tag{4.6}$$

[1]We can prove (4.6) by induction. Extending the definition of $\mathcal{S}_{\boldsymbol{k}}(n)$ in (3.6), we obtain that $s_{\boldsymbol{k},n} = 2^n$ for $0 \le n < L$. This implies

$$\frac{s_{\boldsymbol{k},L}}{s_{\boldsymbol{k},L-1}} = \frac{2^L - 1}{2^{L-1}} = 2 - 2^{1-L} \ge 2 - 2^{2-L}$$

and

$$\frac{s_{\boldsymbol{k},m}}{s_{\boldsymbol{k},m-1}} = \frac{2^m}{2^{m-1}} = 2 \ge 2 - 2^{2-L} \text{ for all } 1 \le m < L.$$

This concludes:

$$c_{\boldsymbol{k},n} \geq \begin{cases} 1, & \text{if } 0 \leq n \leq 1, \\ 2^{n-2}, & \text{if } 2 \leq n \leq L+1, \\ (2-2^{2-L})^{n-L-2}(2^L-1), & \text{if } n \geq L+2. \end{cases} \tag{4.7}$$

where $c_{\boldsymbol{k},0} = 1$ because $\mathcal{C}_{\boldsymbol{k}}(0)$ contains the null codeword, and $c_{\boldsymbol{k},1} \geq 1$ can be verified again by that $\boldsymbol{k}$ cannot be the internal subword of both $\boldsymbol{k}p\boldsymbol{k}$ and $\boldsymbol{k}\bar{p}\boldsymbol{k}$.[2] (4.7) then indicates that if $2^L - 1 \geq (2-2^{2-L})^L$, we can immediately have the desired exponential lower bound for $c_{\boldsymbol{k},n}$, i.e.,

$$c_{\boldsymbol{k},n} \geq \begin{cases} 1, & \text{if } 0 \leq n \leq 1, \\ (2-2^{2-L})^{n-2}, & \text{if } n \geq 2. \end{cases} \tag{4.8}$$

We accordingly subtract $2^L - 1$ from $(2-2^{2-L})^L$ to confirm $2^L - 1 \geq (2-2^{2-L})^L$ below:

$$2^L - 1 - (2-2^{2-L})^L > 2^L - 1 - 2^{L-1}(2-2^{2-L}) = 1.$$

Hence, codeword lengths of the optimal UDOOC code must satisfy:[3]

$$\ell(\phi(u_m)) \leq \log_{2-2^{2-L}}(m) + 2 \quad \text{for } m \geq 3.$$

Now we suppose that for some $n \geq L$ fixed, (4.6) is true for all $1 \leq m \leq n$, i.e.,

$$\frac{s_{\boldsymbol{k},m}}{s_{\boldsymbol{k},m-1}} \geq 2 - 2^{2-L} \quad \text{for all } 1 \leq m \leq n.$$

Then, we derive by (3.23) that

$$\begin{aligned} \frac{s_{\boldsymbol{k},n+1}}{s_{\boldsymbol{k},n}} &\geq 2 - \frac{s_{\boldsymbol{k},n-L+1}}{s_{\boldsymbol{k},n}} \geq 2 - \frac{s_{\boldsymbol{k},n-L+1}}{s_{\boldsymbol{k},n-L+1}(2-2^{2-L})^{L-1}} \\ &= 2 - (2-2^{2-L})^{1-L} \geq 2 - 2^{2-L}. \end{aligned}$$

This completes the proof of (4.6).

[2]If it were not true, then there exist $2 \leq i < j \leq L+1$ such that $\boldsymbol{k} = k_i \cdots k_L p k_1 \cdots k_{i-2} = k_j \cdots k_L \bar{p} k_1 \cdots k_{j-2}$; hence, $p - k_i = \bar{p} - k_j$ with $k_i = k_j = k_1$. The desired contradiction is obtained.

[3]By (4.1) and (4.8), we have that for $m \geq 3$ and $n = \ell(\phi(u_m))$,

$$m > F_{\boldsymbol{k},n-1} = \sum_{t=0}^{n-1} c_{\boldsymbol{k},t} \geq 2 + \frac{(2-2^{2-L})^{n-2} - 1}{1-2^{2-L}}$$

which implies $\log_{2-2^{2-L}}[(m-2)(1-2^{2-L})+1] + 2 > n = \ell(\phi(u_m))$. Since $(m-2)(1-2^{2-L})+1 \leq m$ for $m \geq 2 - 2^{L-2}$, we obtain

$$\ell(\phi(u_m)) < \log_{2-2^{2-L}}[(m-2)(1-2^{2-L})+1] + 2 \leq \log_{2-2^{2-L}}(m) + 2.$$

Consequently,

$$\begin{aligned} L_{\boldsymbol{k}} &= L + \sum_{m=2}^{M} p_m \ell(\phi(u_m)) \\ &= L + p_2 + \sum_{m=3}^{M} p_m \ell(\phi(u_m)) \\ &\leq L + p_2 + \sum_{m=3}^{M} p_m \left(\log_{2-2^{2-L}}(m) + 2\right) \\ &= L + 2 - 2p_1 - p_2 + \sum_{m=3}^{M} p_m \log_{2-2^{2-L}}(m) \\ &\leq L + 2 - 2p_1 - p_2 + \sum_{m=3}^{M} p_m \log_{2-2^{2-L}}\left(\frac{1}{p_m}\right) \\ &= L + 2 - 2p_1 - p_2 + \frac{\mathrm{H}(\mathcal{U}) + p_1 \log_2(p_1) + p_2 \log_2(p_2)}{\log_2(2 - 2^{2-L})} \end{aligned} \tag{4.9}$$

where (4.9) follows from that $p_1 \geq p_2 \geq \cdots \geq p_M$ implies $p_m \leq \frac{1}{m}$ for $1 \leq m \leq M$. □

We can further generalize the propositions on upper bounds of $L_{\boldsymbol{k}}$ by grouping t source symbols to form a new grouped source for UDOOC compression. The alphabet of the new source is therefore $\mathcal{U}^t$ of size M^t. Let $L_{\boldsymbol{k},t}$ be the per-letter average codeword length of UDOOCs for the grouped source. Then, by Proposition 7, we can upper-bound $L_{\boldsymbol{k},t}$ by:

$$L_{\boldsymbol{k},t} \leq \frac{1}{t}\left(L + \frac{\mathrm{H}(\mathcal{U}^t) + q_1 \log_2(q_1)}{\log_2(g_{\boldsymbol{k}})} + (1 - q_1)(1 - \log_{g_{\boldsymbol{k}}}(K_{\boldsymbol{k},t}))\right),$$

where

$$K_{\boldsymbol{k},t} = \min\left\{ g_{\boldsymbol{k}}^{1-n_{m,t}} \sum_{n=0}^{n_{m,t}-1} c_{\boldsymbol{k},n} \,:\, m = 2, \cdots, M^t \right\}, \tag{4.10}$$

$n_{m,t}$ is the smallest integer satisfying

$$\sum_{n=0}^{n_{m,t}} c_{\boldsymbol{k},n} \geq \frac{1}{q_m},$$

and q_m is the mth largest probability of the grouped source. For independent and identically distributed (i.i.d.) source, we have $\mathrm{H}(\mathcal{U}^t) = t\,\mathrm{H}(\mathcal{U})$, and $K_{\boldsymbol{k},t}$ can be shown to converge to

$$\begin{aligned} K_{\boldsymbol{k},\infty} &:= \lim_{t\to\infty} K_{\boldsymbol{k},t} \\ &= \frac{\underline{x}_{\boldsymbol{k}}^{\top}\mathrm{adj}\,(\mathtt{I}-\mathtt{A}_{\boldsymbol{k}}z)\,\mathtt{A}_{\boldsymbol{k}}^{L-1}\underline{y}_{\boldsymbol{k}}}{\det\,(\mathtt{I}-\mathtt{A}_{\boldsymbol{k}}z)}(1-g_{\boldsymbol{k}}z)\mid_{z=g_{\boldsymbol{k}}^{-1}} \end{aligned}$$

where the last step follows from the conventional expansion theory for power series and also from the fact of $g_{\boldsymbol{k}}$ being the unique maximal eigenvalue of the adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$ under $L > 2$ (cf. Proposition 3).

Also, from the power series expansion, we have that $c_{\boldsymbol{k},n} = \sum_i a_{i,n}\lambda_i^n$, where $\{\lambda_i\}$ is the set of nonzero distinct eigenvalues of $\mathtt{A}_{\boldsymbol{k}}$, and $a_{i,n}$ is the coefficient associated with λ_i. In particular, assuming λ_1 is the largest eigenvalue, we can establish that $K_{\boldsymbol{k},\infty} = a_{1,n} = a_1$, where the second equality emphasizes that $a_{1,n}$ is a constant independent of n since $\lambda_1^{-1} = g_{\boldsymbol{k}}^{-1}$ is a simple zero for $h_{\boldsymbol{k}}(z)$ when the digraph $G_{\boldsymbol{k}}$ is strongly connected.

By further exploring the above derivation, the next proposition can be resulted.

Proposition 9. *Given $g_{\boldsymbol{k}} > 1$ and i.i.d. source, we have*

$$\lim_{t\to\infty} L_{\boldsymbol{k},t} \le \frac{H(\mathcal{U})}{\log_2(g_{\boldsymbol{k}})} \le \frac{H(\mathcal{U})}{\log_2(2-2^{2-L})}. \tag{4.11}$$

Two remarks are made based on Proposition 9. First, the larger asymptotic bound in (4.11) immediately gives

$$\lim_{L\to\infty}\lim_{t\to\infty} L_{\boldsymbol{k},t} = \mathrm{H}(\mathcal{U}).$$

Hence, if both t and L are sufficiently large, the per-letter average codeword length of UDOOCs can achieve the entropy rate $\mathrm{H}(\mathcal{U})$ of the i.i.d. source. Secondly, the better asymptotic bound in (4.11) is actually achievable by taking the all-zero UW when the source is uniformly distributed. In other words,

$$\lim_{t\to\infty} L_{\boldsymbol{a},t} = \frac{\mathrm{H}(\mathcal{U})}{\log_2(g_{\boldsymbol{a}})} = \frac{\log_2(M)}{\log_2(g_{\boldsymbol{a}})},$$

where $\boldsymbol{a} = 0 \ldots 0$. For better readability, we defer the confirmation of $\lim_{t\to\infty} L_{\boldsymbol{a},t} = \log_2(M)/\log_2(g_{\boldsymbol{a}})$ in Appendix D.

Tables 4.1 and 4.2 evaluate the bounds for English text source with letter probabilities from [35] and the true source from *Alice's Adventure in Wonderland* with empirical frequencies directly calculated from the book, respectively. The source alphabet of the English text and that from *Alice's Adventure in Wonderland* is of size 27, where letters of upper and lower cases are regarded the same and all symbols other than the 26 English letters are treated as one. It can be observed from Table 4.1 that bound (4.3) is always the best among all three bounds but still has a visible gap to the resultant average codeword length $L_{\boldsymbol{k}}$. Table 4.2 however shows that the three bounds may take turn to be on top of the other two. For example, under $\boldsymbol{k} = \boldsymbol{a}$, (4.3), (7) and (8) are the lowest when $(L,t) = (3,1)$, $(L,t) = (5,2)$ and $(L,t) = (6,3)$, respectively. Table 4.2 also indicates that enlarging the value of t may help improving the per-letter average codeword length as well as the bounds of UDOOCs. Comparison of the per-letter average codeword length of UDOOCs with the source entropy will be provided later in the simulation chapter.

Table 4.1: Upper bounds (4.3), (7) and (8) on the average codeword length $L_{\boldsymbol{k}}$ of UDOOCs for English text source with letter probabilities from [35]. Here, $\boldsymbol{a} = 0\cdots0$ and $\boldsymbol{b} = 0\cdots01$.

$\boldsymbol{k}$		$L=3$	$L=4$	$L=5$	$L=6$
$\boldsymbol{a}$	$L_{\boldsymbol{k}}$	6.432	7.411	8.411	9.411
	(4.3)	8.330	9.330	10.330	11.330
	(7)	9.648	10.555	11.536	12.537
$\boldsymbol{b}$	$L_{\boldsymbol{k}}$	5.215	6.185	7.185	8.185
	(4.3)	6.553	7.553	8.553	9.553
	(7)	8.569	9.145	10.031	10.988
–	(8)	10.831	10.140	10.652	11.456

Table 4.2: Upper bounds (4.3), (7) and (8) on the average codeword length $L_{\boldsymbol{k}}$ of UDOOCs for the source from *Alice's Adventure in Wonderland*. Here, $\boldsymbol{a} = 0\cdots0$ and $\boldsymbol{b} = 0\cdots01$.

$\boldsymbol{k}$			$L=3$	$L=4$	$L=5$	$L=6$
$\boldsymbol{a}$	$L_{\boldsymbol{k},t}$	$t=1$	5.773	6.757	7.757	7.757
		$t=2$	4.498	4.920	5.397	5.891
		$t=3$	3.862	4.089	4.388	4.709
	(4.3)	$t=1$	7.459	8.459	9.459	10.459
		$t=2$	6.569	7.069	7.569	7.608
		$t=3$	5.770	5.786	6.119	6.134
	(7)	$t=1$	8.738	9.647	10.627	11.625
		$t=2$	6.549	6.886	7.333	7.815
		$t=3$	5.375	5.532	5.805	6.115
$\boldsymbol{b}$	$L_{\boldsymbol{k},t}$	$t=1$	4.792	5.774	6.774	7.774
		$t=2$	3.791	4.134	4.598	5.090
		$t=3$	3.455	3.532	3.802	4.115
	(4.3)	$t=1$	6.716	6.973	7.973	8.973
		$t=2$	6.108	6.147	6.647	7.147
		$t=3$	5.452	5.150	5.483	5.816
	(7)	$t=1$	7.880	8.479	9.371	10.331
		$t=2$	6.079	6.089	6.472	6.936
		$t=3$	5.228	5.019	5.216	5.519
–	(8)	$t=1$	9.676	9.221	9.801	10.632
		$t=2$	8.106	7.035	7.399	7.816
		$t=3$	6.947	5.815	5.726	5.889

4.2 General Encoding and Decoding Mappings for UDOOCs

In this section, the encoding and decoding mappings for a UDOOC with general UW are introduced.

The practice of UDOOC requires a bijection mapping $\phi_{\boldsymbol{k}}$ from the set of source letters $\mathcal{U}_{\boldsymbol{k}}(n) := \{u_m : F_{\boldsymbol{k},n-1} < m \leq F_{\boldsymbol{k},n}\}$ to the set of length-n codewords $\mathcal{C}_{\boldsymbol{k}}(n)$. Since the resulting average codeword length will be the same for any bijection mapping from $\mathcal{U}_{\boldsymbol{k}}(n)$ to $\mathcal{C}_{\boldsymbol{k}}(n)$, we are free to devise one that facilities efficient encoding and decoding of message u_m. The bijection encoding mapping $\phi_{\boldsymbol{k}}$ that we propose is described in the following.

We define for any binary stream $\boldsymbol{d}$ of length $\leq n$,

$$\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}, n) := \{\boldsymbol{c} \in \mathcal{C}_{\boldsymbol{k}}(n) \,:\, \boldsymbol{d} \text{ is a prefix of } \boldsymbol{c} \text{ or } \boldsymbol{d} = \boldsymbol{c}\}\,. \tag{4.12}$$

Obviously, $\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}, n) \cap \mathcal{C}_{\boldsymbol{k}}(\tilde{\boldsymbol{d}}, n) = \emptyset$ for every pair of distinct $\boldsymbol{d}$ and $\tilde{\boldsymbol{d}}$ of the same length, and for any $1 \leq i \leq n$ fixed,

$$\mathcal{C}_{\boldsymbol{k}}(n) \;=\; \bigcup_{\boldsymbol{d} \in \mathbb{F}_2^i} \mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}, n). \tag{4.13}$$

Then, the proposed encoding mapping $\phi_{\boldsymbol{k}}$ decides the codeword $\phi_{\boldsymbol{k}}(u_m) = c_1 c_2 \cdots c_n$ of source letter u_m recursively according to:

$$c_i = \begin{cases} 0, & \text{if } \rho_{i-1} \leq |\mathcal{C}_{\boldsymbol{k}}(c_1 \cdots c_{i-1} 0, n)| \\ 1, & \text{if } \rho_{i-1} > |\mathcal{C}_{\boldsymbol{k}}(c_1 \cdots c_{i-1} 0, n)| \end{cases} \tag{4.14}$$

where the progressive metric ρ_i is also maintained recursively as:

$$\begin{aligned} \rho_i \;&:=\; \rho_{i-1} - c_i \left|\mathcal{C}_{\boldsymbol{k}}(c_1 \cdots c_{i-1} 0, n)\right| \\ &=\; \begin{cases} \rho_{i-1}, & \text{if } c_i = 0 \\ \rho_{i-1} - \left|\mathcal{C}_{\boldsymbol{k}}(c_1 \cdots c_{i-1} 0, n)\right|, & \text{if } c_i = 1 \end{cases} \end{aligned} \tag{4.15}$$

with initial value $\rho_0 = m - F_{\boldsymbol{k},n-1}$. This encoding mapping actually assigns codewords according to the exact alphabetical order. Taking $\boldsymbol{k} = 010$ as an example, we can see from Figure 2.2 that the seven codewords of length 4, i.e., 0000, 0011, 0110, 0111, 1100, 1110 and 1111, will be respectively assigned to source letters u_9, u_{10}, u_{11}, u_{12}, u_{13}, u_{14} and u_{14}. The progressive metrics $\rho_0\rho_1\rho_2\rho_3$ for source letter u_{11} are 3311 with $|\mathcal{C}_{\boldsymbol{k}}(0, 4)| = 4$, $|\mathcal{C}_{\boldsymbol{k}}(00, 4)| = 2$, $|\mathcal{C}_{\boldsymbol{k}}(010, 4)| = 0$ and $|\mathcal{C}_{\boldsymbol{k}}(0110, 4)| = 1$. Clearly, the progressive metric records the remaining number of walks that should take from the current codeword node at level n of the UDOOC code tree to the final node at the same level, counting from top. Note again that given m (equivalently, u_m), n can be determined via $F_{\boldsymbol{k},n-1} < m \leq F_{\boldsymbol{k},n}$. At the end of the nth recursion, we must have

$$m \;=\; F_{\boldsymbol{k},n-1} + \sum_{i=1}^{n-1} c_i \left|\mathcal{C}_{\boldsymbol{k}}(c_1^{i-1} 0, n)\right| + 1. \tag{4.16}$$

We emphasize that (4.16) actually gives the corresponding computation-based decoding function $\psi_{\boldsymbol{k}} : \mathcal{C}_{\boldsymbol{k}}(n) \to \mathcal{U}_{\boldsymbol{k}}(n)$ for codeword $\boldsymbol{c}$ of length n.

One straightforward way to implement $\phi_{\boldsymbol{k}}$ and $\psi_{\boldsymbol{k}}$ is to pre-store the value of $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)|$ for every $\boldsymbol{d}$ and n. By considering the huge number of possibilities on prefix $\boldsymbol{d}$ for each n, this straightforward approach does not seem to be an attractive one.

Alternatively, we found that $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)|$ can be obtained through adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$ introduced in Chapter 2. The advantage of this alternative approach is that no need to pre-store or pre-construct any part of the codebook $\mathcal{C}_{\boldsymbol{k}}$, and the value of $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)|$ is computed only when it is acquired during the encoding or decoding processes. Furthermore, for specific UWs such as $00\ldots0$ or $00\ldots01$, we can further simplify the acquired computations.

In the following sections, we will first introduce the encoding and decoding algorithms for specific UWs as they can be straightforwardly understood. Generalization of these algorithms to general ones that compute $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)|$ for general UW will be next done.

4.3 Encoding and Decoding Algorithms for UW $= 11\ldots1$

It has been inferred from Proposition 2 that the encoding and decoding of the UDOOC with UW $\boldsymbol{k} = 00\ldots0$ can be equivalently done through the encoding and decoding of the UDOOC with UW $\boldsymbol{k} = 11\ldots1$ as one can be obtained from the other by binary complementing. Thus, we only focus on the case of $\boldsymbol{k} = 11\ldots1$ in this section.

For this specific UW, we observe that a codeword $\boldsymbol{c} = \boldsymbol{d}0\boldsymbol{b} \in \mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)$ if, and only if, $0\boldsymbol{b} \in \mathcal{C}_{\boldsymbol{k}}(n - \ell(\boldsymbol{d}))$ is a codeword of length $n - \ell(\boldsymbol{d})$, where $\ell(\boldsymbol{d})$ is the length of prefix bitstream $\boldsymbol{d}$. We thus obtain

$$|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)| = c_{\boldsymbol{k}, n-\ell(\boldsymbol{d})}. \tag{4.17}$$

The LCCDE for $\boldsymbol{k} = 11\ldots1$ gives

$$c_{\boldsymbol{k},n} = \sum_{i=1}^{L} c_{\boldsymbol{k},n-i} \tag{4.18}$$

with initial values

$$c_{\boldsymbol{k},n} = \begin{cases} 1, & n = 0, 1, 2, \\ 2^{n-2}, & n = 3, \ldots, L \end{cases} \tag{4.19}$$

Based on (4.17), (4.18) and (4.19), the algorithmic encoding and decoding procedures can be described below.

Algorithm 1 Encoding of UDOOC with $\boldsymbol{k} = 11\ldots1$

Input: Index m for message u_m

Output: Codeword $\phi_{\boldsymbol{k}}(u_m) = c_1 \ldots c_n$

1: Compute $c_{\boldsymbol{k},0}, c_{\boldsymbol{k},1}, c_{\boldsymbol{k},2}, \ldots$ using (4.18) and (4.19) to determine the smallest n such that $F_{\boldsymbol{k},n} \geq m$. If $n = 0$, then $\phi_{\boldsymbol{k}}(u_m) =$ null and stop the algorithm.
2: Initialize $\rho_0 \leftarrow m - F_{\boldsymbol{k},n-1}$
3: **for** $i = 1$ to n **do**
4: **if** $\rho_{i-1} \leq c_{\boldsymbol{k},n-i+1}$ **then**
5: $c_i \leftarrow 0$ and $\rho_i \leftarrow \rho_{i-1}$
6: **else**
7: $c_i \leftarrow 1$ and $\rho_i \leftarrow \rho_{i-1} - c_{\boldsymbol{k},n-i+1}$
8: **end if**
9: **end for**

Algorithm 2 Decoding of UDOOC with $\boldsymbol{k} = 11\ldots1$

Input: Codeword $\boldsymbol{c} = c_1 \ldots c_n$

Output: Index m for message $u_m = \psi_{\boldsymbol{k}}(\boldsymbol{c})$

1: Compute $c_{\boldsymbol{k},0}, c_{\boldsymbol{k},1}, \ldots, c_{\boldsymbol{k},n}$ using (4.18) and (4.19)
2: Initialize $m \leftarrow F_{\boldsymbol{k},n-1} + 1$
3: **for** $i = 1$ to n **do**
4: **if** $c_i = 1$ **then**
5: $m \leftarrow m + c_{\boldsymbol{k},n-i+1}$
6: **end if**
7: **end for**

4.4 Encoding and Decoding Algorithms for $11\cdots10$

Again, Proposition 2 infers that the encoding and decoding of the UDOOC with UW $\boldsymbol{k} = 00\ldots01$ can be equivalently done through the encoding and decoding of the UDOOC with UW $\boldsymbol{k} = 11\ldots10$. We simply assume $\boldsymbol{k} = 11\cdots10$ as a demonstration.

Derive from (3.10) that for $\boldsymbol{k} = 11\cdots10$,

$$c_{\boldsymbol{k},n} = 2c_{\boldsymbol{k},n-1} - c_{\boldsymbol{k},n-L} \tag{4.20}$$

with initial condition

$$c_{\boldsymbol{k},n} = \begin{cases} 1, & n = 0 \\ 2^n, & n = 1,\ldots,L-1 \\ 2^L - 1, & n = L. \end{cases} \tag{4.21}$$

It remains to determine $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)|$. Observe that $\boldsymbol{d}0\boldsymbol{b} \in \mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)$ if, and only if, $\boldsymbol{b} \in \mathcal{C}_{\boldsymbol{k}}(n-i-1)$; hence, $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)| = c_{\boldsymbol{k},n-\ell(d)-1}$. We summarize the encoding and decoding algorithms of UDOOCs with $\boldsymbol{k} = 11\ldots10$ in Algorithms 3 and 4, respectively.

Algorithm 3 Encoding of UDOOC with $\boldsymbol{k} = 11\cdots10$

Input: Index m for message u_m

Output: Codeword $\phi_{\boldsymbol{k}}(u_m) = c_1\ldots c_n$

1: Compute $c_{\boldsymbol{k},0}, c_{\boldsymbol{k},1}, c_{\boldsymbol{k},2}, \ldots$ using (4.20) and (4.21) to determine the smallest n such that $F_{\boldsymbol{k},n} \geq m$. If $n = 0$, then $\phi_{\boldsymbol{k}}(u_m) =$ null and stop the algorithm.
2: Initialize $\rho_0 \leftarrow m - F_{\boldsymbol{k},n-1}$
3: **for** $i = 1$ to n **do**
4: **if** $\rho_{i-1} \leq c_{\boldsymbol{k},n-i}$ **then**
5: $c_i \leftarrow 0$ and $\rho_i \leftarrow \rho_{i-1}$
6: **else**
7: $c_i \leftarrow 1$ and $\rho_i \leftarrow \rho_{i-1} - c_{\boldsymbol{k},n-i}$
8: **end if**
9: **end for**

Algorithm 4 Decoding of UDOOC with $\boldsymbol{k} = 11\cdots10$

Input: Codeword $\boldsymbol{c} = c_1 \ldots c_n$
Output: Index m for message $u_m = \psi_{\boldsymbol{k}}(\boldsymbol{c})$

```
Compute c_{k,0}, c_{k,1}, ..., c_{k,n} using (4.20) and (4.21)
Initialize m ← F_{k,n-1} + 1
for i = 1 to n do
    if c_i = 1 then
        m ← m + c_{k,n-i}
    end if
end for
```

4.5 Encoding and Decoding Algorithms for General UW $\boldsymbol{k}$

From the previous two sections, it is clear that in the encoding algorithm, we only need to keep the most recent ρ_{i-1} for the examination in (4.14), instead of retaining sequentially all of $\rho_0 \ldots \rho_{n-1}$. We address the recursion form for the update of ρ_i in (4.15) only to facilitate our interpretation on the operational function of the progressive metric. The same tradition will be followed in the presentation of the general encoding algorithm below, where a progressive matrix $\mathtt{D}_i$ is used in addition to the progressive metric ρ_i.

The general encoding algorithm of a UDOOC consists of two phases. Given the index m, we first identify the smallest n such that $F_{\boldsymbol{k},n} \geq m$. Note that the computation of $F_{\boldsymbol{k},n}$ requires the knowledge of $c_{\boldsymbol{k},n}$, which can be recursively obtained using the LCCDE in (3.16). In the second phase, as seen from the two previous sections, we need to determine the size of $\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}, n)$ for any prefix $\boldsymbol{d}$ with $\ell(\boldsymbol{d}) \leq n$. Thus, our target in this section is to address an expression for the size of $\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}, n)$ that holds for general $\boldsymbol{k}$ and $\boldsymbol{d}$.

Define $E_{\boldsymbol{k},0}$ and $E_{\boldsymbol{k},1}$ for digraph $G_{\boldsymbol{k}} = (V, E_{\boldsymbol{k}})$ as

$$E_{\boldsymbol{k},0} := \{(\boldsymbol{i}, \boldsymbol{j}) \in E_{\boldsymbol{k}} \ : \ j_{L-1} = 0\} \tag{4.22}$$
$$E_{\boldsymbol{k},1} := \{(\boldsymbol{i}, \boldsymbol{j}) \in E_{\boldsymbol{k}} \ : \ j_{L-1} = 1\}\,. \tag{4.23}$$

Literally speaking, $E_{\boldsymbol{k},0}$ (resp. $E_{\boldsymbol{k},1}$) is the set of edges in $E_{\boldsymbol{k}}$, whose ending vertex has its

last bit j_{L-1} equal to 0 (resp. 1). Let $\mathtt{A}_{\boldsymbol{k},0}$ and $\mathtt{A}_{\boldsymbol{k},1}$ be the adjacency matrices respectively for digraphs $G_{\boldsymbol{k},0} = (V, E_{\boldsymbol{k},0})$ and $G_{\boldsymbol{k},1} = (V, E_{\boldsymbol{k},1})$. Obviously, $\mathtt{A}_{\boldsymbol{k}} = \mathtt{A}_{\boldsymbol{k},0} + \mathtt{A}_{\boldsymbol{k},1}$. Based on the two adjacency matrices, we derive

$$|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}, n)| \;=\; \underline{x}_{\boldsymbol{k}}^{\top} \mathtt{D}_i \mathtt{A}_{\boldsymbol{k}}^{(n+L-1)-i} \underline{y}_{\boldsymbol{k}} \tag{4.24}$$

where for a prefix stream $\boldsymbol{d} = d_1 \dots d_i$,

$$\mathtt{D}_i := \prod_{t=1}^{i} \mathtt{A}_{\boldsymbol{k},d_t}, \tag{4.25}$$

and $\underline{x}_{\boldsymbol{k}}$ and $\underline{y}_{\boldsymbol{k}}$ are the initial and ending vectors for digraph $G_{\boldsymbol{k}}$ defined in Chapter 3.1. With (4.24) and (4.25), the general encoding and decoding algorithms are given in Algorithms 5 and 6, respectively, where their verification is relegated to Appendix C for better readability.

Algorithm 5 Encoding of UDOOC with General $\boldsymbol{k}$

Input: Index m for message u_m
Output: Codeword $\phi_{\boldsymbol{k}}(u_m) = c_1 \dots c_n$

1: Compute $c_{\boldsymbol{k},0}, c_{\boldsymbol{k},1}, c_{\boldsymbol{k},2}, \dots$ using (3.16) to determine the smallest n such that $F_{\boldsymbol{k},n} \geq m$. If $n = 0$, then $\phi_{\boldsymbol{k}}(u_m) =$ null and stop the algorithm.
2: Initialize $\rho_0 \leftarrow m - F_{\boldsymbol{k},n-1}$ and $\mathtt{D}_0 \leftarrow \mathtt{I}$
3: **for** $i = 1$ to n **do**
4: Compute $temp \leftarrow \underline{x}_{\boldsymbol{k}}^{\top} \mathtt{D}_{i-1} \mathtt{A}_{\boldsymbol{k},0} \mathtt{A}_{\boldsymbol{k}}^{(n+L-1)-i} \underline{y}_{\boldsymbol{k}}$
5: **if** $\rho_{i-1} \leq temp$ **then**
6: $c_i \leftarrow 0$, $\rho_i \leftarrow \rho_{i-1}$ and $\mathtt{D}_i \leftarrow \mathtt{D}_{i-1}\mathtt{A}_{\boldsymbol{k},0}$
7: **else**
8: $c_i \leftarrow 1$, $\rho_i \leftarrow \rho_{i-1} - temp$ and $\mathtt{D}_i \leftarrow \mathtt{D}_{i-1}\mathtt{A}_{\boldsymbol{k},1}$
9: **end if**
10: **end for**

4.6 Complexity Reduction of the Encoding and Decoding Algorithms for General UW $\boldsymbol{k}$

The matrix expressions in (4.24) and (4.25) facilitate the presentation of Algorithms 5 and 6 for general UW; however, their implementation involve computation-extensive matrix operations. Since the entries in each row or column of $\mathtt{A}_{\boldsymbol{k}}$ are mostly 0's except at most two

Algorithm 6 Decoding of UDOOC with General $\boldsymbol{k}$

Input: Codeword $\boldsymbol{c} = c_1 \dots c_n$
Output: Index m for message $u_m = \psi_{\boldsymbol{k}}(\boldsymbol{c})$

```
Compute c_{k,0}, c_{k,1}, ..., c_{k,n} using (3.16)
Initialize m ← F_{k,n-1} + 1 and D_0 ← I
for i = 1 to n do
    if c_i = 1 then
        m ← m + x_k^⊤ D_{i-1} A_{k,0} A_k^{(n+L-1)-i} y_k
    end if
    D_i ← D_{i-1} A_{k,c_i}
end for
```

1's, the complexity of computing

$$c_{\boldsymbol{k},i} = \underline{x}_{\boldsymbol{k}}^{\top} \mathtt{A}_{\boldsymbol{k}}^{i+L-1} \underline{y}_{\boldsymbol{k}}$$

and

$$|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)| = \underline{x}_{\boldsymbol{k}}^{\top} \mathtt{D}_{\ell(\boldsymbol{d})} \mathtt{A}_{\boldsymbol{k},0} \mathtt{A}_{\boldsymbol{k}}^{n+L-\ell(\boldsymbol{d})} \underline{y}_{\boldsymbol{k}}$$

should be reducible. The key is to utilize a finite state machine to construct the digraph $G_{\boldsymbol{k}}$, by which approach $c_{\boldsymbol{k},i}$ and $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)|$ can be evaluated without resorting to matrix operations.

Notations that are used to describe the finite state machine are addressed first. Let $\mathfrak{S}_{L-1} = \{s_{00\cdots0}, s_{00\cdots01}, \cdots, s_{11\cdots1}\}$ be the set of states indexed by all binary bit-streams of length $L-1$. We say $s_{\boldsymbol{i}} = s_{i_1 \cdots i_{L-1}}$ is a *counting state* if the $(i+1)$th component of $\mathtt{A}_{\boldsymbol{k}}^{L-1} \underline{y}_{\boldsymbol{k}}$ is 1, where i is the integer corresponding to binary representation of $\boldsymbol{i} = i_1 \dots i_{L-1}$. Denote by $\mathfrak{C}_{\boldsymbol{k}}$ the set of all counting states corresponding to $\boldsymbol{k}$. Also, denote by $\mathfrak{N}_{\boldsymbol{k},\mathrm{h}}(s_{\boldsymbol{i}})$ the set of states that link directionally to $s_{\boldsymbol{i}}$, and by $\mathfrak{N}_{\boldsymbol{k},\mathrm{t}}(s_{\boldsymbol{i}})$ the set of states that are linked directionally by $s_{\boldsymbol{i}}$. Further, $\mathfrak{N}_{\boldsymbol{k},\mathrm{t},0}(s_{\boldsymbol{i}})$ is the set of states that link to $s_{\boldsymbol{i}}$ via a so-called 0-edge in $E_{\boldsymbol{k},0}$. We can similarly define $\mathfrak{N}_{\boldsymbol{k},\mathrm{h},1}(s_{\boldsymbol{i}})$, $\mathfrak{N}_{\boldsymbol{k},\mathrm{t},0}(s_{\boldsymbol{i}})$ and $\mathfrak{N}_{\boldsymbol{k},\mathrm{t},1}(s_{\boldsymbol{i}})$.

In our state machine, we associate each state $s_{\boldsymbol{i}}$ with an integer. Without ambiguity, we conveniently use $s_{\boldsymbol{i}}$ to also denote the integer associated with it. Define an operator

$\Xi_{\boldsymbol{k}} = \Xi_{\boldsymbol{k}}(\mathfrak{S})$ that operates on a set of states $\mathfrak{S}$, which updates the value associated with each state according to:

$$s_{\boldsymbol{i}} \leftarrow \sum_{s_{\boldsymbol{j}} \in \mathfrak{N}_{\boldsymbol{k},\mathrm{h}}(s_{\boldsymbol{i}})} s_{\boldsymbol{j}} \quad \text{for all } s_{\boldsymbol{i}} \in \mathfrak{S}.$$

It should be noted that all states in $\mathfrak{S}$ are updated by operator $\Xi_{\boldsymbol{k}}$ in parallel. Also, if $\mathfrak{N}_{\boldsymbol{k},\mathrm{h}}(s_{\boldsymbol{i}})$ is an empty set, operator $\Xi_{\boldsymbol{k}}$ will perform $s_{\boldsymbol{i}} \leftarrow 0$. We similarly define operators $\Xi_{\boldsymbol{k},0}$ and $\Xi_{\boldsymbol{k},1}$ respectively as

$$s_{\boldsymbol{i}} \leftarrow \sum_{s_{\boldsymbol{j}} \in \mathfrak{N}_{\boldsymbol{k},\mathrm{h},0}(s_{\boldsymbol{i}})} s_{\boldsymbol{j}} \text{ and } s_{\boldsymbol{i}} \leftarrow \sum_{s_{\boldsymbol{j}} \in \mathfrak{N}_{\boldsymbol{k},\mathrm{h},1}(s_{\boldsymbol{i}})} s_{\boldsymbol{j}}.$$

An example is provided below to help clarifying these notations.

Example 3. *For UW $\boldsymbol{k} = 000$ of length $L = 3$, there are four possible states in $\mathfrak{S}_2 = \{s_{00}, s_{01}, s_{10}, s_{11}\}$. Because $\mathtt{A}_{\boldsymbol{k}}^{L-1} \underline{y}_{\boldsymbol{k}} = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}^\top$, the set of counting states is $\mathfrak{C}_{\boldsymbol{k}} = \{s_{01}, s_{11}\}$. Create the 0-edges and 1-edges in the digraph in Fig. 4.1. Table 4.3 then gives $\mathfrak{N}_{\boldsymbol{k},h}(s_{\boldsymbol{i}})$, $\mathfrak{N}_{\boldsymbol{k},t}(s_{\boldsymbol{i}})$, $\mathfrak{N}_{\boldsymbol{k},h,0}(s_{\boldsymbol{i}})$, $\mathfrak{N}_{\boldsymbol{k},h,1}(s_{\boldsymbol{i}})$, $\mathfrak{N}_{\boldsymbol{k},t,0}(s_{\boldsymbol{i}})$ and $\mathfrak{N}_{\boldsymbol{k},t,1}(s_{\boldsymbol{i}})$.*

Table 4.3: Various state sets for UW $\boldsymbol{k} = 000$

$s_{\boldsymbol{i}}$	s_{00}	s_{01}	s_{10}	s_{11}
$\mathfrak{N}_{\boldsymbol{k},\mathrm{h}}(s_{\boldsymbol{i}})$	$\{s_{10}\}$	$\{s_{00}, s_{10}\}$	$\{s_{01}, s_{11}\}$	$\{s_{01}, s_{11}\}$
$\mathfrak{N}_{\boldsymbol{k},\mathrm{t}}(s_{\boldsymbol{i}})$	$\{s_{01}\}$	$\{s_{10}, s_{11}\}$	$\{s_{00}, s_{01}\}$	$\{s_{10}, s_{11}\}$
$\mathfrak{N}_{\boldsymbol{k},\mathrm{h},0}(s_{\boldsymbol{i}})$	$\{s_{10}\}$	$\{\}$	$\{s_{01}, s_{11}\}$	$\{\}$
$\mathfrak{N}_{\boldsymbol{k},\mathrm{h},1}(s_{\boldsymbol{i}})$	$\{\}$	$\{s_{00}, s_{10}\}$	$\{\}$	$\{s_{01}, s_{11}\}$
$\mathfrak{N}_{\boldsymbol{k},\mathrm{t},0}(s_{\boldsymbol{i}})$	$\{\}$	$\{s_{10}\}$	$\{s_{00}\}$	$\{s_{10}\}$
$\mathfrak{N}_{\boldsymbol{k},\mathrm{t},1}(s_{\boldsymbol{i}})$	$\{s_{01}\}$	$\{s_{11}\}$	$\{s_{01}\}$	$\{s_{11}\}$

According to the first row in Table 4.3, applying operator $\Xi_{\boldsymbol{k}}$ to $\mathfrak{S}_2$ results in:

$$\begin{cases} s_{00} \leftarrow s_{10}; \\ s_{01} \leftarrow s_{00} + s_{10}; \\ s_{10} \leftarrow s_{01} + s_{11}; \\ s_{11} \leftarrow s_{01} + s_{11}. \end{cases}$$

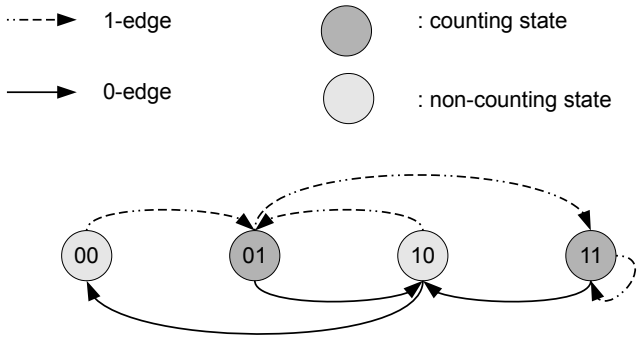

Figure 4.1: Digraph G_{000} for UW $\boldsymbol{k} = 000$

Likewise, applying operators $\Xi_{\boldsymbol{k},0}$ *and* $\Xi_{\boldsymbol{k},1}$ *to* $\mathfrak{S}_2$ *respectively yields:*

$$\begin{cases} s_{00} \leftarrow s_{10}; \\ s_{01} \leftarrow 0; \\ s_{10} \leftarrow s_{01} + s_{11}; \\ s_{11} \leftarrow 0, \end{cases} \quad \textit{and} \quad \begin{cases} s_{00} \leftarrow 0; \\ s_{01} \leftarrow s_{00} + s_{10}; \\ s_{10} \leftarrow 0; \\ s_{11} \leftarrow s_{01} + s_{11}. \end{cases}$$

So, if $s_{00} = 2$, $s_{01} = 4$, $s_{10} = 1$ *and* $s_{11} = 6$ *before the update, then* $\Xi_{\boldsymbol{k}}(\mathfrak{S}_2) = \{s_{00} = 1, s_{01} = 3, s_{10} = 10, s_{00} = 10\}$, $\Xi_{\boldsymbol{k},0}(\mathfrak{S}_2) = \{s_{00} = 1, s_{01} = 0, s_{10} = 10, s_{00} = 0\}$ *and* $\Xi_{\boldsymbol{k},1}(\mathfrak{S}_2) = \{s_{00} = 0, s_{01} = 3, s_{10} = 0, s_{00} = 10\}$.

We next demonstrate through this example how to compute $c_{\boldsymbol{k},i}$ *and* $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)|$ *with the help of the state machine.*

For the computation of

$$c_{\boldsymbol{k},i} = \underline{x}_{\boldsymbol{k}}^{\top} \mathtt{A}_{\boldsymbol{k}}^{i+L-1} \underline{y}_{\boldsymbol{k}} = \underline{x}_{\boldsymbol{k}}^{\top} \mathtt{A}_{\boldsymbol{k}}^{i} \left(\mathtt{A}_{\boldsymbol{k}}^{L-1} \underline{y}_{\boldsymbol{k}} \right),$$

we initialize $s_{k_2 \ldots k_L} = 1$ *and assign* $s_{\boldsymbol{i}} = 0$ *for all remaining* $\boldsymbol{i} \neq k_2^L$. *Note that the above*

initial values associated with states exactly copy the component values in the initial vector $\underline{x}_{\boldsymbol{k}}^{\top}$. *Recall that the* $(i+1)$*th component of* $\left(\mathtt{A}_{\boldsymbol{k}}^{L-1}\underline{y}_{\boldsymbol{k}}\right)$ *is* 1 *whenever* $s_{\boldsymbol{i}} = s_{i_1\cdots i_{L-1}}$ *is a counting state; so, only the computation of* $\underline{x}_{\boldsymbol{k}}^{\top}\mathtt{A}_{\boldsymbol{k}}^{i}$ *needs to be taken care in the sequel. Updating the value associated with each state by operator* $\Xi_{\boldsymbol{k}}$ *once yields* $s_{00} = 0$, $s_{01} = 1$, $s_{10} = 0$ *and* $s_{11} = 0$, *which corresponds exactly to* $\underline{x}_{\boldsymbol{k}}^{\top}\mathtt{A}_{\boldsymbol{k}}$. *The second update makes* $s_{00} = 0$, $s_{01} = 0$, $s_{10} = 1$ *and* $s_{11} = 1$, *which reflects the vector component values in* $\underline{x}_{\boldsymbol{k}}^{\top}\mathtt{A}_{\boldsymbol{k}}^{2}$. *After applying operator* $\Xi_{\boldsymbol{k}}$ *four times, we obtain* $s_{00} = 1$, $s_{01} = 2$, $s_{10} = 2$ *and* $s_{11} = 2$, *which corresponds to* $\underline{x}_{\boldsymbol{k}}^{\top}\mathtt{A}_{\boldsymbol{k}}^{4}$. *Hence, the sum of the values associated with all counting states gives*

$$c_{000,4} = \underline{x}_{\boldsymbol{k}}^{\top}\mathtt{A}_{\boldsymbol{k}}^{4}\left(\mathtt{A}_{\boldsymbol{k}}^{2}\underline{y}_{\boldsymbol{k}}\right) = s_{01} + s_{11} = 4.$$

On the other hand, to compute

$$\begin{aligned}|C_{\boldsymbol{k}}(\boldsymbol{d}0,n)| &= \underline{x}_{\boldsymbol{k}}^{\top}\mathtt{D}_i\mathtt{A}_{\boldsymbol{k},0}\mathtt{A}_{\boldsymbol{k}}^{n+L-i}\underline{y}_{\boldsymbol{k}}\\ &= \left(\underline{x}_{\boldsymbol{k}}^{\top}\mathtt{D}_i\right)\mathtt{A}_{\boldsymbol{k},0}\mathtt{A}_{\boldsymbol{k}}^{n-i+1}\left(\mathtt{A}_{\boldsymbol{k}}^{L-1}\underline{y}_{\boldsymbol{k}}\right),\end{aligned}$$

for a given prefix $\boldsymbol{d} = d_1\cdots d_i$, *we let* $\boldsymbol{u} = u_1^{L-1}$ *be the last* $(L-1)$ *tuples of* $k_2\cdots k_L\boldsymbol{d}$, *and initialize the values associated with all states to be zero except* $s_{u_1\ldots u_{L-1}} = 1$. *Note that the assigned initial values correspond to the component values in row vector* $\underline{x}_{\boldsymbol{k}}^{\top}\mathtt{D}_i$. *Apply the operator* $\Xi_{\boldsymbol{k},0}$ *to* $\mathfrak{S}_2$ *once, followed by updating* $\mathfrak{S}_2$ $(n-i+1)$ *times via operator* $\Xi_{\boldsymbol{k}}$. *Then, the sum of all counting states equals* $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0,n)|$.

For completeness, we incorporate the above complexity reduction computations into the encoding and decoding procedures and summarize them in Algorithms 7 and 8.

4.7 Extension to N-ary UDOOCs

All the previous discussions focus on binary UDOOCs. In certain applications, an extension from binary UDOOCs to N-ary ones may be of practical use. This section is then added to

Algorithm 7 Encoding of UDOOC with General $\boldsymbol{k}$

Input: Index m for message u_m

Output: Codeword $\phi_{\boldsymbol{k}}(u_m) = c_1 \dots c_n$

1: Create state set $\mathfrak{S}_{L-1}$ and find the set of counting states $\mathfrak{C}_{\boldsymbol{k}}$ by computing $\mathtt{A}_{\boldsymbol{k}}^{L-1}\underline{y}_{\boldsymbol{k}}$
2: Initialize $s_{\boldsymbol{i}} = 0$ for all $\boldsymbol{i}$ except $s_{k_2^L} = 1$
3: Repeat $\mathfrak{S}_{\boldsymbol{k}} \leftarrow \Xi(\mathfrak{S}_{\boldsymbol{k}})$ and $c_{\boldsymbol{k},i} = \sum_{s_{\boldsymbol{j}} \in \mathfrak{C}_{\boldsymbol{k}}} s_{\boldsymbol{j}}$ to determine the smallest n such that $F_{\boldsymbol{k},n} \geq m$. If $n = 0$, then $\phi_{\boldsymbol{k}}(u_m) =$ null and stop the algorithm.
4: $\rho_0 \leftarrow m - F_{\boldsymbol{k},n-1}$ and $\boldsymbol{d} = \{\}$
5: **for** $i = 1$ to n **do**
6: Let $\boldsymbol{u}$ be the last $(L-1)$ tuples of $k_2^L\boldsymbol{d}$
7: Set $s_{\boldsymbol{i}} = 0$ for all $\boldsymbol{i}$ except $s_{\boldsymbol{u}} = 1$
8: $\mathfrak{S}_{L-1} \leftarrow \Xi_{\boldsymbol{k},0}(\mathfrak{S}_{L-1})$
9: Update $\mathfrak{S}_{L-1}$ $(n-i)$ times by operator $\Xi_{\boldsymbol{k}}$
10: $temp \leftarrow \sum_{s_{\boldsymbol{j}} \in \mathfrak{C}_{\boldsymbol{k}}} s_{\boldsymbol{j}}$
11: **if** $\rho_{i-1} \leq temp$ **then**
12: $c_i \leftarrow 0$, $\rho_i \leftarrow \rho_{i-1}$ and $\boldsymbol{d} \leftarrow \boldsymbol{d}0$
13: **else**
14: $c_i \leftarrow 1$, $\rho_i \leftarrow \rho_{i-1} - temp$ and $\boldsymbol{d} \leftarrow \boldsymbol{d}1$
15: **end if**
16: **end for**

Algorithm 8 Decoding of UDOOC with General $\boldsymbol{k}$

Input: Codeword $\boldsymbol{c} = c_1 \dots c_n$

Output: Index m for message $u_m = \psi_{\boldsymbol{k}}(\boldsymbol{c})$

1: Create state set $\mathfrak{S}_{L-1}$ and find the set of counting states $\mathfrak{C}_{\boldsymbol{k}}$ by computing $\mathtt{A}_{\boldsymbol{k}}^{L-1}\underline{y}_{\boldsymbol{k}}$
2: Initialize $s_{\boldsymbol{i}} = 0$ for all $\boldsymbol{i}$ except $s_{k_2^L} = 1$
3: **for** $i = 1$ to n **do**
4: $\mathfrak{S}_{L-1} \leftarrow \Xi_{\boldsymbol{k}}(\mathfrak{S}_{L-1})$ and $c_{\boldsymbol{k},i} = \sum_{s_{\boldsymbol{j}} \in \mathfrak{C}_{\boldsymbol{k}}} s_{\boldsymbol{j}}$
5: **end for**
6: Initialize $m \leftarrow F_{\boldsymbol{k},n-1} + 1$ and $\boldsymbol{d} = \{\}$
7: **for** $i = 1$ to n **do**
8: **if** $c_i = 1$ **then**
9: Set $\boldsymbol{u}$ be the last $(L-1)$ tuples of $k_2^L\boldsymbol{d}$
10: Set $s_{\boldsymbol{i}} = 0$ for all $\boldsymbol{i}$ except $s_{\boldsymbol{u}} = 1$
11: $\mathfrak{S}_{L-1} \leftarrow \Xi_{\boldsymbol{k},0}(\mathfrak{S}_{L-1})$
12: Update $\mathfrak{S}_{L-1}$ $(n-i)$ times by operator $\Xi_{\boldsymbol{k}}$
13: $m \leftarrow m + \sum_{s_{\boldsymbol{j}} \in \mathfrak{C}_{\boldsymbol{k}}} s_{\boldsymbol{j}}$
14: **end if**
15: $\boldsymbol{d} \leftarrow \boldsymbol{d}c_i$
16: **end for**

this end.

Let $\mathcal{A} = \{a_1, \ldots, a_N\}$ be the code alphabet for N-ary UDOOCs. For UW $\boldsymbol{k} \in \mathcal{A}^L$ of length L, we say $\mathcal{C}_{\boldsymbol{k}}(n)$ is a UDOOC of length n associated with $\boldsymbol{k}$ if it contains all N-ary tuples $\boldsymbol{b} \in \mathcal{A}^n$ such that $\boldsymbol{k}$ is not an internal subword of $\boldsymbol{kbk}$.

Following the discussion in Chapter 2, we can construct the digraph $G_{\boldsymbol{k}} = (V, E_{\boldsymbol{k}})$ with adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$, where the vertex set is extended to $V = \mathcal{A}^{L-1}$ and the edge set is defined as

$$E_{\boldsymbol{k}} = \left\{(\boldsymbol{i}, \boldsymbol{j}) \in V^2 : i_2^{L-1} = j_1^{L-1}, i_1\boldsymbol{j} \neq \boldsymbol{k}\right\}.$$

The adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$ for N-ary UDOOC is of the size $N^{L-1} \times N^{L-1}$, and

$$(\mathtt{A}_{\boldsymbol{k}})_{i+1,j+1} = \begin{cases} 1, & \text{if } (\boldsymbol{i}, \boldsymbol{j}) \in E_{\boldsymbol{k}} \\ 0, & \text{otherwise} \end{cases}$$

where

$$i = i_1 \cdot N^{L-2} + i_2 \cdot N^{L-3} + \cdots + i_{L-1}$$

and

$$j = j_1 \cdot N^{L-2} + j_2 \cdot N^{L-3} + \cdots + j_{L-1}$$

are the integers corresponding to N-ary representations of $\boldsymbol{i} = i_1 \ldots i_{L-1}$ and $\boldsymbol{j} = j_1 \ldots j_{L-1}$, respectively. We can likewisely define the initial vector $\underline{x}_{\boldsymbol{k}}$ and the ending vector $\underline{y}_{\boldsymbol{k}}$ for N-ary code alphabet as similar to Chapter 3.1. It then follows that

$$c_{\boldsymbol{k},n} \;=\; |\mathcal{C}_{\boldsymbol{k}}(n)| \;=\; \underline{x}_{\boldsymbol{k}}^{\top} \mathtt{A}_{\boldsymbol{k}}^{n+L-1} \underline{y}_{\boldsymbol{k}}. \tag{4.26}$$

For the encoding and decoding of N-ary UDOOCs, we also define $E_{\boldsymbol{k},a} = \{(\boldsymbol{i}, \boldsymbol{j}) \in E_{\boldsymbol{k}} : j_{L-1} = a\}$ and subsequently $\mathtt{A}_{\boldsymbol{k},a}$ for each $a \in \mathcal{A}$. By retaining $F_{\boldsymbol{k},n} = \sum_{t=0}^{n} c_{\boldsymbol{k},t}$, the encoding and decoding algorithms for N-ary UDOOCs can be described in Algorithms 9 and 10, respectively.

We end this section by pointing out that it can be seen that the extension of binary UDOOCs to N-ary ones turns straightforward because of the introduction of the digraph and

Algorithm 9 Encoding of N-ary UDOOC with General $\boldsymbol{k}$

Input: Index m for message u_m

Output: Codeword $\phi_{\boldsymbol{k}}(u_m) = c_1 \dots c_n$

1: Compute $c_{\boldsymbol{k},0}, c_{\boldsymbol{k},1}, \dots$ using (4.26) to determine the smallest n such that $F_{\boldsymbol{k},n} \geq m$. If $n = 0$, then $\phi_{\boldsymbol{k}}(u_m) =$ null and stop the algorithm.

2: $\rho_0 \leftarrow m - F_{\boldsymbol{k},n-1}$ and $\mathtt{D}_0 \leftarrow \mathtt{I}$

3: **for** $i = 1$ to n **do**

4: Find the smallest t among $1 \leq t \leq N$ such that

$$\rho_{i-1} \leq \underline{x}_{\boldsymbol{k}}^{\top} \mathtt{D}_{i-1} \left(\sum_{j=1}^{t} \mathtt{A}_{\boldsymbol{k},a_j} \right) \mathtt{A}_{\boldsymbol{k}}^{(n-L+1)-i} \underline{y}_{\boldsymbol{k}}$$

5: $c_i \leftarrow a_t$

6: $\mathtt{D}_i \leftarrow \mathtt{D}_{i-1} \mathtt{A}_{\boldsymbol{k},a_t}$

7: $\rho_i \leftarrow \rho_{i-1} - \underline{x}_{\boldsymbol{k}}^{\top} \mathtt{D}_{i-1} \left(\sum_{j=1}^{t-1} \mathtt{A}_{\boldsymbol{k},a_j} \right) \mathtt{A}_{\boldsymbol{k}}^{(n+L-1)-i} \underline{y}_{\boldsymbol{k}}$

8: **end for**

Algorithm 10 Decoding of N-ary UDOOC with General $\boldsymbol{k}$

Input: Codeword $\boldsymbol{c} = c_1 \dots c_n$

Output: Index m for message $u_m = \psi_{\boldsymbol{k}}(\boldsymbol{c})$

1: Compute $c_{\boldsymbol{k},0}, c_{\boldsymbol{k},1}, \dots, c_{\boldsymbol{k},n}$ using (4.26)

2: Initialize $m \leftarrow F_{\boldsymbol{k},n-1} + 1$ and $\mathtt{D} \leftarrow \mathtt{I}$

3: **for** $i = 1$ to n **do**

4: Find the index t such that $c_i = a_t$

5: $m \leftarrow m + \underline{x}_{\boldsymbol{k}}^{\top} \mathtt{D}_{i-1} \sum_{j=1}^{t-1} \mathtt{A}_{\boldsymbol{k},a_j} \mathtt{A}_{\boldsymbol{k}}^{(n+L-1)-i} \underline{y}_{\boldsymbol{k}}$

6: $\mathtt{D}_i \leftarrow \mathtt{D}_{i-1} \mathtt{A}_{\boldsymbol{k},c_i}$

7: **end for**

its associated adjacency matrix. This confirms further the powerfulness of such technique. Nevertheless, Step 4 in Algorithm 9 and also Step 4 in Algorithm 10 imply the non-triviality of such an extension. We thus add this section for completeness.

Chapter 5

Practice and performance of UDOOCs

In Figs. 5.1-5.4, we compare the numbers of length-n codewords for all UWs of lengths $L = 2$, 3, 4 and 5, respectively. These numbers are plotted in logarithmic scale and are normalized against the number of length-n codewords for the all-zero UW $\boldsymbol{k} = 0 \ldots 00$ to facilitate their comparison. By the equivalence defined in Definition 2, only one UW in each equivalence class needs to be illustrated. We then made the following observations.

1. The logarithmic ratio $\log_2(c_{\boldsymbol{k},n}/c_{\boldsymbol{a},n})$, where $\boldsymbol{a} = 0 \ldots 00$, exhibits some transient fluctuation for $n \leq L$ but becomes a steady straight line of negative slope after $n > L$. This hints that $c_{\boldsymbol{k},n}$ has a steady exponential growth when n is beyond L.

2. The number $c_{\boldsymbol{b},n}$, where $\boldsymbol{b} = 00 \ldots 01$, always gives the largest value among all UWs when n is small. However, this number has an apparent trend to be overtaken by those of other UWs as n grows and will be eventually smaller than the number of length-n codewords for the all-zero UW. The result asymptotically matches what has been proven in Theorem 2.

3. As a contrary, the number $c_{\boldsymbol{a},n}$ for the all-zero UW $\boldsymbol{k} = \boldsymbol{a} = 0 \ldots 00$ is the smallest among all $c_{\boldsymbol{k},n}$ for UWs of the same length when n is small. Although Theorem 2 indicates that this number will be eventually the largest, these figures show that such

would happen only when n is very large, or only for the compression of sources with impractically large size.

4. As a result of the two previous observations, UW $\boldsymbol{k} = 00\ldots01$ perhaps remains a better choice in the compression of sources with practical number of source letters even though it is asymptotically the worst. We will confirm this inference by the later practice of UDOOCs on a real text source from the book *Alice's Adventure in Wonderland*.

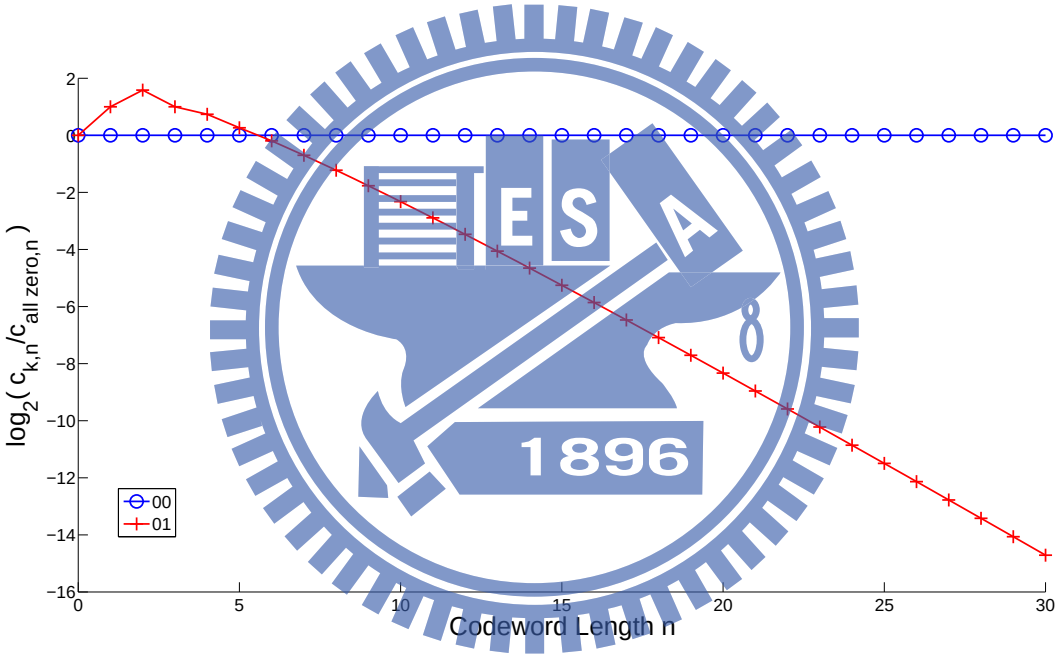

Figure 5.1: Normalized numbers of length-n codewords for UWs of length $L = 2$

We next investigate the compression rates of UDOOCs and compare them with those of the Huffman and Lempel-Ziv (LZ77 and LZ78) codes. In this experiment, the standard Huffman code [6] in the communication toolbox of Matlab is used instead of the adaptive Huffman code. The LZ77 executable is obtained from the basic compression library in [33], while the LZ78 is self-implemented using C++ programing language. As a convention,

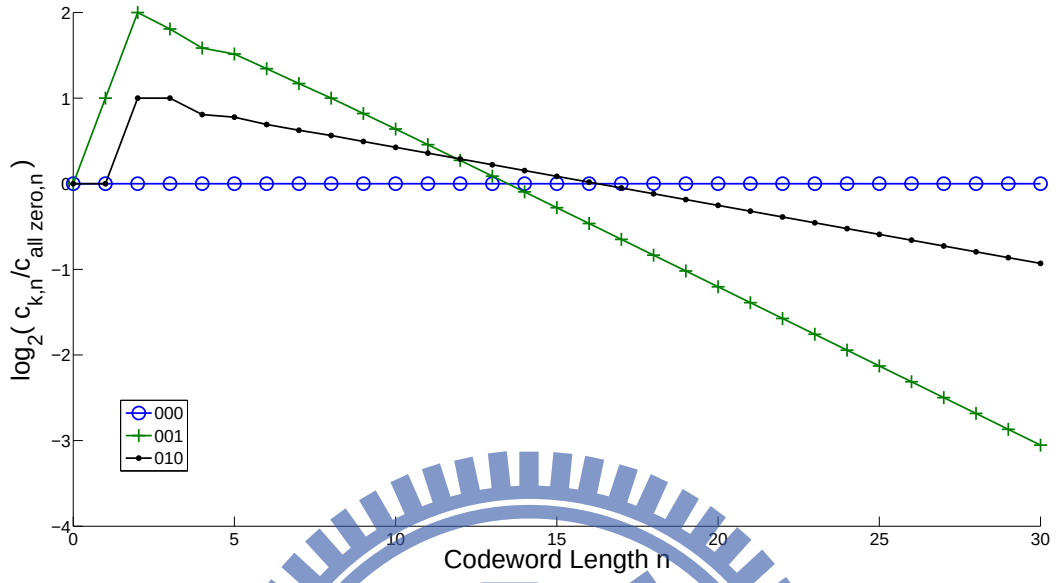

Figure 5.2: Normalized numbers of length-n codewords for UWs of length $L = 3$

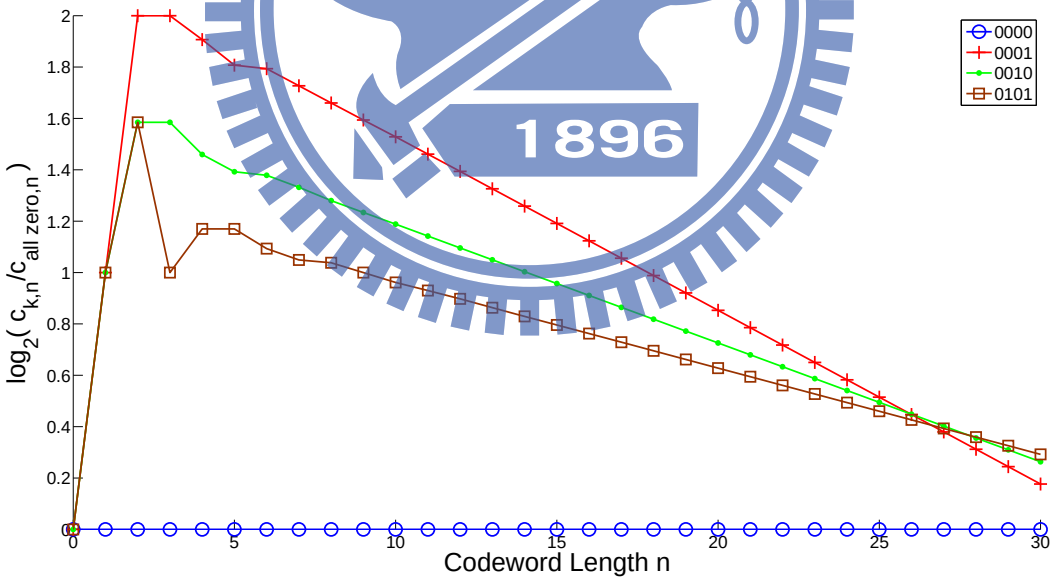

Figure 5.3: Normalized numbers of length-n codewords for UWs of length $L = 4$

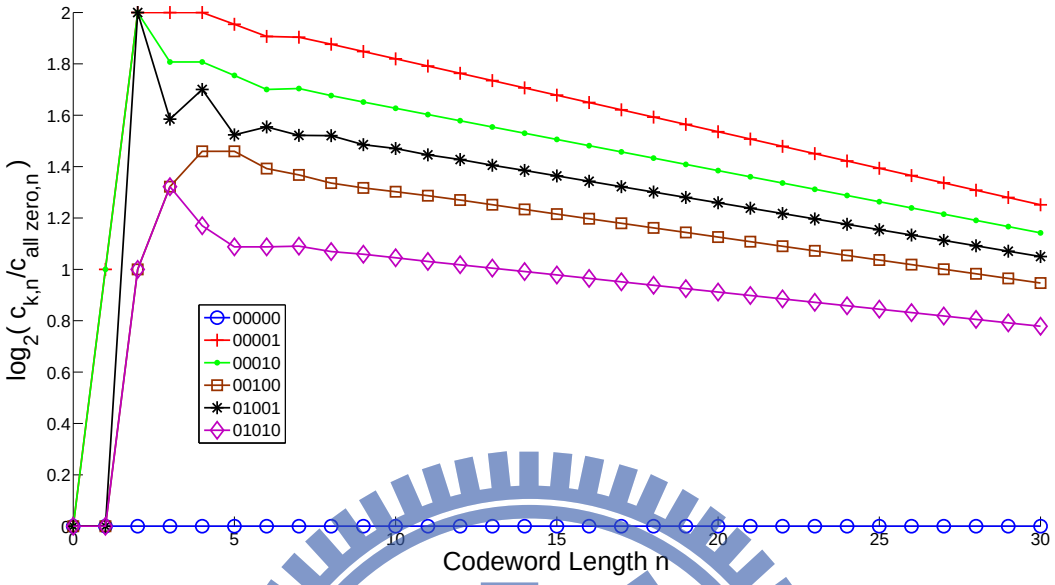

Figure 5.4: Normalized numbers of length-n codewords for UWs of length $L = 5$

the data is binary ASCII encoded before it is fed into the two Lempel Ziv compression algorithms. The sliding window for the LZ77 is set as 10000 bits, and the tree-structured LZ78 is implemented without any windowing.

Three different English text sources are used, in which uppercase and lowercase English letters are treated as one symbol, and specifically for the third English text source, any symbols other than the 26 English alphabet are regarded as a "space." The first English text source is distributed uniformly over the 26 symbols. The second English text source is assumed independent and identically distributed (i.i.d.) with marginal statistics from [35]. The third one is a true English text source from *Alice's Adventure in Wonderland*. In addition, the effect of grouping t symbols as a grouped source for compression is studied, which will be termed t-grouper in the below remarks. The results are summarized in Tables 5.1 and 5.2, in which the average codeword length of UDOOCs has already taken into account the length of UWs. We remark on the experimental results as follows.

1. First of all, it can be observed from Table 5.1 that the length-2 UW $\boldsymbol{k} = 01$ gives a good per-letter average codeword length only when $t = 1$. When the size of source alphabet increases by grouping $t = 2$ or $t = 3$ letters into one symbol for UDOOC compression, the per-letter average codeword length dramatically grows. Note that $\boldsymbol{k} = 01$ is the only UW, whose number of length-n codewords has a linear growth with respect to n, and is equal to $c_{01,n} = n + 1$. Since the size of source alphabet increases exponentially in t when t-grouper is employed, the resulting per-letter average codeword length also increases exponentially as t grows; therefore, when UW $= 01$, t-grouper will result in an extremely poor performance for moderately large t.

2. By independently generating 10^6 letters according to the statistics in [35] for compression, we record the per-letter average codeword in the second row of Table 5.1. As expected, the Huffman coding scheme gives the smallest per-letter average codeword length, i.e., 4.253 bits per letter, when 3-grouper is used. The gap of per-letter average codeword lengths between the 3-grouper Huffman and the 3-grouper UDOOC however can be made as small as $4.795 - 5.253 = 0.542$ bits per source letter if UW $= 0001$. This is in contrast to the gap of 1.007 bits when uniform independent English text source is the one to be compressed (cf. the first row in Table 5.1). We would like to point out that the error propagation of UDOOCs is limited firmly by at most two codewords, while that of the Huffman code may be statistically beyond this range. In comparison with the LZ77 and LZ78, the UDOOC clearly performs better in compression rate for usual independent English text source.

3. When the compression of a source with memory from the book titled *Alice's Adventures in Wonderland* [34] is concerned, the third row in Table 5.1 shows that the gap of per-letter average codeword lengths between the optimal 3-grouper Huffman and the 3-grouper UDOOC with UW $= 0001$ is narrowed down to 0.305 bits per letter. The 3-

grouper UDOOC with the all-zero UW also performs well when what to be compressed is a source with memory from *Alice's Adventures in Wonderland*. Note that part of the per-letter average codeword length of UDOOCs is contributed by the UW, i.e., L/t; hence, in a sense, a larger t and a smaller L are favored (except for $L = 2$), and the best performance is usually parameterized by $t = 3$ and $L = 4$ in Table 5.1.

4. For the third English text source, the LZ77 performs better than all of the 1-grouper UDOOC compression schemes but one. We then compare the running time of both algorithms in Table 5.2 and observe that when the LZ77 has similar running time to the 1-grouper UDOOC scheme by reducing its window size, its performance degrades down to 5.234 bits per letter, which is larger than that of the 1-grouper UDOOC. Note that we only compare their running time in encoding in Table 5.2 as the decoding efficiency of UDOOCs is seemingly better than that of the LZ77. Considering also the low memory consumption of UDOOCs when a specific UW is pre-given in addition to its simplicity in implementation, the UDOOC can be regarded as a cost-economic compression scheme for practical applications.

Table 5.1: Average codeword lengths in bits per source symbol for the compression of three different sources. The best one among 1-grouper, 2-grouper and 3-grouper of the same compression scheme is boldfaced.

Type	**Entropy**			**LZ77**	**LZ78**	**Huffman**				**UW** $= 00\cdots0$			**UW** $= 00\cdots01$		
	$t=1$	$t=2$	$t=3$			$t=1$	$t=2$	$t=3$		$t=1$	$t=2$	$t=3$	$t=1$	$t=2$	$t=3$
Uniform Independent English Letters	4.700	4.700	4.700	7.992	7.178	4.768	4.738	**4.702**	$L=2$	6.961	6.820	**6.790**	**5.846**	12.76	41.99
									$L=4$	8.576	6.831	**6.213**	7.000	5.899	**5.709**
									$L=6$	10.58	7.748	**6.746**	9.000	6.768	**6.104**
	$t=1$	$t=2$	$t=3$			$t=1$	$t=2$	$t=3$		$t=1$	$t=2$	$t=3$	$t=1$	$t=2$	$t=3$
Usual Independent English Letters	4.246	4.246	4.246	7.925	6.626	4.274	4.261	**4.253**	$L=2$	5.597	**5.550**	5.637	**4.557**	7.771	20.907
									$L=4$	7.411	5.872	**5.351**	6.185	4.970	**4.795**
									$L=6$	9.411	6.818	**5.924**	8.185	5.882	**5.274**
	$t=1$	$t=2$	$t=3$			$t=1$	$t=2$	$t=3$		$t=1$	$t=2$	$t=3$	$t=1$	$t=2$	$t=3$
Alice's Adventures in Wonderland	3.914	3.570	3.215	4.661	6.028	3.940	3.585	**3.226**	$L=2$	4.887	4.340	**3.958**	**4.068**	4.975	7.573
									$L=4$	6.757	4.920	**4.089**	5.774	4.133	**3.531**
									$L=6$	8.757	5.890	**4.709**	7.774	5.089	**4.115**

Table 5.2: Average codeword lengths in bits per source symbol and running time in seconds for the UDOOC encoding and the LZ77 encoding on *Alice's Adventures in Wonderland*. The programs are implemented using C++, and are executed in a Microsoft Windows-based desktop with intel-Core7 2.4G CUP and 8G memory.

	Type	Average Codewrod length	Running Time
UDOOC	UW $\boldsymbol{k} = 00$	4.887	0.0162 sec
	UW $\boldsymbol{k} = 01$	4.068	0.0158 sec
LZ77	Window Size = 10^4 bits	4.661	0.0328 sec
	Window Size = 3000 bits	5.234	0.01607 sec

Chapter 6

Conclusion

In this thesis, we have provided a general construction of UDOOCs with arbitrary UW. Combinatorial properties of UDOOCs are subsequently investigated. Based on our studies, the appropriate UW for the UDOOC compression of a given source can be chosen. Various encoding and decoding algorithms for general UDOOCs, as well as their efficient counterparts for specific UWs like $\boldsymbol{k} = 00\ldots0$, $00\ldots01$, are also provided. Performances of UDOOCs are then compared with the Huffman and Lemple-Ziv codes. From our experimental results, UDOOCs, in spite of its simpleness in idea, can be a good practical candidate for lossless data compression when a cost-economic solution is required.

Appendix A

Proof of Theorem 1

In this chapter, we will provide a proof for Theorem 1. It is a technique similar to that in [22].

Let $\mathcal{U}^* := \bigcup_{n\geq 0} \mathbb{F}^n$ be the set of all binary sequences. Recall that $s_{\boldsymbol{k},n} = |\mathcal{S}_{\boldsymbol{k}}(n)|$ is the size of set $\mathcal{S}_{\boldsymbol{k}}(n)$ defined in (3.6). For a word $\boldsymbol{w} = w_1 \dots w_n \in \mathcal{U}^*$ of length n, let $\mathcal{F}_{\boldsymbol{k}}(\boldsymbol{w})$ be the set of index pairs indicating the places that $\boldsymbol{w}$ contains $\boldsymbol{k}$ as a subword, i.e.,

$$\mathcal{F}_{\boldsymbol{k}}(\boldsymbol{w}) \;=\; \left\{(i,j) \;:\; \boldsymbol{k} = w_i^j\right\}.$$

Further denote by $\ell(\boldsymbol{w})$ the length of word $\boldsymbol{w}$. Then

$$\begin{aligned} f(z) &= \sum_{n\geq 0} s_{\boldsymbol{k},n} z^n \\ &\overset{\text{(a)}}{=} \sum_{\boldsymbol{w}\in\mathcal{U}^*} z^{\ell(\boldsymbol{w})} 0^{|\mathcal{F}_{\boldsymbol{k}}(\boldsymbol{w})|} \\ &= \sum_{\boldsymbol{w}\in\mathcal{U}^*} z^{\ell(\boldsymbol{w})} \prod_{a\in\mathcal{F}_{\boldsymbol{k}}(\boldsymbol{w})} (1+(-1)) \\ &\overset{\text{(b)}}{=} \sum_{\boldsymbol{w}\in\mathcal{U}^*} z^{\ell(\boldsymbol{w})} \sum_{A\subseteq\mathcal{F}_{\boldsymbol{k}}(\boldsymbol{w})} (-1)^{|A|} \end{aligned} \tag{A.1}$$

where in (a) we have adopted the convention of $0^0 = 1$ and (b) follows from the inclusion-exclusion principle. In light of (A.1), we will regard the pair $(\boldsymbol{w}, A)$ with $A \subseteq \mathcal{F}_{\boldsymbol{k}}(\boldsymbol{w})$ as a *marked word*. The set of all marked words is thus defined as

$$\mathcal{M}_{\boldsymbol{k}} \;:=\; \left\{(\boldsymbol{w}, A) \;:\; \boldsymbol{w}\in\mathcal{U}^* \text{ and } A\subseteq\mathcal{F}_{\boldsymbol{k}}(\boldsymbol{w})\right\}.$$

Using the following weight function for elements in $\mathcal{M}_{\boldsymbol{k}}$

$$\pi(\boldsymbol{w}, A) \;:=\; z^{\ell(\boldsymbol{w})}(-1)^{|A|} \tag{A.2}$$

we can rewrite (A.1) as

$$f(z) \;=\; \sum_{(\boldsymbol{w},A)\in\mathcal{M}_{\boldsymbol{k}}} \pi(\boldsymbol{w}, A). \tag{A.3}$$

To determine $f(z)$, below we introduce the concept of a *cluster*.

Definition 6 (Cluster)**.** *We say the marked word* $(\boldsymbol{w}, A)$ *is a* cluster *if, and only if,*

$$\bigcup_{(i_t,j_t)\in A} [i_t, j_t] = [1, \ell(\boldsymbol{w})]$$

where by $[a, b]$ *we mean the closed interval* $\{x \in \mathbb{R} : a \le x \le b\}$ *on the real line. The set of all clusters is thus*

$$\mathcal{T}_{\boldsymbol{k}} \;=\; \{(\boldsymbol{w}, A) \in \mathcal{M}_{\boldsymbol{k}} \;:\; (\boldsymbol{w}, A) \text{ is a cluster}\}\,.$$

Definition 7 (Concatenation of sets of marked words)**.** *For any two sets of marked words* $\mathcal{A}_{\boldsymbol{k}}$ *and* $\mathcal{B}_{\boldsymbol{k}}$*, we define the concatenation of* $\mathcal{A}_{\boldsymbol{k}}$ *and* $\mathcal{B}_{\boldsymbol{k}}$ *as*

$$\mathcal{A}_{\boldsymbol{k}} \vee \mathcal{B}_{\boldsymbol{k}} := \{(\boldsymbol{ab}, A \cup \mathfrak{I}(B, \ell(\boldsymbol{a})) : (\boldsymbol{a}, A) \in \mathcal{A}_{\boldsymbol{k}}, (\boldsymbol{b}, B) \in \mathcal{B}_{\boldsymbol{k}}\}$$

where by $\boldsymbol{ab}$ *we meant the usual concatenation of strings* $\boldsymbol{a}$ *and* $\boldsymbol{b}$*, and where the function* $\mathfrak{I}(B, \ell(\boldsymbol{a}))$ *is*

$$\mathfrak{I}(B, \ell(\boldsymbol{a})) \;:=\; \{(i_t + \ell(\boldsymbol{a}), j_t + \ell(\boldsymbol{a})) : (i_t, j_t) \in B\}\,.$$

Having defined the concatenation operation $\vee$ for sets of marked words, we next claim the following decomposition for the set $\mathcal{M}_{\boldsymbol{k}}$

$$\mathcal{M}_{\boldsymbol{k}} \;=\; \{(\text{null}, \emptyset)\} \cup (\mathcal{M}_{\boldsymbol{k}} \vee \mathcal{F}) \cup (\mathcal{M}_{\boldsymbol{k}} \vee \mathcal{T}_{\boldsymbol{k}})\,. \tag{A.4}$$

where $\mathcal{F} := \{(b, \emptyset) : b \in \mathbb{F}\}$.

To show (A.4), for any $(\boldsymbol{w}, A) \in \mathcal{M}_{\boldsymbol{k}}$ we distinguish the following three disjoint cases:

1. If $\ell(\boldsymbol{w}) = 0$, it is obvious that $\boldsymbol{w}$ is a null word and $A = \emptyset$ from the definition of $\mathcal{F}_{\boldsymbol{k}}(\boldsymbol{w})$.

2. For $\ell(\boldsymbol{w}) \geq 1$, appending an arbitrary binary word to $\boldsymbol{w}$ results in another marked word $(\boldsymbol{w}b, A)$, which cannot be a cluster since

$$\bigcup_{(i_t,j_t)\in A} [i_t, j_t] \subset [1, \ell(\boldsymbol{w}) + 1].$$

Conversely, take any marked word $(\boldsymbol{w}, A)$ from $\mathcal{M}_{\boldsymbol{k}}$ with $\ell(\boldsymbol{w}) = n$. If $j_t < \ell(\boldsymbol{w}) = n$ for all $(i_t, j_t) \in A$, then we can delete the last bit from $\boldsymbol{w}$, and the resulting pair (w_1^{n-1}, A) is still a marked word. Summarizing the above gives the following equalities between two sets of marked words

$$\begin{aligned} &\{(\boldsymbol{w}, A) \in \mathcal{M}_{\boldsymbol{k}} : j_t < \ell(\boldsymbol{w}) \text{ for all } (i_t, j_t) \in A)\} \\ &= \{(\boldsymbol{w}b, A) : (\boldsymbol{w}, A) \in \mathcal{M}_{\boldsymbol{k}}, b \in \mathbb{F}\} \\ &= \mathcal{M}_{\boldsymbol{k}} \vee \mathcal{F} \end{aligned} \tag{A.5}$$

where the last equality follows from the definition of concatenation operation $\vee$.

3. The last case concerns the situation when $(\boldsymbol{w}, A)$ satisfies $\ell(\boldsymbol{w}) = n \geq 1$, $A = \{(i_1, j_1), \ldots, (i_m, j_m)\}$ and $i_1 < \cdots < i_m < j_m = n$. In other words, this is the case when $\max\{j_t : (i_t, j_t) \in A\} = \ell(\boldsymbol{w})$, which is disjoint from the second case. For this, let u be the smallest index such that $[i_{u+t}, j_{u+t}] \cap [i_{u+t+1}, j_{u+t+1}] \neq \emptyset$ for all $t = 0, 1, \ldots, m - u + 1$. Then obviously we have the following de-concatenation of $(\boldsymbol{w}, A)$

$$\begin{aligned} (\boldsymbol{w}, A) = &(w_1^{i_u - 1}, \{(i_t, j_t) : t = 1, \ldots, u - 1\}) \vee \\ &(w_{i_u}^{n}, \{(i_t - i_u + 1, j_t - i_u + 1) : t = u, \ldots, m\}). \end{aligned}$$

Clearly, the first marked word $(w_1^{i_u - 1}, \{(i_t, j_t) : t = 1, \ldots, u - 1\}) \in \mathcal{M}_{\boldsymbol{k}}$. The second marked word $(w_{i_u}^{n}, \{(i_t - i_u + 1, j_t - i_u + 1) : t = u, \ldots, m\})$ is a cluster since

$$\bigcup_{t=u}^{m} [i_t - i_u + 1, j_t - i_u + 1] = [1, n - i_u + 1]$$

by the choice of u. Hence we arrive at the following equality between two sets of marked words

$$\begin{aligned} &\{(\boldsymbol{w}, A) \in \mathcal{M}_{\boldsymbol{k}} \ : \ \max\{j_t : (i_t, j_t) \in A\} = \ell(\boldsymbol{w})\} \\ &= \mathcal{M}_{\boldsymbol{k}} \vee \mathcal{T}_{\boldsymbol{k}}. \end{aligned} \tag{A.6}$$

Combining the case of null word and equations (A.5) and (A.6) proves the desired claim of (A.4).

Using the decomposition (A.4), we can rewrite (A.3) in terms of the three sets, i.e., the set for null word, $\mathcal{M}_{\boldsymbol{k}} \vee \mathcal{F}$, and $\mathcal{M}_{\boldsymbol{k}} \vee \mathcal{T}_{\boldsymbol{k}}$. In particular, we have

$$\begin{aligned}
&\sum_{(\boldsymbol{w},W)\in\mathcal{M}_{\boldsymbol{k}}\vee\mathcal{T}_{\boldsymbol{k}}} \pi(\boldsymbol{w},W) \\
&= \sum_{(\boldsymbol{a},A)\in\mathcal{M}_{\boldsymbol{k}}} \sum_{(\boldsymbol{b},B)\in\mathcal{T}_{\boldsymbol{k}}} z^{\ell(\boldsymbol{ab})}(-1)^{|A\cup\mathfrak{J}(B,\ell(\boldsymbol{a}))|} \\
&= \sum_{(\boldsymbol{a},A)\in\mathcal{M}_{\boldsymbol{k}}} \sum_{(\boldsymbol{b},B)\in\mathcal{T}_{\boldsymbol{k}}} z^{\ell(\boldsymbol{a})+\ell(\boldsymbol{b})}(-1)^{|A|+|B|} \\
&= \left(\sum_{(\boldsymbol{a},A)\in\mathcal{M}_{\boldsymbol{k}}} \pi(\boldsymbol{a},A)\right)\left(\sum_{(\boldsymbol{b},B)\in\mathcal{T}_{\boldsymbol{k}}} \pi(\boldsymbol{b},B)\right). \qquad \text{(A.7)}
\end{aligned}$$

Similarly, one can show that

$$\sum_{(\boldsymbol{w},A)\in\mathcal{M}_{\boldsymbol{k}}\vee\mathcal{F}_2} \pi(\boldsymbol{w},A) = 2z \sum_{(\boldsymbol{w},A)\in\mathcal{M}_{\boldsymbol{k}}} \pi(\boldsymbol{w},A). \qquad \text{(A.8)}$$

Substituting (A.7) and (A.8) into (A.3) gives

$$f(z) = \sum_{(\boldsymbol{w},A)\in\mathcal{M}_{\boldsymbol{k}}} \pi(\boldsymbol{w},A) = 1 + 2zf(z) + f(z)T(z),$$

or equivalently,

$$f(z) = \frac{1}{1-2z-T(z)}, \qquad \text{(A.9)}$$

where $T(z)$ is the weight-enumeration function of elements in $\mathcal{T}_{\boldsymbol{k}}$ given by

$$T(z) := \sum_{(\boldsymbol{b},B)\in\mathcal{T}_{\boldsymbol{k}}} \pi(\boldsymbol{b},B). \qquad \text{(A.10)}$$

Determining $T(z)$ is now relatively easy. Recall that the overlap function $r_{\boldsymbol{k}}(i) = 1(k_1^{L-i} = k_{i+1}^L)$ shows exactly whether the length-$(L-i)$ prefix of $\boldsymbol{k}$ is also a suffix of $\boldsymbol{k}$. Let $\mathcal{R}_{\boldsymbol{k}} = \{i : 1 \le i \le L-1, r_{\boldsymbol{k}}(i) = 1\}$. For any cluster $(\boldsymbol{b},B) \in \mathcal{T}_{\boldsymbol{k}}$ with $\boldsymbol{b} = b_1 \dots b_n$, we must have $b_{n-L+1}^n = \boldsymbol{k}$ by Definition 6. So for any $i \in \mathcal{R}_{\boldsymbol{k}}$, i.e., $r_{\boldsymbol{k}}(i) = 1$, we have $b_{n-L+i+1}^n = k_{i+1}^L =$

k_1^{L-i}. Hence the pair

$$(\boldsymbol{b}k_{L-i+1}\dots k_L, B\cup\{(n+i-L+1,n+i)\})$$

is a cluster in $\mathcal{T}_{\boldsymbol{k}}$. It implies that for $i\in\mathcal{R}_{\boldsymbol{k}}$, the set

$$\mathcal{T}_{\boldsymbol{k},i} := \left\{ \begin{array}{r} \left(\boldsymbol{b}k_{L-i+1}^{L}, B\cup\{(n+i-L+1,n+i)\}\right) : \\ (\boldsymbol{b},B)\in\mathcal{T}_{\boldsymbol{k}}, n=\ell(\boldsymbol{b}) \end{array} \right\} \tag{A.11}$$

is a subset of $\mathcal{T}_{\boldsymbol{k}}$.

On the other hand, take any $(\boldsymbol{b},B)\in\mathcal{T}_{\boldsymbol{k}}$ with $\ell(\boldsymbol{b})=n$ and $B=\{(i_t,j_t):t=1,\dots,m\}$, where $1=i_1<i_2<\cdots<i_m<j_m=n$ and $i_m=n-L+1$. If $m=1$, then $\boldsymbol{b}=\boldsymbol{k}$ and $B=\{(1,L)\}$. Hence we consider the case when $m>1$. As $(\boldsymbol{b},B)$ is a cluster, $[i_{m-1},j_{m-1}]\cap[i_m,j_m]\neq\emptyset$ and $b_{i_{m-1}}^{j_{m-1}}=b_{i_m}^{j_m}=\boldsymbol{k}$. Therefore, we must have $b_{i_m}^{j_{m-1}}=k_1^v=k_{L-v+1}^L$, where $v=j_{m-1}-i_m+1$. Thus, $r_{\boldsymbol{k}}(L-v)=1$ and $(\boldsymbol{b},B)\in\mathcal{T}_{\boldsymbol{k},L-v}$. The above discussion then gives the following decomposition for $\mathcal{T}_{\boldsymbol{k}}$

$$\mathcal{T}_{\boldsymbol{k}} \;=\; \{(\boldsymbol{k},\{(1,L)\})\}\cup\left(\bigcup_{i\in\mathcal{R}}\mathcal{T}_{\boldsymbol{k},i}\right). \tag{A.12}$$

For the weight-enumeration of elements in $\mathcal{T}_{\boldsymbol{k}}$, we further claim that $\mathcal{T}_{\boldsymbol{k},i}\cap\mathcal{T}_{\boldsymbol{k},j}=\emptyset$ for all $i\neq j$. This simply follows from the definition of $\mathcal{T}_{\boldsymbol{k},i}$ in (A.11) that for any $(\boldsymbol{b},B)\in\mathcal{T}_{\boldsymbol{k},i}$ and $(\boldsymbol{b}',B')\in\mathcal{T}_{\boldsymbol{k},j}$, say $B=\{(i_t,j_t):t=1,\dots,m\}$ and $B'=\{(i_t,j_t):t=1,\dots,m'\}$, where the pairs (i_t,j_t) are arranged in ascending order, we have that $j_m-j_{m-1}=i$ for B and $j_{m'}-j_{m'-1}=j$ for B'. This proves our claim. Finally, using (A.12) and the fact that the sets $\mathcal{T}_{\boldsymbol{k},i}$ are disjoint, we obtain

$$\begin{aligned} T(z) &= \pi(\boldsymbol{k},\{(1,L)\}) + \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) \sum_{(\boldsymbol{b},B)\in\mathcal{T}_{\boldsymbol{k},i}} \pi(\boldsymbol{b},B) \\ &= z^{\ell(\boldsymbol{k})}(-1) + \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) \sum_{(\boldsymbol{b},B)\in\mathcal{T}_{\boldsymbol{k}}} z^{\ell(\boldsymbol{b})+i}\,(-1)^{|B|+1} \\ &= -z^L - \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) z^i T(z). \end{aligned}$$

Hence

$$T(z) \;=\; -\frac{z^L}{1+\sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i)z^i}.$$

Substituting the above into (A.9) proves Theorem 1.

Appendix B

Degree of $\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)$

In this chapter, we will determine the degree of polynomial $\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)$ that is required in the derivation in Chapter 3.

Proposition 10. *Let $\mathtt{A}_{\boldsymbol{k}}$ be the adjacency matrix for the digraph $G_{\boldsymbol{k}}$ associated with UW $\boldsymbol{k}$ defined in Chapter 3. Then*

$$\deg \det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z) = L. \tag{B.1}$$

Proof. First, we note that $\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)$ is divisible by $[(1-2z)(1 + \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) z^i) + z^L]$; hence, it is obvious that

$$\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z) \geq L.$$

It then remains to confirm that $\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z) \leq L$.

Assume $\underline{e}_{d_1^{L-1}}$ is a length-2^{L-1} vector, given by $(\underline{e}_{d_1^{L-1}})_{j+1} = 1$ if j has the binary representation d_1^{L-1}, and $(\underline{e}_{d_1^{L-1}})_{j+1} = 0$, otherwise. Let $\mathtt{H}_L = \mathtt{A}_{\boldsymbol{k}} + \underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^{\top}$; apparently, $\mathtt{H}_L$ is the adjacency matrix without UW forbidden constraint. As an example, if the length of $\boldsymbol{k}$ is 3, then

$$\mathtt{H}_L = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Further, it can be easily verified that all entries in $\mathtt{H}_L^{L-1}$ is 1 and

$$\mathtt{H}_L^i \underline{e}_{k_1^{L-1}} = \sum_{a \in \mathbb{F}^i} \underline{e}_{ak_1^{L-i-1}} \text{ for } 1 \leq i \leq L-1.$$

Define for $1 \leq i \leq L-1$,

$$\mathbb{E}_i := \{ak_1^{L-i-1} : \forall a \in \mathbb{F}^i\}.$$

By this definition, we have $\mathbb{E}_0 = \{k_1^{L-1}\}$, $\mathbb{E}_1 = \{0k_1^{L-2}, 1k_1^{L-2}\}$ and $\mathbb{E}_2 = \{00k_1^{L-3}, 01k_1^{L-3}, 10k_1^{L-3}, 11k_1^{L-3}\}$, etc. Therefore, $(\mathtt{H}^i \underline{e}_{k_1^{L-1}})_{j+1} \neq 0$ only when j has the binary representation in $\mathbb{E}_i$.

We then claim that if $a \in \mathbb{E}_i$ and $a \in \mathbb{E}_j$ for $i < j$, then

$$\mathbb{E}_i \subset \mathbb{E}_j. \tag{B.2}$$

This claim is proved by noting that $a \in \mathbb{E}_i$ and $a \in \mathbb{E}_j$ for $i < j$ implies $k_1^{L-j-1} = k_{1+j-i}^{L-i-1}$, which in turns implies $\{p_1^{j-i-1}k_1^{j-i}\} \subset \{q_1^{j-i}\}$ for all $p_1^{j-i-1} \in \mathbb{F}^{j-i-1}$ and $q_1^{j-i} \in \mathbb{F}^{j-i}$.

Now for two square matrices A and B, we know that

$$\begin{aligned}(A-B)^{L-1} = \; & A^{L-1} - A^{L-2}B \\ & -A^{L-3}B(A-B) \\ & -A^{L-4}B(A-B)^2 \\ & \vdots \\ & -AB(A-B)^{L-3} \\ & -B(A-B)^{L-2}\end{aligned}$$

Applying this expression to $\mathtt{A}_{\boldsymbol{k}}^{L-1} = (\mathtt{H}_L - \underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top)^{L-1}$ yields:

$$\begin{aligned}
\mathtt{A}_{\boldsymbol{k}}^{L-1} &= (\mathtt{H}_L - \underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top)^{L-1} \\
&= \mathtt{H}_L^{L-1} - \mathtt{H}_L^{L-2}(\underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top) \\
&\quad -\mathtt{H}_L^{L-3}(\underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top)(\mathtt{H}_L - \underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top) \\
&\quad -\mathtt{H}_L^{L-4}(\underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top)(\mathtt{H}_L - \underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top)^2 \\
&\quad \vdots \\
&\quad -\mathtt{H}_L(\underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top)(\mathtt{H}_L - \underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top)^{L-3} \\
&\quad -(\underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top)(\mathtt{H}_L - \underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top)^{L-2}
\end{aligned} \tag{B.3}$$

There are L terms in (B.3). For the first term, all entries of $\mathtt{H}_L^{L-1}$ are 1; so all rows of $\mathtt{H}_L^{L-1}$ are the same. For the second term, since $\mathtt{H}_L^{L-2} \underline{e}_{k_1^{L-1}} = \sum_{a \in \mathbb{E}_{L-2}} \underline{e}_a$, the entries of $\mathtt{H}_L^{L-2}(\underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top)$ are all zeros except those in the $(j+1)$th row for every j having its binary representation in $\mathbb{E}_{L-2}$. In particular, these non-zero rows of $\mathtt{H}_L^{L-2}(\underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top)$ are all the same and are equal to $\underline{e}_{k_2^L}^\top$ because the elements of vector $\mathtt{H}_L^{L-2} \underline{e}_{k_1^{L-1}}$ are all zeros except those 1's located at position $(j+1)$ for every j with binary representation in $\mathbb{E}_{L-2}$. For the third term, we can similarly infer that $\mathtt{H}_L^{L-3}(\underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top)(\mathtt{H}_L - \underline{e}_{k_1^{L-1}} \underline{e}_{k_2^L}^\top)$ has zero entries in all rows except those in the $(j+1)$th row for every j having its binary representation in $\mathbb{E}_{L-3}$, and these non-zero rows are all equal. Continueing this inference, we can identify the each term in (B.3).

According to (B.3) and the discussion above as well as the claim of (B.2), we conclude that there are at most L distinct rows in $\mathtt{A}_{\boldsymbol{k}}^{L-1}$. This then indicates $\text{rank}(\mathtt{A}_{\boldsymbol{k}}^{L-1}) \leq L$, which in turns implies $\text{rank}(\mathtt{A}_{\boldsymbol{k}}^{L}) \leq L$. As a result, the algebraic multiplicity of eigenvalue 0 for $\mathtt{A}_{\boldsymbol{k}}$ is at least $2^{L-1} - L$. Equivalently, it means that the degree of the LCCDE of $c_{\boldsymbol{k},n}$, i.e., $h_{\boldsymbol{k}}(z) = \det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)$, is at most L. The proof is thus completed. □

Appendix C

Verification of Algorithms 5 and 6

For completeness, we verify Algorithms 5 and 6 in this chapter.

For message u_1, i.e., the most likely message, we have from line 1 in Algorithm 5 that $m = 1$ and $n = 0$ since $F_{\boldsymbol{k},0} = c_{\boldsymbol{k},0} = 1$. This results in the encoding output of the null codeword. In parallel, when receiving a null codeword, Algorithm 6 will assign $m = 1$ at line 2 as $F_{\boldsymbol{k},-1} = 0$. This completes the verification of Algorithms 5 and 6 for message u_1.

For $m \geq 2$, we shall show that $\phi_{\boldsymbol{k}}(u_m)$ is a bijection from $\mathcal{U}_{\boldsymbol{k}}(n) = \{u_m : F_{\boldsymbol{k},n-1} < m \leq F_{\boldsymbol{k},n}\}$ to $\mathcal{C}_{\boldsymbol{k}}(n)$, and $\psi_{\boldsymbol{k}}$ is the functional inverse of $\phi_{\boldsymbol{k}}$.

To this end, when being given codeword $\boldsymbol{c} = c_1 \dots c_n$, Algorithm 6 will output $\psi_{\boldsymbol{k}}(\boldsymbol{c}) = m$ satisfying

$$m = \sum_{i=1}^{n} c_i \underline{x}_{\boldsymbol{k}}^{\top} \left(\prod_{j=1}^{i-1} \mathtt{A}_{\boldsymbol{k},c_i} \right) \mathtt{A}_{\boldsymbol{k},0} \mathtt{A}_{\boldsymbol{k}}^{(n+L-1)-i} \underline{y}_{\boldsymbol{k}} + F_{\boldsymbol{k},n-1} + 1. \tag{C.1}$$

Equivalently,

$$m = \sum_{i=1}^{n} c_i \left| \mathcal{C}_{\boldsymbol{k}}(c_1^{i-1}0, n) \right| + F_{\boldsymbol{k},n-1} + 1. \tag{C.2}$$

It can then be verified that with the above m, lines 3-10 in Algorithm 5 will produce $\phi_{\boldsymbol{k}}(u_m) = \boldsymbol{c}$; hence, $\psi_{\boldsymbol{k}}$ is the functional inverse of $\phi_{\boldsymbol{k}}$. Moreover, (C.1), or its equivalence (C.2), implies that $\phi_{\boldsymbol{k}}$ is at least injective.

We then prove the bijectiveness of $\phi_{\boldsymbol{k}}$ as follows. Give any two codewords $\boldsymbol{c} = c_1 \dots c_n$

and $\tilde{\boldsymbol{c}} = \tilde{c}_1 \dots \tilde{c}_n$ with $\phi_{\boldsymbol{k}}(u_m) = \boldsymbol{c}$ and $\phi_{\boldsymbol{k}}(u_{m'}) = \tilde{\boldsymbol{c}}$. Suppose $c_i = \tilde{c}_i$ for $1 \le i < s$ and $c_s = 1 \neq \tilde{c}_s = 0$. Then, line 5 in Algorithm 5 indicates that

$$m > \sum_{i=1}^{t} c_i \left| \mathcal{C}_{\boldsymbol{k}}(c_1^{i-1}0, n) \right| + F_{\boldsymbol{k},n-1} + 1 \ge m';$$

hence $m \neq m'$. This concludes that $\phi_{\boldsymbol{k}}$ must be bijective.

It remains to show that $\phi_{\boldsymbol{k}}(u_m) \in \mathcal{C}_{\boldsymbol{k}}(n)$ for any m satisfying $F_{\boldsymbol{k},n-1} < m \le F_{\boldsymbol{k},n}$.

Order the elements in set $\mathbb{F}^n$ alphabetically as $\{00\cdots0, 00\cdots01, \cdots, 11\cdots1\}$, and let $\vartheta(t)$ be the tth element in the ordered list. Then $\phi_{\boldsymbol{k}}$ actually assigns codeword $\vartheta(t)$ to u_m, where t is the smallest integer satisfying

$$m = F_{\boldsymbol{k},n-1} + \sum_{j=1}^{t} |\mathcal{C}_{\boldsymbol{k}}(\vartheta(j), n)|. \tag{C.3}$$

It is however inefficient to assign codewords according to (C.3). Note that $|\mathcal{C}_{\boldsymbol{k}}(\vartheta(j), n)| = 1$ if $\vartheta(j) \in \mathcal{C}_{\boldsymbol{k}}(n)$, and $|\mathcal{C}_{\boldsymbol{k}}(\vartheta(j), n)| = 0$, otherwise.

So we reformulate (C.3) using $\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}, n)$ defined in (4.12) to determine $\vartheta(t)$ in a bit-by-bit fashion. Specifically, if the first bit of $\vartheta(t)$, i.e., $(\vartheta(t))_1$, is 1, then

$$\begin{aligned} m - F_{\boldsymbol{k},n-1} &= \sum_{j=1}^{t} |\mathcal{C}_{\boldsymbol{k}}(\vartheta(j), n)| \\ &= |\mathcal{C}_{\boldsymbol{k}}(0, n)| + \sum_{j=2^{n-1}+1}^{t} |\mathcal{C}_{\boldsymbol{k}}(\vartheta(j), n)| \qquad \text{(C.4)} \\ &> |\mathcal{C}_{\boldsymbol{k}}(0, n)|. \qquad \text{(C.5)} \end{aligned}$$

If however $(\vartheta(t))_1 = 0$, then by the alphabetical ordering, we should have

$$\begin{aligned} m - F_{\boldsymbol{k},n-1} &= \sum_{j=1}^{t} |\mathcal{C}_{\boldsymbol{k}}(\vartheta(j), n)| \\ &\le |\mathcal{C}_{\boldsymbol{k}}(0, n)|. \qquad \text{(C.6)} \end{aligned}$$

(C.5) and (C.6) then confirm (4.14) for $i = 1$.

The next step is to determine the second bit $(\vartheta(t))_2$. Given the first bit is 1, (C.4) implies $m - F_{\boldsymbol{k},n-1} - |\mathcal{C}_{\boldsymbol{k}}(0,n)| = \sum_{j=2^{n-1}+1}^{t} |\mathcal{C}_{\boldsymbol{k}}(\vartheta(j),n)|$. We can similarly compare $m - F_{\boldsymbol{k},n-1} - |\mathcal{C}_{\boldsymbol{k}}(0,n)|$ with $|\mathcal{C}_{\boldsymbol{k}}(10,n)|$ to determine the second bit of $\vartheta(t)$. In other words, if $(\vartheta(t))_2 = 1$, then

$$\begin{aligned} & m - F_{\boldsymbol{k},n-1} - |\mathcal{C}_{\boldsymbol{k}}(0,n)| \\ = & \sum_{j=2^{n-1}+1}^{t} |\mathcal{C}_{\boldsymbol{k}}(\vartheta(j),n)| \\ = & |\mathcal{C}_{\boldsymbol{k}}(10,n)| + \sum_{j=2^{n-1}+2^{n-2}+1}^{t} |\mathcal{C}_{\boldsymbol{k}}(\vartheta(j),n)| \\ > & |\mathcal{C}_{\boldsymbol{k}}(10,n)|; \end{aligned} \tag{C.7}$$

else, we should have

$$\begin{aligned} & m - F_{\boldsymbol{k},n-1} - |\mathcal{C}_{\boldsymbol{k}}(0,n)| \\ = & \sum_{j=2^{n-1}+1}^{t} |\mathcal{C}_{\boldsymbol{k}}(\vartheta(j),n)| \\ \leq & |\mathcal{C}_{\boldsymbol{k}}(10,n)|. \end{aligned} \tag{C.8}$$

In case at the time to determined the second bit, the first bit of $\vartheta(t)$ is 0, we can determine $(\theta(t))_2$ by comparing $m - F_{\boldsymbol{k},n-1}$ with $|\mathcal{C}_{\boldsymbol{k}}(00,n)|$.

Continue the process for the remaining bits. Assume c_1^{i-1} have been determined. Let

$$\Upsilon(c_1^{i-1}) := \{j : c_j = 1\}.$$

Then we can compare the

$$\begin{aligned} & m - F_{\boldsymbol{k},n-1} - \sum_{j \in \Upsilon(c_1^{i-1})} |\mathcal{C}_{\boldsymbol{k}}(c_1^{j-1}0,n)| \\ = & \sum_{j=1+J_n(c_1^{i-1})}^{t} |\mathcal{C}_{\boldsymbol{k}}(\vartheta(j),n)| \end{aligned}$$

with $|\mathcal{C}_{\boldsymbol{k}}(c_1^{i-1}0, n)|$ to determine the ith bit, where $J_n(\boldsymbol{c}) = \sum_{j\in\Upsilon(\boldsymbol{c})} 2^{n-j}$. In other words, $c_i = 0$ if $m - F_{\boldsymbol{k},n-1} - \sum_{j\in\Upsilon(c_1^{i-1})} |\mathcal{C}_{\boldsymbol{k}}(c_1^{j-1}0, n)| \le |\mathcal{C}_{\boldsymbol{k}}(c_0^{i-1}0, n)|$, and $c_i = 1$, otherwise.

Finally, by $F_{\boldsymbol{k},n-1} < m \le F_{\boldsymbol{k},n}$ and (4.13), (4.16) should hold, i.e.,

$$m - \ F_{\boldsymbol{k},n-1} - \sum_{i=1}^{n-1} c_i \left|\mathcal{C}_{\boldsymbol{k}}(c_1^{i-1}0, n)\right| = |\mathcal{C}_{\boldsymbol{k}}(c_1^n, n)| = 1,$$

and

$$c_1^n = \vartheta(t).$$

Appendix D

Computation of $\lim_{t\to\infty} L_{\boldsymbol{k},t}$ for all-zero UW and uniform i.i.d. source

By (3.21), we have

$$\sum_{n=0}^{\infty} c_{\boldsymbol{k},n} z^n = \frac{f(z)}{h_{\boldsymbol{k}}(z)} = \frac{f(z)}{1-\sum_{i=1}^{L} z^i}.$$

Then,

$$\begin{aligned}
f(z) &= \sum_{n=0}^{\infty} c_{\boldsymbol{k},n} z^n \left(1-\sum_{i=1}^{L} z^i\right) \\
&= \sum_{n=0}^{\infty} c_{\boldsymbol{k},n} z^n - \sum_{n=0}^{\infty}\sum_{i=1}^{L} c_{\boldsymbol{k},n} z^{n+i} \quad (n' = n+i) \\
&= \sum_{n=0}^{L} c_{\boldsymbol{k},n} z^n + \sum_{n=L+1}^{\infty} c_{\boldsymbol{k},n} z^n - \sum_{n'=1}^{L}\sum_{i=0}^{n'-1} c_{\boldsymbol{k},i} z^{n'} - \sum_{n'=L+1}^{\infty}\sum_{i=1}^{L} c_{\boldsymbol{k},n'-i} z^{n'} \\
&= \sum_{n=0}^{L} c_{\boldsymbol{k},n} z^n - \sum_{n'=1}^{L}\sum_{i=0}^{n'-1} c_{\boldsymbol{k},i} z^{n'} + \sum_{n=L+1}^{\infty}\left(c_{\boldsymbol{k},n} - \sum_{i=1}^{L} c_{\boldsymbol{k},n-i}\right) z^n \\
&\overset{(i)}{=} \sum_{n=0}^{L} c_{\boldsymbol{k},n} z^n - \sum_{n=1}^{L}\sum_{i=0}^{n-1} c_{\boldsymbol{k},i} z^n
\end{aligned}$$

$$
\begin{aligned}
&= 1 + \sum_{n=1}^{L} c_{\boldsymbol{k},n} z^n - \sum_{n=1}^{L} \sum_{i=0}^{n-1} c_{\boldsymbol{k},i} z^n \\
&= 1 + \sum_{n=1}^{L} \left(c_{\boldsymbol{k},n} - \sum_{i=0}^{n-1} c_{\boldsymbol{k},i} \right) z^n \\
&\overset{(ii)}{=} 1 - \sum_{n=2}^{L} z^n,
\end{aligned}
$$

where (i) holds because of (4.18) and (ii) is a consequence of (4.19). Note that the all-zero and all-one UWs are equivalent; hence, (4.18) and (4.19) also apply here. Accordingly,

$$
\sum_{n=0}^{\infty} c_{\boldsymbol{k},n} z^n = \frac{1 - \sum_{n=2}^{L} z^n}{1 - \sum_{i=1}^{L} z^i} = 1 + \frac{z}{1 - \sum_{i=1}^{L} z^i}. \tag{D.1}
$$

Denote by $\lambda_1 \cdots \lambda_L$ the nonzero eigenvalues of $\mathtt{A}_{\boldsymbol{k}}$, and assume without loss of generality that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_L|$. Then

$$
c_{\boldsymbol{k},n} = \delta_n + \sum_{i=1}^{L} a_i (\lambda_i)^n
$$

where $a_1 \cdots a_L$ are coefficients such that (D.1) holds. We can also obtain the closed-form expression for $F_{\boldsymbol{k},n}$ as

$$
F_{\boldsymbol{k},n} = 1 + \sum_{i=1}^{L} a_i \frac{\lambda_i^{n+1} - 1}{\lambda_i - 1}.
$$

Consider uniform i.i.d. source of alphabet size M^t with $M > 1$. The average codeword length per source symbol is given by

$$
\begin{aligned}
L_{\boldsymbol{k},t} &= \frac{1}{t} \left(L + \sum_{i=2}^{M^t} p_i \ell(\phi(u_i)) \right) \\
&= \frac{1}{t} \left(L + \frac{1}{M^t} \sum_{i=2}^{M^t} \ell(\phi(u_i)) \right)
\end{aligned}
$$

Let N be the smallest integer such that $F_{\boldsymbol{k},N} \geq M^t > F_{\boldsymbol{k},N-1}$. Then

$$
\begin{aligned}
L_{\boldsymbol{k},t} &\geq \frac{1}{t} \left(L + \frac{1}{F_{\boldsymbol{k},N}} \sum_{i=2}^{M^t} \ell(\phi(u_i)) \right) \\
&\geq \frac{1}{t} \left(L + \frac{1}{F_{\boldsymbol{k},N}} \sum_{i=2}^{N-1} i \cdot c_{\boldsymbol{k},i} \right) \\
&\geq \frac{1}{\log_M(F_{\boldsymbol{k},N})} \left(L + \frac{1}{F_{\boldsymbol{k},N}} \sum_{i=2}^{N-1} i \cdot c_{\boldsymbol{k},i} \right).
\end{aligned}
$$

Consequently,

$$\begin{aligned}
\lim_{t\to\infty} L_{\boldsymbol{k},t} &\geq \lim_{N\to\infty} \frac{1}{\log_M(F_{\boldsymbol{k},N})}\left(L+\frac{1}{F_{\boldsymbol{k},N}}\sum_{i=2}^{N-1} i\cdot c_{\boldsymbol{k},i}\right)\\
&= \lim_{N\to\infty} \frac{\sum_{i=2}^{N-1} i\cdot c_{\boldsymbol{k},i}}{F_{\boldsymbol{k},N}\log_M(F_{\boldsymbol{k},n})}\\
&= \lim_{N\to\infty} \frac{\displaystyle\sum_{i=1}^{L} a_i\frac{\lambda_i[(N-1)\lambda_i^N - N\lambda_i^{N-1}+1]}{(\lambda_i-1)^2}}{\left(1+\displaystyle\sum_{i=1}^{L} a_i\frac{\lambda_i^{N+1}-1}{\lambda_i-1}\right)\log_M\left(1+\displaystyle\sum_{i=1}^{L} a_i\frac{\lambda_i^{N+1}-1}{\lambda_i-1}\right)}\\
&= \frac{1}{\log_M(\lambda_1)} = \frac{1}{\log_M(g_{\boldsymbol{k}})}.
\end{aligned}$$

This concludes to

$$\lim_{t\to\infty} L_{\boldsymbol{k},t} \geq \frac{\log_2(M)}{\log_2(g_{\boldsymbol{k}})} = \frac{\mathrm{H}(\mathcal{U})}{\log_2(g_{\boldsymbol{k}})}$$

when $\boldsymbol{k}$ is the all-zero sequence.

Bibliography

[1] N. Alon and A. Orlitsky, "A lower bound on the expected length of one-to-one codes," *IEEE Trans. Inf. Theory*, vol. 40, no. 5, pp. 1670-1672, September 1994.

[2] J. Bang-Jensen and G. Z. Gutin, *Theory, Algorithms and Applications*, Springer Monographs in Mathematics, 2009.

[3] N. Biggs, *Algebraic Graph Theory*, Cambridge Mathematical Library, 1994.

[4] C. Blundo and R. D. Prisco, "New bounds on the expected length of one-to-one codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 1, pp. 246-250, January 1996.

[5] J. Cheng, T.-K. Huang and C. Weidmann, "New bounds on the expected length of optimal one-to-one codes," *IEEE Trans. Inf. Theory*, vol. 53, no. 5, pp. 1884-1895, May 2007.

[6] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York, NY: John Wiley & Sons, 1991.

[7] R. Doroslovački, "The set of all the words of length n over any alphabet with a forbidden good subword," *Univ. u Novom Sadu, Zb. Rad. Prirod.-Mat. Fak. Ser. Mat.*, 23:2, pp. 239-244, 1993.

[8] R. Doroslovački, "The set of all the words of length n over alphabet $\{0,1\}$ with any forbidden subword of length three," *Univ. u Novom Sadu, Zb. Rad. Prirod.-Mat. Fak. Ser. Mat.*, 25: 2, pp. 111-115, 1995.

[9] R. Doroslovački, "Binary n-Words without the subword $1010\cdots10$," *Novi Sad J. Math.*, vol. 28, no. 2, pp. 127-133, 1998.

[10] R. Doroslovački, "On binary n-words with forbidden 4-subwords," *Novi Sad J. Math.*, vol. 29, no. 1, pp. 27-32, 1999.

[11] R. Doroslovački, "n-words over any alphabet with forbidden any 3-subwords," *Novi Sad J. Math.*, vol. 30, no. 2, pp. 159-163, 2000.

[12] J. G. Dunham, "Optimal noiseless coding of random variables," *IEEE Trans. Inf. Theory*, vol. IT-26, no. 3, p. 345, May 1980.

[13] Sam E. Ganis, *Notes on the Fibonacci Sequence*, Amer. Math. Monthly, 1959, pp. 129-130.

[14] C. Godsil and G. F. Royle, *Algebraic Graph Theory*, Springer, 2001.

[15] I. Goulden and D. M. Jackson, "An inversion theorem for cluster decompositions of sequences with distinguished subsequences," *J. London Math. Soc*, pp. 567-576, 1979.

[16] L. J. Guibas and A. M. Odlyzko, "Periods in strings," *J. Combinatorial Theory*, series A 30, pp. 19-42, 1981.

[17] K. M. Hoffman and R. Kunze, *Linear Algebra*, 2nd edition, Pearson,1971.

[18] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd edition, Cambridge University Press, 2012.

[19] E. J. Kupin and D. S. Yuster, "Generalizations of the Goulden-Jackson cluster method," *J. Difference Eq. Appl.*, 16:12, pp. 1463-1480, 2010.

[20] S. K. Leung-Yan-Cheong and T. M. Cover, "Some equivalences between Shannon entropy and Kolmogorov complexity," *IEEE Trans. on Information theory*, vol. IT-24, no. 3, pp. 331-338, May 1978.

[21] J. Noonan, "New upper bounds for the connective constants of self-avoiding walks," *J. Statistical Physics*, vol. 91, nos. 5/6, 1998.

[22] J. Noonan and D. Zeilberger, "The Goulden-Jackson cluster method: extensions, applications, and implementations," *J. Difference Eq. Appl.*, 5: pp. 355-377, 1999.

[23] J. Rissanen, "Tight lower bounds for optimum code length," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 348-349, March 1982.

[24] E. Rivals and S. Rahmann, "Combinatorics of periods in strings," *J. Combinatorial Theory*, series A 104, pp. 95-113, 2003.

[25] S. A. Savari, "On one-to-one codes for memoryless cost channels," *IEEE Trans. Inf. Theory*, vol. 54, no. 1, pp. 367-379, January 2008.

[26] S. A. Savari and A.Naheta, "Bounds on the expected cost of one-to-one codes," *IEEE International Symposium on Information Theory*, June 2004.

[27] R. Stanley, *Enumerative Combinatorics*, vols. 1 and 2, Cambridge Studies in Advanced Mathematics, 2011.

[28] W. Szpankowski, "A one-to-one code and its anti-redundancy," *IEEE Trans. Inf. Theory*, vol. 54, no. 10, pp. 4762-4766, October 2008.

[29] W. Szpankowskia and S. Verd u, "Minimum expected length of fixed-to-variable lossless compression of memoryless sources," *IEEE International Symposium on Information Theory*, Seoul, Korea, July 2009.

[30] E. I. Verriest, "An achievable bound for optimal noiseless coding of a random variable," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 4, pp. 592-594, July 1986.

[31] X. Wen, "The symbolic Goulden-Jackson cluster method," *J. Difference Eq. Appl.*, 11:2, pp. 173-179, 2006.

[32] A. D. Wyner, "An upper bound on the entropy series," *Inf. Control*, vol. 20, 30: pp. 176-181, 1972.

[33] http://bcl?.comli.eu/?download-e?n.html

[34] http://corpus.canterbury.ac.nz/descriptions/

[35] http://oxforddictionaries.com/words/what-is-the-frequency-of-the-letters-of-the-alphabet-in-english