On Hierarchies of Balanced Sequences

Sagi Marcovich, *Member, IEEE*, Tuvi Etzion, *Life Fellow, IEEE*, and Eitan Yaakobi, *Senior Member, IEEE*

***Abstract*— Balanced sequences and balanced codes have attracted a lot of research in the last seventy years due to their diverse applications in information theory as well as other areas of computer science and engineering. There have been some methods to classify balanced sequences. This work suggests two new different hierarchies to classify these sequences. The first one is based on the largest ℓ for which each ℓ-tuple is contained the same amount of times in the sequence. This property is a generalization for the property required for de Bruijn sequences. The second hierarchy is based on the number of balanced derivatives of the sequence. Enumeration for each such family of sequences and efficient encoding and decoding algorithms are provided in this paper.**

***Index Terms*— Balanced sequences, de Bruijn sequences, de Bruijn graph, Knuth's balancing algorithm, derivatives of sequences.**

I. Introduction

A BINARY sequence in which the number of *zeros* equals to the number of *ones* (or the latter number is larger by one than the former number if the length of the sequence is odd) is called a *balanced sequence*. It is known that the redundancy of this family of sequences of length n is $0.5\log n+\mathcal{O}(1)$ [28]. Knuth [28] designed an efficient encoding algorithm with linear time complexity which encodes an arbitrary binary sequence of length n to a balanced one using $\log n+\Theta(\log\log n)$ redundancy bits. In the following years, various generalizations of the family of balanced sequences and Knuth's algorithm were studied, e.g., [1], [2], [24], [42], [44], [45], [48], [49], [50], [52].

Classifying balanced sequences was done throughout the years based on their properties. The first classification was suggested by Golomb [21] in his definition of three randomness postulates. His hierarchy was further generalized in [9] and [20]. Later, balanced sequences were classified based on their spectral-null order, e.g., [42], [45], [49]. This classification is important in the context of magnetic recording [25]. Another classification of balanced sequences can be done by their linear complexity [8] which is also related to the depth of a sequence [4], [13] and its derivatives [39]. Moreover, there has been a particular interest in the classification of de Bruijn sequences, see for example [8], [34].

In this paper, further classification of balanced sequences is considered and as a consequence, a hierarchy of these sequences is provided. The first hierarchy is based on an integer ℓ for which each ℓ-tuple appears the same number μ of times as a window of length ℓ in a cyclic sequence. If the length of the sequence is $\mu 2^\ell$, it contains each binary ℓ-tuple as a window of ℓ consecutive symbols exactly μ times. When $\mu=1$ this family of sequences coincides with the well-known family of de Bruijn sequences. This motivates some comparisons of these sequences with de Bruijn sequences.

The binary de Bruijn graph of order ℓ, G_ℓ, was introduced in 1946 by de Bruijn [6]. His target in introducing this graph was to find a recursive method to enumerate the number of cyclic binary sequences of length 2^ℓ such that each ℓ-tuple appears as a window of ℓ consecutive symbols exactly once in each sequence. These sequences were later called *de Bruijn sequences*. It should be mentioned that in parallel also Good [22] defined the same graph. The vertices of G_ℓ are represented by the binary $(\ell-1)$-tuples, and the edges are associated with the binary ℓ-tuples. The edge $u\to v$ is associated with an ℓ-tuple x when v is associated with the $(\ell-1)$-prefix of x and u is associated with the $(\ell-1)$-suffix from x. Eulerian cycles in de Bruijn graphs, i.e., cycles that visit each edge of G_ℓ exactly once, are called *de Bruijn cycles*. It was proved in [6] that the number of de Bruijn cycles in G_ℓ is $2^{2^{\ell-1}-\ell}$.

Each de Bruijn cycle induces a single (cyclic) de Bruijn sequence of length 2^ℓ, by picking any edge in the cycle as a starting point, considering its first entry and appending the first entry of each consecutive edge in the cycle. All sequences that can be generated in this way from the same cycle are considered as the same sequence. If we decide to designate a specific ℓ-tuple as the first ℓ-tuple of the cyclic sequence, then the number of such de Bruijn sequences is $2^{2^{\ell-1}}$ since each one of the 2^ℓ distinct ℓ-tuples can be chosen as the first one. This is the same as to consider these sequences as (acyclic) words, but ℓ-tuples are also wrapped around from the end of the word to its starting point. In both scenarios (cyclic or acyclic), the asymptotic rate of this family of sequences is $1/2$, i.e., the rate of these sequences approaches $1/2$ as $\ell\to\infty$.

One of the first applications of the de Bruijn graph was in the introduction of shift-register sequences and linear feedback shift registers [21]. Throughout the years, an extensive number of papers have studied de Bruijn graph and its sequences. Several of those include [8], [10], [12], [15], [16], [23], [30],

Manuscript received 24 July 2022; revised 15 November 2022; accepted 29 November 2022. Date of publication 21 December 2022; date of current version 21 April 2023. This work was supported in part by the Technion Data Science Initiative. The work of Sagi Marcovich and Eitan Yaakobi was supported in part by the United States–Israel Binational Science Foundation (BSF) under Grant 2018048. The work of Tuvi Etzion was supported in part by the Israel Science Foundation (ISF) under Grant 222/19. An earlier version of this paper was presented in part at the IEEE International Symposium on Information Theory, Melbourne, Australia, 2021 [DOI: 10.1109/ISIT45174.2021.9517873]. *(Corresponding author: Sagi Marcovich.)*

The authors are with the Department of Computer Science, Technion—Israel Institute of Technology, Haifa 3200003, Israel (e-mail: sagimar@cs.technion.ac.il; etzion@cs.technion.ac.il; yaakobi@cs.technion.ac.il).

Communicated by X. Zhang, Associate Editor for Coding and Decoding.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2022.3227467.

Digital Object Identifier 10.1109/TIT.2022.3227467

[31], [33], [41], [47]. Most recently, DNA storage has brought fresh interest to this family of sequences; see [3], [26], [46].

The second hierarchy that we provide is based on the number of balanced derivatives produced by the sequence. For a cyclic sequence $\boldsymbol{s} = [s_0, s_1, \ldots, s_{n-1}]$, the *derivative* of $\boldsymbol{s}$, $\boldsymbol{s}'$, is defined by $\boldsymbol{s}' = [s_0 + s_1, s_1 + s_2, \ldots, s_{n-1} + s_0]$. These sequences and their applications were studied in several papers, e.g., [4], [13], [19], [31], [39]. The two hierarchies which are defined are not completely independent. For example, we will prove in the sequel that in a sequence each 2-tuple appears the same number of times if and only if the sequence and its derivative are both balanced.

Note that throughout the paper, most of the time and unless mentioned otherwise, the sequences studied will be cyclic ones, i.e., windows are read cyclically, but the order of their symbols will be important, i.e., two sequences which are the same cyclically, but have different order of symbols will be considered as distinct sequences. For example, the sequences 001010 and 000101 are considered distinct, although that cyclically they are the same sequence.

The rest of this paper is organized as follows. In Section II, we formally define the families of sequences studied in this paper, and review several previous results. In Section III, we study the set of sequences of length n that contain each 2-tuple exactly $n/4$ times. We present an efficient encoding algorithm for this set of sequences. This algorithm which requires $2\log n + \Theta(\log\log n)$ redundancy bits is based on Knuth's algorithm. In Section IV, we present an enumeration for sequences of length $\mu 2^\ell$ that contain each binary ℓ-tuple as a window exactly μ times, for all admissible values of ℓ and μ. This enumeration is based on a generalization of the de Bruijn graph, denoted by $G_{\ell,\mu}$, which has the same set of vertices as G_ℓ, and each edge of G_ℓ is duplicated μ times. That is, each vertex of $G_{\ell,\mu}$ is associated with a binary $(\ell-1)$-tuple, and it has two sets of μ parallel outgoing edges, each such set of parallel edges is associated with a different binary ℓ-tuple. This enumeration implies that the cardinality of this set of sequences is

$$2^{2^{\ell-1}} \binom{2\mu-1}{\mu-1}^{2^{\ell-1}}.$$

Moreover, we derive explicit expressions for the redundancy and the asymptotic rate of this set of sequences for some values of ℓ and μ. Later, we study a variation of this set of sequences where two sequences that are identical cyclically are considered to be the same sequence; it is shown that the cardinality of this set of sequences is

$$\frac{2^{2^{\ell-1}-\ell}}{\mu} \sum_{\substack{i=1 \\ \lambda=\gcd(\mu,i)}}^{\mu} \binom{2\lambda-1}{\lambda-1}^{2^{\ell-1}}.$$

Furthermore, in Section V we present a generic encoding scheme for this set of sequences for $\ell = \mathcal{O}(\log\log n)$ that uses $2^{\ell-1}\log\mu + \Theta(2^{\ell-1}\log\log\mu)$ redundancy bits. In Section VI, we study sequences of length n which satisfy the property that their first k derivatives are balanced, for a fixed integer k. We study the cardinality of this novel family of sequences for several values of k and present an efficient encoding algorithm which requires $k\log n + \Theta(\log\log n)$ redundancy bits. This algorithm is also based on Knuth's algorithm. Finally, a conclusion is presented in Section VII.

II. Definitions and Preliminaries

For two integers $i, k \in \mathbb{N}$ such that $i \leqslant k$ we denote by $[i, k]$ the set $\{i, \ldots, k\}$ and use $[k]$ as a shorthand for $[0, k-1]$. Let $n \in \mathbb{N}$ be an integer, let $\boldsymbol{s} = [s_0, \ldots, s_{n-1}] \in \mathbb{F}_2^n$ denote a cyclic sequence of length n, and let $\boldsymbol{y} = (y_0, \ldots, y_{n-1})$ denote an acyclic sequence of length n. For two positive integers $i, k \leqslant n$, let $\boldsymbol{s}_{i,k}$ denote the *window* $(s_i, \ldots, s_{i+k-1})$, where the indices are taken modulo n. Additionally, let $\mathrm{Pref}_k(\boldsymbol{s}) \triangleq \boldsymbol{s}_{0,k}$, $\mathrm{Suff}_k(\boldsymbol{s}) \triangleq \boldsymbol{s}_{n-k,k}$ denote the *k-prefix*, *k-suffix* of $\boldsymbol{s}$, respectively, and let $w_H(\boldsymbol{s})$ denote the *Hamming weight* of $\boldsymbol{s}$. The *binary compliment* of $\boldsymbol{s}$, is the sequence $\bar{\boldsymbol{s}} \triangleq [\bar{s}_0, \ldots, \bar{s}_{n-1}]$, where $\bar{s}_i = 1 - s_i$ is the binary complement of s_i for every $i \in [n]$. Let $p(\boldsymbol{s})$ denote the *period* of $\boldsymbol{s}$, that is, the smallest positive integer that satisfies $s_i = s_{i+p(\boldsymbol{s}) \bmod n}$ for every $i \in [n]$. If $p(\boldsymbol{s}) = |\boldsymbol{s}|$ we say that $\boldsymbol{s}$ is *aperiodic*. Otherwise, we say that $\boldsymbol{s}$ is *periodic*. The notation $\boldsymbol{s} \circ \boldsymbol{v}$ denotes the concatenation of $\boldsymbol{s}$ and another sequence $\boldsymbol{v}$, and $\boldsymbol{s}^i$ denotes the concatenation of $\boldsymbol{s}$ i times, i.e., $\boldsymbol{s}^i = \boldsymbol{s}^{i-1} \circ \boldsymbol{s}$. For an index $i \in [n]$, let $b(i)$ denote its binary representation using $\lceil \log(n) \rceil$ bits, where the least significant bit is the last bit. Finally, the *redundancy* of a set $A \subseteq \mathbb{F}_2^n$ is defined as $\mathrm{red}(A) \triangleq n - \log|A|$, where all logarithms in this paper are taken to base 2.

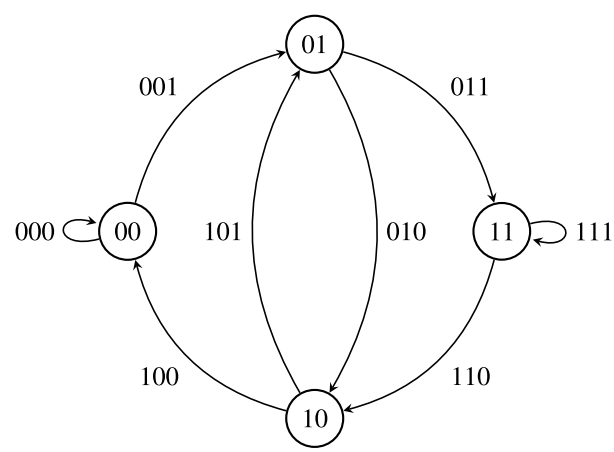
Definition 1: The ℓ-th order binary **de Bruijn graph** G_ℓ is the digraph (V, E), where $V = \mathbb{F}_2^{\ell-1}$ and

$$E = \{((s_0, s_1, \ldots, s_{\ell-2}), (s_1, s_2, \ldots, s_{\ell-1})) \,|\, s_i \in \mathbb{F}_2, i \in [\ell]\}.$$

An edge $((s_0, s_1, \ldots, s_{\ell-2}), (s_1, s_2, \ldots, s_{\ell-1})) \in E$ is denoted by the ℓ-tuple $(s_0, s_1, \ldots, s_{\ell-1})$.

Note that the edges of G_ℓ are associated with the set of binary ℓ-tuples, $\mathbb{F}_2^\ell$.

Example 1: The graph G_3 is given by



□

Definition 2: Let ℓ be an integer and $n = 2^\ell$. A cyclic sequence $\boldsymbol{s} \in \mathbb{F}_2^n$ is called a **de Bruijn sequence** of order ℓ if $\boldsymbol{s}$ contains each binary ℓ-tuple as a window exactly once.

The connection between Eulerian cycles in G_ℓ and de Bruijn sequences of order ℓ is as follows. In order to generate a sequence from a cycle, we pick any edge in the cycle to start

from and consider the first entry of each consecutive edge in the cycle.

Example 2: Let $\ell = 3, n = 8$. Then,

$$\boldsymbol{s} = [00010111]$$

is a de Bruijn sequence of order ℓ (in subsequent examples, we sometimes omit the parentheses from $\boldsymbol{s}$). The sequence $\boldsymbol{s}$ can be generated from G_3 using the Eulerian cycle

$$00 \overset{000}{\to} 00 \overset{001}{\to} 01 \overset{010}{\to} 10 \overset{101}{\to} 01 \overset{011}{\to} 11 \overset{111}{\to} 11 \overset{110}{\to} 10 \overset{100}{\to} 00.$$

□

Definition 3: Let $n \in \mathbb{N}$ be an integer and let $\boldsymbol{s} \in \mathbb{F}_2^n$ be a sequence. We say that $\boldsymbol{s}$ is **balanced** if $w_H(\boldsymbol{s}) = \lceil n/2 \rceil$.

Note that Definition 3 is applicable for both even and odd sequence lengths. When n is even, it is required that the numbers of *zeros* and *ones* are identical. When n is odd, the bit *one* appears one more time than the bit *zero*. Additionally, we define for $\boldsymbol{s}$ its *imbalance*, denoted by $\delta(\boldsymbol{s}) \triangleq \lceil n/2 \rceil - w_H(\boldsymbol{s})$. Note that $\boldsymbol{s}$ is balanced if and only if $\delta(\boldsymbol{s}) = 0$.

The redundancy of the set of length n balanced sequences is $0.5 \log n + \mathcal{O}(1)$ implied directly from Stirling's central binomial approximation [38] (see [28] for an accurate calculation). In [28], Knuth designed an efficient encoding algorithm that balances a sequence $\boldsymbol{x} \in \mathbb{F}_2^n$ using $\log n + \Theta(\log \log n)$ bits. The algorithm first finds a *balancing index* i of $\boldsymbol{x}$, which satisfies that flipping $\text{Pref}_i(\boldsymbol{x})$, that is, replacing it with its binary compliment, results in a balanced sequence. Then, the binary encoding of the index i is encoded as a balanced sequence using $\log n + \Theta(\log \log n)$ bits and it is appended to $\boldsymbol{x}$ after flipping the i-prefix of $\boldsymbol{x}$. Let $\mathcal{E}_K$ denote this encoder which will be utilized in several encoding algorithms in this paper.

Example 3: Let $n = 16$. Then, the sequence

$$\boldsymbol{s} = 1011000000000111$$

is not balanced since its imbalance is $\delta(\boldsymbol{s}) = 8 - 6 = 2$. In order to balance the sequence, we follow Knuth's balancing algorithm and iterate $i = 0, \ldots, n-1$ until we find that flipping the i-prefix of $\boldsymbol{s}$ balances the sequence. This is satisfied by $i = 8$, and after flipping $\text{Pref}_i(\boldsymbol{s})$ we obtain

$$\hat{\boldsymbol{s}} = 0100111100000111.$$

Next, in order to be able to decode $\boldsymbol{s}$, we need to append to $\hat{\boldsymbol{s}}$ a balanced binary encoding of the index $i = 8$ using $\log n + \Theta(\log \log n)$ bits. There are several known methods to do so, we use for this example a simple approach that requires exactly $\log n + 2 \log \log n$ redundancy bits. We take the binary encoding $\boldsymbol{v} = b(i) = 1000$ of length $\log n$ and balance it using the same method. We obtain the balanced sequence $\hat{\boldsymbol{v}} = 0110$ and the balancing index $j = 3$. Then, we take the binary encoding $\boldsymbol{u} = b(j) = 11$ of length $\log \log n$ and concatenate it with its binary compliment to receive $\hat{\boldsymbol{u}} = \boldsymbol{u} \circ \bar{\boldsymbol{u}} = 1100$, a balanced sequence of length $2 \log \log n$. Finally, we concatenate these sequences to receive a unique balanced sequence,

$$\boldsymbol{s}^* = \hat{\boldsymbol{s}} \circ \hat{\boldsymbol{v}} \circ \hat{\boldsymbol{u}} = 010011110000011101101100.$$

□

We define the main family of sequences discussed in this paper, which induces a hierarchy of balanced sequences. This hierarchy classifies balanced sequences based on the multiplicity of the appearances of ℓ-tuples in the sequence; it will be shown that this property has a strong connection to de Bruijn graph of order ℓ.

Definition 4: Let ℓ, μ be integers, and $n = \mu 2^\ell$. A cyclic sequence $\boldsymbol{s} \in \mathbb{F}_2^n$ is called an (ℓ, μ)**-balanced de Bruijn** sequence, or an (ℓ, μ)**-BdB** sequence in short, if $\boldsymbol{s}$ contains each binary ℓ-tuple as a window exactly μ times, i.e., for every $\boldsymbol{v} \in \mathbb{F}_2^\ell$, there exist exactly μ distinct indices $i_0, \ldots, i_{\mu-1} \in [n]$ such that $\boldsymbol{s}_{i_j,\ell} = \boldsymbol{v}$ for every $j \in [\mu]$.

Example 4: Let $\ell = 3, \mu = 2, n = 16$. Then, the sequence

$$\boldsymbol{s} = 0000100101101111$$

is a $(3, 2)$-BdB sequence, since it contains each binary 3-tuple as a window exactly twice. □

We denote the set of all (ℓ, μ)-BdB sequences over $\mathbb{F}_2$ by $\mathcal{B}(\ell, \mu)$. We also define the asymptotic rate of $\mathcal{B}(\ell, \mu)$, $\mathbb{R}(\mu)$, as a function of the multiplicity μ by

$$\mathbb{R}(\mu) \triangleq \limsup_{\ell \to \infty} \frac{\log |\mathcal{B}(\ell, \mu)|}{\mu 2^\ell}.$$

Observe that when $\mu = 1$ the set $\mathcal{B}(\ell, 1)$ is exactly the set of de Bruijn sequences, and hence $\mathbb{R}(1) = 1/2$. On the other hand, for $\ell = 1$, the set $\mathcal{B}(1, \mu = n/2)$ is the set of balanced sequences of length n, and thus

$$\text{red}(\mathcal{B}(1, n/2)) = 0.5 \log n + \mathcal{O}(1).$$

Next, we present the useful definitions of the derivative sequence and the integral sequence. These mappings were defined before by Lempel [31] as the D-morphism and its inverse D^{-1}, respectively, and have since been applied in various papers, see for example [4], [5], [8], [13], [14], [17], [18], [19], [39], [40].

Definition 5: Let $\boldsymbol{s} \in \mathbb{F}_2^n$ be a cyclic sequence. Its derivative sequence, denoted by $\boldsymbol{s}'$, is the cyclic sequence

$$\boldsymbol{s}' = [s_0 + s_1, s_1 + s_2, \ldots, s_{n-2} + s_{n-1}, s_{n-1} + s_0] \in \mathbb{F}_2^n.$$

The derivative of an acyclic sequence $\boldsymbol{y} \in \mathbb{F}_2^n$ is an acyclic sequence of length $n - 1$,

$$\boldsymbol{y}' = (y_0 + y_1, y_1 + y_2, \ldots, y_{n-2} + y_{n-1}).$$

Notice that for every cyclic sequence $\boldsymbol{s} \in \mathbb{F}_2^n$, the Hamming weight of $\boldsymbol{s}'$ is even, since

$$\begin{aligned} w_H(\boldsymbol{s}') \bmod 2 &= \sum_{i=0}^{n-2} (s_i + s_{i+1}) \bmod 2 + (s_{n-1} + s_0) \bmod 2 \\ &= \sum_{i=0}^{n-1} 2\ s_i \bmod 2 \\ &= 0 \bmod 2. \end{aligned}$$

Additionally, note that $\bar{\boldsymbol{s}}$, the binary compliment of $\boldsymbol{s}$, satisfies that $(\bar{\boldsymbol{s}})' = \boldsymbol{s}'$, and no other sequence of $\mathbb{F}_2^n$ shares this derivative. The same holds for any acyclic sequence (we use later the notation $\bar{\boldsymbol{s}}$ for acyclic sequences as well).

TABLE I
TABLE OF DEFINITIONS AND NOTATIONS

Notation	Meaning	Remarks
n	Sequence length	Sec. II
$\mathbb{F}_2$	The binary alphabet $\{0,1\}$	Sec. II
$[k] = [0, k-1]$	The set $\{0,\ldots,k-1\}$	Sec. II
$\boldsymbol{s} = [s_0,\ldots,s_{n-1}]$	A cyclic sequence of length n	Sec. II
$\boldsymbol{y} = (y_0,\ldots,y_{n-1})$	An acyclic sequence of length n	Sec. II
$\boldsymbol{s}_{i,k}$	An acyclic window of $\boldsymbol{s}$	Sec. II
$\mathrm{Pref}_k(\boldsymbol{s})$, $\mathrm{Suff}_k(\boldsymbol{s})$	k-prefix, k-suffix of $\boldsymbol{s}$	Sec. II
$w_H(\boldsymbol{s})$	Hamming weight of $\boldsymbol{s}$	Sec. II
$p(\boldsymbol{s})$	Period of $\boldsymbol{s}$	Sec. II
$b(i)$	Binary representation of index i	Sec. II
$\mathrm{red}(A)$	Redundancy of set A	Sec. II
G_ℓ	The ℓ-th order binary de Bruijn graph	Def. 1
$\delta(\boldsymbol{s})$	Imbalance of $\boldsymbol{s}$	Sec. II
$\mathcal{E}_K$	Knuth's balanced sequences encoder	Def. 3
$\mathcal{B}(\ell,\mu)$	The set of (ℓ,μ)-BdB sequences	Def. 4
$\mathbb{R}(\mu)$	Asymptotic rate of $\mathcal{B}(\ell,\mu)$	Def. 4
$\boldsymbol{s}'$	Derivative sequence of $\boldsymbol{s}$	Def. 5
$\int(\boldsymbol{s},\sigma)$	Integral sequence of $\boldsymbol{s},\sigma$	Def. 6
$G_{\ell,\mu}$	Generalized de Bruijn graph	Sec. IV
$\boldsymbol{r}$	The all-zero vertex of $G_{\ell,\mu}$	Sec. IV
T	A reverse spanning tree of $G_{\ell,\mu}$	Sec. IV
$\mathcal{D}(n,k)$	The set of k-order balanced sequences	Sec. VI

Definition 6: Let $\boldsymbol{s} \in \mathbb{F}_2^n$ be a cyclic sequence where its Hamming weight is even, and let $\sigma \in \mathbb{F}_2^n$ be a symbol. The **integral sequence** derived from $\boldsymbol{s},\sigma$, denoted by $\int(\boldsymbol{s},\sigma)$, is the sequence

$$\int(\boldsymbol{s},\sigma) = [\widehat{s}_0,\ldots,\widehat{s}_{n-1}] \in \mathbb{F}_2^n,$$

where $\widehat{s}_0 = \sigma$ and for every $i \in [1, n-1]$, $\widehat{s}_i = s_{i-1} + \widehat{s}_{i-1}$. Notice that $\boldsymbol{s} = \int(\boldsymbol{s}', s_0)$. Moreover, notice that every cyclic sequence of length n with even Hamming weight is the derivative of two unique cyclic sequences of length n.

Example 5: Let $n = 12$ and $\boldsymbol{s} = 100001000001$. The derivative sequence $\boldsymbol{s}'$ is

$$\boldsymbol{s}' = 100011000010.$$

Additionally,

$$\int(\boldsymbol{s}', 1) = 100001000001 = \boldsymbol{s},$$

and,

$$\int(\boldsymbol{s}', 0) = 011110111110 = \bar{\boldsymbol{s}}.$$

□

Finally, for the convenience of the reader, relevant notations and terminology referred to throughout the paper are summarized in Table I.

III. Encoding $(2, n/4)$-BdB Sequences Using Knuth's Algorithm

First, we present an encoding algorithm for (ℓ,μ)-BdB sequences of length $n = 4\mu$ with $\ell = 2$, which is based on the connection of this set to the family of balanced sequences.

We seek to extend Knuth's algorithm [28], which encodes an arbitrary binary sequence to a balanced one using $\log n + \Theta(\log\log n)$ redundancy bits, for the case where each 2-tuple appears exactly $n/4$ times. The following lemma characterizes this set of sequences.

Lemma 7: If $\boldsymbol{s} \in \mathbb{F}_2^n$, then $\boldsymbol{s}$ is a $(2, n/4)$-BdB sequence if and only if $\boldsymbol{s}$ is balanced and $\boldsymbol{s}'$ is balanced.

Proof: First, assume that $\boldsymbol{s} \in \mathcal{B}(2, n/4)$. Since each 2-tuple in $\mathbb{F}_2^2 = \{00, 01, 10, 11\}$ appears $n/4$ times in $\boldsymbol{s}$, it follows that $\boldsymbol{s}$ is balanced. Moreover, this implies that each 2-tuple of $\boldsymbol{s}$ yields $n/2$ occurrences of the bit *zero* in $\boldsymbol{s}'$ and $n/2$ occurrences of the bit *one* in $\boldsymbol{s}'$. Hence, $\boldsymbol{s}'$ is balanced.

For the other direction, assume that $\boldsymbol{s}$ and $\boldsymbol{s}'$ are balanced, and let $a_{00}, a_{01}, a_{10}, a_{11}$ be variables such that for every $\boldsymbol{y} \in \mathbb{F}_2^2$, $a_{\boldsymbol{y}}$ denotes the number of occurrences of the 2-tuple $\boldsymbol{y}$ as a window of $\boldsymbol{s}$. First, notice that for any sequence we have that

$$a_{01} = a_{10}. \tag{1}$$

Since $\boldsymbol{s}$ is balanced, it has an equal number of *zeros* and *ones*. It follows that

$$2\ a_{00} + a_{01} + a_{10} = 2\ a_{11} + a_{01} + a_{10} \Rightarrow a_{00} = a_{11}. \tag{2}$$

Additionally, since $\boldsymbol{s}'$ is balanced, it has an equal number of *zeros* and *ones* as well. Derivating the 2-tuples 00 or 11 induces an occurrence of *zero* in $\boldsymbol{s}'$, and similarly derivating the 2-tuples 01 or 10 induces an occurrence of *one* in $\boldsymbol{s}'$. Therefore, we have

$$a_{00} + a_{11} = a_{01} + a_{10}. \tag{3}$$

By combining equations (1), (2), and (3) we have $a_{00} = a_{01} = a_{10} = a_{11} = n/4$. ■

Notice that by flipping some i-prefix of a sequence $\boldsymbol{s}$, the new derivative is changed by exactly two bits compared to $\boldsymbol{s}'$ (at positions i and n) unless $i \in \{0, n\}$. Thus, the imbalance of the new derivative is changed by at most 2 compared to $\boldsymbol{s}'$. Additionally, denote the sequences $\omega_{-1} = 0111, \omega_0 = 0011, \omega_1 = 0001$ and notice that $\delta(\omega_{-1}) = -1$, $\delta(\omega_0) = 0, \delta(\omega_1) = 1$.

Algorithm 1 receives an input sequence $\boldsymbol{s} \in \mathbb{F}_2^{n-\rho}$ where $\rho = 2\log n + \Theta(\log\log n)$, and outputs a unique sequence $\widehat{\boldsymbol{x}} \in \mathcal{B}(2, n/4)$. The algorithm first uses Knuth's algorithm to balance $\boldsymbol{s}'$ and obtain a sequence $(\boldsymbol{s}^*)'$ as a result. Then, Knuth's algorithm is applied on the integral $\boldsymbol{s}^* = \int((\boldsymbol{s}^*)', s_0)$ and receives $\widehat{\boldsymbol{s}}$ as a result. The balancing indices of $\boldsymbol{s}'$ and $\boldsymbol{s}^*$ are concatenated to construct the indices sequence $\boldsymbol{v}$ of length[1] $2\log n$. Then, the process is repeated with $\boldsymbol{v}$ as an input to create $\widehat{\boldsymbol{v}}$ and its indices sequence $\boldsymbol{u}$ of length $2\log(2\log n)$. Since $\boldsymbol{u}$ is relatively a short sequence, it can be transformed to a balanced sequence with a balanced derivative, $\widehat{\boldsymbol{u}}$, using a fixed encoding lookup table, denoted by LT_n, with $2^{2\log(2\log n)} = \Theta(\log^2 n)$ elements. It will be proved that for every $\boldsymbol{u}$, $2\log\log n + \Theta(\log\log\log n)$ bits are sufficient for $\widehat{\boldsymbol{u}}$. Later, we set $\boldsymbol{x} = \omega_0 \circ \widehat{\boldsymbol{s}} \circ \widehat{\boldsymbol{v}} \circ \widehat{\boldsymbol{u}}$ to receive a balanced sequence that starts with a *zero* and its derivative has a constant imbalance. Finally, the derivative of $\boldsymbol{x}$, $\boldsymbol{x}'$, is balanced manually to receive a sequence $\widehat{\boldsymbol{x}}'$ by appending a sequence of constant predetermined length m_1. Then, the integral $\widehat{\boldsymbol{x}} = \int(\widehat{\boldsymbol{x}}', x_0)$ is balanced manually by appending

[1]For simplicity, in the rest of this paper we omit some of the ceiling notations.

a sequence of constant predetermined length m_0. The lengths m_0, m_1 are known in advance independently of n as explained later after Lemma 12. The result $\widehat{\boldsymbol{x}}$ is a balanced sequence with a balanced derivative of length n and hence belongs to $\mathcal{B}(2, n/4)$ by Lemma 7.

Remark 1: Throughout Algorithm 1, when applying the integral operation $\int(\boldsymbol{y}, \sigma)$, for a sequence $\boldsymbol{y}$ and a symbol σ, $\boldsymbol{y}$ is always balanced and its length is a multiple of 4; therefore, the Hamming weight of $\boldsymbol{y}$ is even and it is a valid input for the integral operation as discussed after Definition 5.

Algorithm 1 Encoding $(2, n/4)$-BdB Sequences

Input: A sequence $\boldsymbol{s} \in \mathbb{F}_2^{n-\rho}$
Output: A sequence $\widehat{\boldsymbol{x}} \in \mathcal{B}(2, n/4)$
Additional Ingredients: Knuth's algorithm $\mathcal{E}_K$, redundancy $\rho = 2\log n + \Theta(\log\log n)$, lookup table LT_n, sequences $\omega_{-1}, \omega_0, \omega_1$, constant predetermined values m_0, m_1.

1: Derivate $\boldsymbol{s}$ to receive $\boldsymbol{s}'$.
2: Use $\mathcal{E}_K$ to find i_1, the balancing index of $\boldsymbol{s}'$. Flip the i_1-prefix of $\boldsymbol{s}'$ to receive $(\boldsymbol{s}^*)'$ and obtain $\boldsymbol{s}^* = \int((\boldsymbol{s}^*)', s_0)$.
3: Use $\mathcal{E}_K$ to find i_0, the balancing index of $\boldsymbol{s}^*$, and flip its i_0-prefix to receive $\widehat{\boldsymbol{s}}$.
4: Construct the indices sequence $\boldsymbol{v} = b(i_0) \circ b(i_1)$ of length $2\log n$.
5: Repeat Steps 1-4 with $\boldsymbol{v}$ as an input to receive $\widehat{\boldsymbol{v}}$ and the indices sequence $\boldsymbol{u}$ of length $2\log(2\log n)$.
6: Use the lookup table to receive $\widehat{\boldsymbol{u}} = LT_n(\boldsymbol{u})$, a balanced sequence with a balanced derivative of length $2\log\log n + \Theta(\log\log\log n)$.
7: Let $\boldsymbol{x} = \omega_0 \circ \widehat{\boldsymbol{s}} \circ \widehat{\boldsymbol{v}} \circ \widehat{\boldsymbol{u}}$.
8: Derivate $\boldsymbol{x}$ to receive $\boldsymbol{x}'$, and let $g = \delta(\boldsymbol{x}')$.
9: **if** $g \geqslant 0$ **then**
10: Set $\widehat{\boldsymbol{x}'} = \boldsymbol{x}' \circ 1^{2g}(01)^{\frac{m_1}{2}-g}$.
11: **else**
12: Set $\widehat{\boldsymbol{x}}' = \boldsymbol{x}' \circ 0^{2|g|}(01)^{\frac{m_1}{2}-|g|}$.
13: **end if**
14: Obtain $\widehat{\boldsymbol{x}} = \int(\widehat{\boldsymbol{x}}', x_0)$, and let $h = \delta(\widehat{\boldsymbol{x}})$.
15: **if** $h \geqslant 0$ **then**
16: Set $\widehat{\boldsymbol{x}} = \widehat{\boldsymbol{x}} \circ \omega_{-1}^{h} \circ \omega_0^{\frac{m_0}{4}-h}$.
17: **else**
18: Set $\widehat{\boldsymbol{x}} = \widehat{\boldsymbol{x}} \circ \omega_1^{|h|} \circ \omega_0^{\frac{m_0}{4}-|h|}$.
19: **end if**
20: Return $\widehat{\boldsymbol{x}}$.

Example 6: Let $n = 184$, $\rho = 44$. The input sequence for Algorithm 1 is

$$\boldsymbol{s} = 1^8 0^{80} 1^{52},$$

of length $n - \rho = 140$. Its derivative sequence is computed in Step 1,

$$\boldsymbol{s}' = 0^7 1 0^{79} 1 0^{52}.$$

Notice that both $\boldsymbol{s}$ and $\boldsymbol{s}'$ are not balanced. At Step 2, the algorithm finds the balancing index of $\boldsymbol{s}'$, $i_1 = 70$, and flips the i_1-prefix of $\boldsymbol{s}'$ to receive the derivative of $\boldsymbol{s}^*$,

$$(\boldsymbol{s}^*)' = 1^7 0 1^{62} 0^{17} 1 0^{52}.$$

TABLE II
LOOKUP TABLE OF ALGORITHM 1

n	$\lvert\boldsymbol{u}\rvert$	$\lvert\widehat{\boldsymbol{u}}\rvert$	Table size
$n \in [2^2+1, 2^8]$	8	12	256
$n \in [2^8+1, 2^{32}]$	12	16	4096
$n \in [2^{32}+1, 2^{64}]$	12	16	4096
$n \in [2^{64}+1, 2^{128}]$	16	20	65536

Then, $\boldsymbol{s}^*$ is constructed from $(\boldsymbol{s}^*)'$ and s_0,

$$\boldsymbol{s}^* = \int((\boldsymbol{s}^*)', s_0) = (10)^4 (01)^{31} 0^{18} 1^{52}.$$

At Step 3, the algorithm finds the balancing index $i_0 = 123$ and flips the i_0-prefix of $\boldsymbol{s}^*$ to get

$$\widehat{\boldsymbol{s}} = (01)^4 (10)^{31} 1^{18} 0^{35} 1^{17}.$$

This induces an indices sequence $\boldsymbol{v}$, computed at Step 4,

$$\boldsymbol{v} = b(123) \circ b(70) = 01111011 \circ 01000110.$$

Next, at Step 5 the process is repeated for $\boldsymbol{v}$ and we receive

$$\widehat{\boldsymbol{v}} = 1001101101000110, \text{ and } \boldsymbol{u} = 00110000.$$

Then, at Step 6, $\boldsymbol{u}$ is encoded into a balanced sequence with a balanced derivative of length 12 according to its lexicographic order in the fixed encoding lookup table LT_n (see lookup table sizes in Table II),

$$\widehat{\boldsymbol{u}} = 001001110011.$$

Then, at Step 7 we have

$$\begin{aligned}\boldsymbol{x} &= 0011 \circ (01)^4 (10)^{31} 1^{18} 0^{35} 1^{17} \circ 1001101101000110 \circ 001001110011 \\ &= 0^2 1^2 (01)^4 (10)^{31} 1^{18} 0^{35} 1^{18} 0^2 1^2 01^2 010^3 1^2 0^3 10^2 1^3 0^2 1^2,\end{aligned}$$

and derivatives it at Step 8 is

$$\boldsymbol{x}' = (01)^2 1^7 01^{62} 0^{17} 10^{34} 10^{17} (10)^2 1^2 01^3 0^2 (10)^2 01^2 010^2 (10)^2 1.$$

Next, manual balancing is done at lines 9 through 20 of the algorithm. For this example, we can take $m_0 = 8, m_1 = 4$ (the actual computed values of the constants m_0, m_1 are given in later). Since $g = \delta(\boldsymbol{x}') = -2$ is computed at Step 9, the algorithm appends 0^4 to $\boldsymbol{x}'$ at Step 12 and hence

$$\widehat{\boldsymbol{x}}' = (01)^2 1^7 01^{62} 0^{17} 10^{34} 10^{17} (10)^2 1^2 01^3 0^2 (10)^2 01^2 010^2 (10)^3 0^3.$$

At Step 14 we have, $\widehat{\boldsymbol{x}} = \int(\widehat{\boldsymbol{x}}', x_0)$ and therefore,

$$\widehat{\boldsymbol{x}} = 0^2 1^2 (01)^4 (10)^{31} 1^{18} 0^{35} 1^{18} 0^2 1^2 01^2 010^3 1^2 0^3 10^2 1^3 0^2 1^2 0^4.$$

Finally, $\widehat{\boldsymbol{x}}$ is manually balanced at Step 16 by appending $\omega_{-1}^2 = (0111)^2$ since $h = \delta(\widehat{\boldsymbol{x}}) = 2$, and the result is

$$\widehat{\boldsymbol{x}} = 0^2 1^2 (01)^4 (10)^{31} 1^{18} 0^{35} 1^{18} 0^2 1^2 01^2 010^3 1^2 0^3 10^2 1^3 0^2 1^2 0^5 1^3 01^3.$$

One can verify that each of $\{00, 01, 10, 11\}$ appears $n/4 = 46$ times as a window of $\widehat{\boldsymbol{x}}$. □

The following auxiliary lemmas will be used to prove the correctness of Algorithm 1.

Lemma 8: Let $\boldsymbol{y} \in \mathbb{F}_2^n, \boldsymbol{z} \in \mathbb{F}_2^m$ be sequences with balanced derivatives. If $y_0 = z_0$, then $(\boldsymbol{y} \circ \boldsymbol{z})'$ is balanced.

Proof: We have

$$\begin{aligned}(\boldsymbol{y}\circ\boldsymbol{z})' &= \mathrm{Pref}_{n-1}(\boldsymbol{y}')\circ(y_{n-1}+z_0)\circ\mathrm{Pref}_{m-1}(\boldsymbol{z}')\circ(z_{m-1}+y_0)\\ &= \mathrm{Pref}_{n-1}(\boldsymbol{y}')\circ(y_{n-1}+y_0)\circ\mathrm{Pref}_{m-1}(\boldsymbol{z}')\circ(z_{m-1}+z_0)\\ &= \boldsymbol{y}'\circ\boldsymbol{z}'\end{aligned}$$

Since $\boldsymbol{y}', \boldsymbol{z}'$ are balanced, if follows that $\boldsymbol{y}'\circ\boldsymbol{z}' = (\boldsymbol{y}\circ\boldsymbol{z})'$ is a balanced sequence. ■

Claim 9: Let $\boldsymbol{y}\in\mathbb{F}_2^n, \boldsymbol{z}\in\mathbb{F}_2^m$. Then,

$$|\delta((\boldsymbol{y}\circ\boldsymbol{z})')| \leqslant |\delta(\boldsymbol{y}')| + |\delta(\boldsymbol{z}')| + 2.$$

Proof: Similarly to the proof of Lemma 8, we can write

$$(\boldsymbol{y}\circ\boldsymbol{z})' = \mathrm{Pref}_{n-1}(\boldsymbol{y}')\circ(y_{n-1}+z_0)\circ\mathrm{Pref}_{m-1}(\boldsymbol{z}')\circ(z_{m-1}+y_0).$$

If $y_0 = z_0$, it follows immediately that $(\boldsymbol{y}\circ\boldsymbol{z})' = \boldsymbol{y}'\circ\boldsymbol{z}'$ and the claim holds. Otherwise, if $y_0 \neq z_0$, by observing all possible values of $y_0, y_{n-1}, z_0, z_{m-1}$ we have the weight of $(\boldsymbol{y}\circ\boldsymbol{z})'$ might differ by 2 from the sum of the weights of $\boldsymbol{y}'$ and $\boldsymbol{z}'$, and the claim holds as well. ■

Lemma 10: At the beginning of Step 9 of Algorithm 1, $\boldsymbol{x}$ is balanced and the imbalance of its derivative satisfies $|\delta(\boldsymbol{x}')| \leqslant 8$.

Proof: First, $\boldsymbol{x} = \omega_0\circ\widehat{\boldsymbol{s}}\circ\widehat{\boldsymbol{v}}\circ\widehat{\boldsymbol{u}}$ is balanced since it is a concatenation of the balanced sequences $\omega_0, \widehat{\boldsymbol{s}}, \widehat{\boldsymbol{v}}, \widehat{\boldsymbol{u}}$ (the sequences $\widehat{\boldsymbol{s}}, \widehat{\boldsymbol{v}}, \widehat{\boldsymbol{u}}$ were obtained in Steps 3, 5 and 6 respectively). Next, notice that at Step 3, the imbalance of $\widehat{\boldsymbol{s}}'$ satisfies $|\delta(\widehat{\boldsymbol{s}}')| \leqslant 2$ since flipping a prefix of $\widehat{\boldsymbol{s}}$ changes the imbalance of its derivative by at most 2. Similarly, at Step 5, the imbalance of $\widehat{\boldsymbol{v}}'$ satisfies $|\delta(\widehat{\boldsymbol{v}}')| \leqslant 2$. The deriative $\widehat{\boldsymbol{u}}'$ is balanced since it is obtained from the encoding lookup table at Step 6. Notice that at least two out of $\widehat{\boldsymbol{s}}, \widehat{\boldsymbol{v}}, \widehat{\boldsymbol{u}}$ start with the same bit. By Claim 9 we have

$$\begin{aligned}|\delta(\boldsymbol{x}')| &\leqslant |\delta(\omega_0')| + |\delta(\widehat{\boldsymbol{s}}')| + |\delta(\widehat{\boldsymbol{v}}')| + |\delta(\widehat{\boldsymbol{u}}')| + 2\\ &\leqslant 0 + 2 + 2 + 0 + 2\cdot 2\\ &\leqslant 8.\end{aligned}$$

■

Lemma 11: At the beginning of Step 15 of Algorithm 1, $\widehat{\boldsymbol{x}}$ satisfies that $|\delta(\widehat{\boldsymbol{x}})| \leqslant m_1$ and its derivative is balanced.

Proof: First, by appending to $\boldsymbol{x}'$ at Step 10 or 12 a sequence of length m_1, the imbalance of $\widehat{\boldsymbol{x}} = \int(\widehat{\boldsymbol{x}}', x_0)$ satisfies

$$\begin{aligned}\delta(\widehat{\boldsymbol{x}}) &= |\widehat{\boldsymbol{x}}|/2 - w_H(\widehat{\boldsymbol{x}})\\ &\leqslant (|\boldsymbol{x}| + m_1)/2 - (w_H(\boldsymbol{x}) + m_1)\\ &= \delta(\boldsymbol{x}) + m_1.\end{aligned}$$

Since by Lemma 10 before these steps $\boldsymbol{x}$ is balanced, we have $|\delta(\widehat{\boldsymbol{x}})| \leqslant m_1$.

As for the derivative, we have that if $g = \delta(\boldsymbol{x}') \geqslant 0$, and by Lemma 10 that $g \leqslant 6$, then if $m_1 \geqslant 2g$, the imbalance of $\widehat{\boldsymbol{x}}'$ is

$$\delta(\widehat{\boldsymbol{x}}') = g + \delta(1^{2g}(01)^{\frac{m_1}{2}-|g|}) = g - g = 0.$$

Similarly, $\widehat{\boldsymbol{x}}'$ is balanced if $g < 0$. ■

Lemma 12: At Step 20 of Algorithm 1, $\widehat{\boldsymbol{x}}$ and $\widehat{\boldsymbol{x}}'$ are balanced.

Proof: Remember that $\delta(\omega_1)=1, \delta(\omega_0)=0, \delta(\omega_{-1})=-1$. If $h = \delta(\widehat{\boldsymbol{x}}) \geqslant 0$, by Lemma 11 we have that $h \leqslant m_1$. Then, if $m_0 \geqslant 4h$, the imbalance of $\widehat{\boldsymbol{x}}$ is

$$\delta(\widehat{\boldsymbol{x}}) = h + \delta(\omega_{-1}^h \circ \omega_0^{\frac{m_0}{4}-|h|}) = h - h = 0.$$

Similarly, $\widehat{\boldsymbol{x}}$ is balanced if $h < 0$.

Next, notice that $\widehat{x}_0 = x_0 = 0$. Since $\widehat{\boldsymbol{x}}, \omega_1, \omega_0, \omega_{-1}$ have a balanced derivative and start with 0, it follows from Lemma 8 that $\widehat{\boldsymbol{x}}'$ remains balanced after the concatenation at Steps 16 or 18. ■

It follows from the proofs of Lemmas 11 and 12 that the constants m_0, m_1 should satisfy $m_0 \geqslant 4h$ and $m_1 \geqslant 2g$. Therefore, by Lemmas 10 and 11 the values $m_1 = 16$ and $m_0 = 64$ are sufficient for every n.

Theorem 13: Algorithm 1 returns a sequence $\widehat{\boldsymbol{x}} \in \mathcal{B}(2, n/4)$ that can be uniquely decoded to its input sequence $\boldsymbol{s}$. The algorithm uses $\rho = 2\log n + \Theta(\log\log n)$ redundancy bits and its time complexity is $\mathcal{O}(n)$.

Proof: The correctness of the output of the algorithm follows from Lemma 7 and Lemma 12. As for the redundancy, the algorithm adds $2\log n$ redundancy bits at Step 4, and additional $2\log\log n + \Theta(\log\log\log n)$ redundancy bits at Step 5. At Steps 7 to 20, the algorithm adds $m_1 + m_0 + 4 = \mathcal{O}(1)$ redundancy bits.

The time complexity of the algorithm follows from the time complexity of Knuth's algorithm, $\mathcal{O}(n)$, and the time complexity of the derivative and integral operations, $\Theta(n)$. The rest of the operations invoked throughout the algorithm have constant time complexity.

A decoder for Algorithm 1 recovers $\boldsymbol{x}$ from $\widehat{\boldsymbol{x}}$ by reversing Step 16 (or 18) and Step 10 (or 12) and trimming the last m_0, m_1 bits of $\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{x}}'$, respectively. After that, the decoder extracts $\widehat{\boldsymbol{s}}, \widehat{\boldsymbol{v}}, \widehat{\boldsymbol{u}}$ from $\boldsymbol{x}$, corresponding to Step 7, and recovers $\boldsymbol{u}$ from the fixed encoding table LT_n. Next, it recovers $\boldsymbol{v}$ from $\widehat{\boldsymbol{v}}$ using the balancing indices of $\boldsymbol{u}$ and similarly it recovers $\boldsymbol{s}$ from $\widehat{\boldsymbol{s}}$ using the balancing indices of $\boldsymbol{v}$. This process has the same time complexity as the encoder, $\mathcal{O}(n)$. ■

Remark 2: It will follow from Section IV that the redundancy of $\mathcal{B}(2, n/4)$ is $\log n + O(1)$; therefore, the length $2\log\log n + \Theta(\log\log\log n)$ is sufficient to encode $\widehat{\boldsymbol{u}}$ from $\boldsymbol{u}$ using LT_n at Step 6. Due to its small size, this encoding lookup table can be generated efficiently before invoking the algorithm, using an exhaustive search. Table II presents the lengths of $\boldsymbol{u}, \widehat{\boldsymbol{u}}$ and the size of LT_n for different values of n.

Unfortunately, it is not possible to generalize the ideas presented in this section for larger values of ℓ since the identity presented in Lemma 7 does not hold for $\ell > 2$; this subject will be discussed extensively in Section VI. Nonetheless, in Section V we present an encoding algorithm that uses a different technique and is able to efficiently encode (ℓ, μ)-BdB sequences for any $\ell = \mathcal{O}(\log\log n)$.

IV. Enumerating BdB Sequences Using a Generalization of De Bruijn Graph

In this section, we use a generalization of de Bruijn graph, in order to enumerate and encode the set of (ℓ, μ)-BdB sequences, i.e., $\mathcal{B}(\ell, \mu)$.

Let μ, ℓ be positive integers and denote $n = \mu 2^{\ell}$. We construct a digraph $G_{\ell,\mu} = (V, E_{\mu})$ based on de Bruijn graph G_{ℓ}. As in G_{ℓ}, the set of vertices V contains $2^{\ell-1}$ vertices represented by the binary $(\ell-1)$-tuples. The set E_{μ} has $\mu 2^{\ell}$ edges represented by the binary ℓ-tuples. The edges are the same as in G_{ℓ}, but between any two vertices which have an edge in G_{ℓ}, E_{μ} has μ parallel edges. Note that each Eulerian cycle in $G_{\ell,\mu}$ is associated with a unique (ℓ,μ)-BdB sequence. Therefore, in order to enumerate the number of (ℓ,μ)-BdB sequences we present next an algorithm that generates Eulerian cycles in $G_{\ell,\mu}$; this algorithm follows a known technique of generating an Eulerian cycle in any graph, see [27], [36], [37], [43].

Let $\boldsymbol{r} = 0^{\ell-1}$ denote the all-zero vertex of the graph $G_{\ell,\mu}$ for the rest of this section. We say that an *ordinary spanning tree* of $G_{\ell,\mu}$ is a graph that contains all the vertices of $G_{\ell,\mu}$, and has a unique directed path from $\boldsymbol{r}$ to each of its vertices. Similarly, a *reverse spanning tree* of $G_{\ell,\mu}$ is a graph that contains all the vertices of $G_{\ell,\mu}$, and has a unique directed path from each of its vertices to $\boldsymbol{r}$. Note that in any graph, if we change the direction of the edges of a reverse spanning tree in the graph, we receive an ordinary spanning tree, and vice versa.

Lemma 14: For every $\ell, \mu > 0$, the number of reverse spanning trees of $G_{\ell,\mu}$ is $2^{2^{\ell-1}-\ell}$.

Proof: Clearly, the number of reverse spanning trees of $G_{\ell,\mu}$ is equal to the number of reverse spanning trees of G_{ℓ}. It is known from [36] and [51] that the number of ordinary spanning trees of G_{ℓ} is $2^{2^{\ell-1}-\ell}$.

For an ℓ-tuple $\boldsymbol{y} = (s_0, s_1 \ldots, s_{\ell-2}, s_{\ell-1}) \in \mathbb{F}^{\ell}$, we define its *reverse* by $\boldsymbol{y}^R \triangleq (s_{\ell-1}, s_{\ell-2} \ldots, s_1, s_0)$. Additionally, for a set of edges $S \subseteq E$ (E are the edges of G_{ℓ}) its reverse is given by $S^R \triangleq \{u^R \to v^R \mid v \to u \in S\}$. Note that $(s_0, \ldots, s_{\ell-2}) \to (s_1, \ldots, s_{\ell-1}) \in E$ if an only if $(s_{\ell-1}, \ldots, s_1) \to (s_{\ell-2}, \ldots, s_0) \in E$. For simplicity, we assume that ordinary and reverse spanning trees are defined by their edges.

Let T be an ordinary spanning tree of G_{ℓ}, and consider the set T^R. Let $v \in V$ be any vertex of G_{ℓ}. The vertex v^R belongs to V and has a unique directed path from $\boldsymbol{r}$ in T, $\boldsymbol{r} \to u_1 \to \cdots \to u_m \to v^R$. Therefore, the path $v \to {u_m}^R \to \cdots \to {u_1}^R \to \boldsymbol{r}$ (note that $\boldsymbol{r}^R = \boldsymbol{r}$) is a unique directed path from v to $\boldsymbol{r}$ in T^R. It follows that the set T^R stands for a reverse spanning tree of G_{ℓ}. Similarly, if we take any reverse spanning tree of G_{ℓ} and reverse its edges, we receive an ordinary spanning tree of G_{ℓ}. Therefore, we obtain that the number of reverse spanning trees of G_{ℓ} is $2^{2^{\ell-1}-\ell}$, equal to the number of ordinary spanning trees. ■

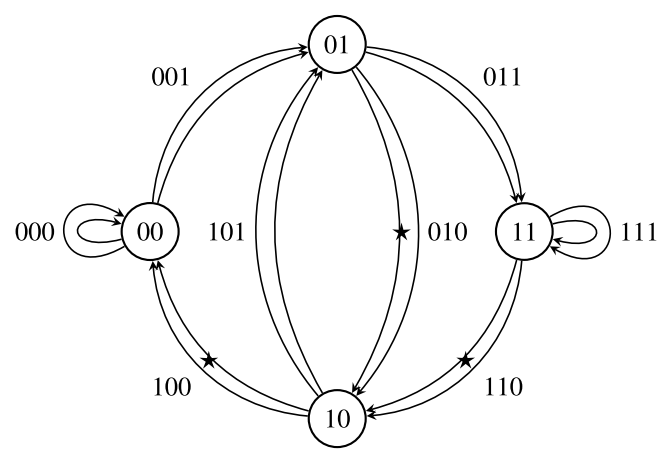
Algorithm 2 is applied to $G_{\ell,\mu}$ and a reverse spanning tree T of $G_{\ell,\mu}$. All the edges of T are notated as *starred*. The algorithm is nondeterministic and generates numerous different paths that visit each one of the edges of $G_{\ell,\mu}$ exactly once. First, all the edges are set to be *unmarked* at the beginning of the algorithm. When a vertex v of $G_{\ell,\mu}$ is visited, the edge on which v is left changes its status from unmarked to *marked*. In order to guarantee that a path can not be generated by two different trees, the algorithm leaves a vertex v on a starred edge only if it is the last unmarked edge left going out of v. Finally, for each generated path, the algorithm constructs its associated BdB sequence.

Algorithm 2 Generating a Sequence in $\mathcal{B}(\ell,\mu)$ From T

Input: A reverse spanning tree T of $G_{\ell,\mu}$
Output: A sequence $\boldsymbol{s} \in \mathcal{B}(\ell,\mu)$

Place a star on all the edges of T. All the edges of $G_{\ell,\mu}$ are set to be unmarked.
Set the current vertex $v := \boldsymbol{r}$.
while there is an unmarked edge going out of v **do**
  if v has an unstarred unmarked outgoing edge **then**
    Pick at random an unstarred edge $e = v \to u$.
    Leave v on the edge e to its endpoint u. Set $v := u$ and change the status of the edge e to marked.
  else
    Leave v on the starred edge, to its endpoint u. Set $v := u$ and change the status of the edge $v \to u$ to marked.
  end if
end while
Construct the output sequence $\boldsymbol{s}$ by considering the first bit of each edge of the path.

Example 7: Let $\ell = 3, \mu = 2$. The graph $G_{\ell,\mu}$ is



where the starred edges correspond to the reverse spanning tree T with the edges

$$\{010, 110, 100\}.$$

One of the paths generated by Algorithm 2 when invoked with T is

$$00 \overset{000}{\to} 00 \overset{000}{\to} 00 \overset{001}{\to} 01 \overset{010}{\to} 10 \overset{100}{\to} 00 \overset{001}{\to} 01 \overset{011}{\to} 11 \overset{111}{\to} 11 \overset{111}{\to} 11 \overset{110}{\to} 10 \overset{101}{\to} 01 \overset{011}{\to} 11 \overset{110\star}{\to} 10 \overset{101}{\to} 01 \overset{010\star}{\to} 10 \overset{100\star}{\to} 00,$$

which corresponds to the $(3,2)$-BdB sequence

$$\boldsymbol{s} = 0000100111101101.$$

□

The next few lemmas and corollaries describe the connection between paths generated by Algorithm 2 and sequences of $\mathcal{B}(\ell,\mu)$. Lemmas 15 and 16 are proved similarly as related

results in [36]; we give those lemmas here for the completeness of the paper.

Lemma 15: The path generated by Algorithm 2 ends at the root $\boldsymbol{r}$.

Proof: From the construction of $G_{\ell,\mu}$, the in-degree of each vertex is the same as its out-degree. Therefore, throughout the algorithm, when a vertex is visited it has one more unmarked outgoing edge than unmarked incoming edges. Hence, when a vertex is visited it has an outgoing edge which was not marked yet. The vertex $\boldsymbol{r}$ is the only vertex that after the algorithm starts has in-degree greater by one from the out-degree and hence the path must end at the vertex $\boldsymbol{r}$. ■

Lemma 16: When Algorithm 2 terminates all the edges of the graph were traversed, i.e., all the edges of $G_{\ell,\mu}$ are marked.

Proof: Assume to the contrary that there are edges which remain unmarked when the algorithm terminates. Assume that u_1 is a vertex for which there is an unmarked edge $u_1 \to u_2$. Since the path ends at the vertex $\boldsymbol{r}$ from Lemma 15, we must have $u_1 \neq \boldsymbol{r}$. W.l.o.g. we assume that $u_1 \to u_2$ is a starred edge, since for each vertex $v \neq \boldsymbol{r}$ there is exactly one outgoing starred edge, and it is the the last edge to be traversed out of v. Let $P = u_1 \to u_2 \cdots u_{m-1} \to u_m \to \boldsymbol{r}$ be the path from u_1 to $\boldsymbol{r}$ in the reverse spanning tree T.

Next, we prove using an induction that the starred edge $u_1 \to u_2$ was traversed, which will imply a contradiction to our initial assumption that it is unmarked. The induction will be obtained by iterating backwards the edges of the path P, and proving that each of them was in fact traversed by Algorithm 2. First, remind that from the construction of $G_{\ell,\mu}$, the in-degree of each vertex is equal to its out-degree. Since by Lemma 15 the algorithm ends at the root $\boldsymbol{r}$, all its outgoing edges were traversed, and therefore all its incoming edges were traversed as well, including the starred edge $u_m \to \boldsymbol{r}$. Next, for the induction step, assume that a starred edge $u_i \to u_{i+1}$ from the path P was traversed, where $u_{m+1} = \boldsymbol{r}$. Since the starred edge is the last edge traversed out of u_i, all its outgoing edges and therefore all its incoming edges were traversed, including the starred edge $u_{i-1} \to u_i$. ■

It follows from Lemma 16 that each reverse spanning tree T corresponds to some sequences of length $\mu 2^\ell$ in which each ℓ-tuple appears exactly μ times, i.e., an (ℓ,μ)-BdB sequence. Since the chosen root is $\boldsymbol{r} = 0^{\ell-1}$, it follows that every such a sequence starts with $\ell-1$ consecutive *zeros*.

Lemma 17: Each sequence of $\mathcal{B}(\ell,\mu)$ which starts with $\ell-1$ consecutive *zeros* is obtained by Algorithm 2 from exactly one reverse spanning tree.

Proof: Let $\boldsymbol{s} \in \mathcal{B}(\ell,\mu)$ be a sequence that starts with $\ell-1$ *zeros*, and observe the cycle in $G_{\ell,\mu}$ that corresponds to $\boldsymbol{s}$. For any vertex $v \neq \boldsymbol{r}$, its last appearance on the sequence $\boldsymbol{s}$ as an $(\ell-1)$-tuple, is related to an edge $v \to u$ in the graph $G_{\ell,\mu}$. Let S denote the set of these edges; we prove next that S stands for a reverse spanning tree of $G_{\ell,\mu}$. Let $v_1, \ldots, v_{2^{\ell-1}-1}$ denote these vertices ordered by the position of their last appearance in $\boldsymbol{s}$, and denote $v_{2^{\ell-1}} = \boldsymbol{r}$. It follows from the definition of S, the ordering of the vertices, and from the fact that the path corresponding to $\boldsymbol{s}$ ends at the root $\boldsymbol{r}$ (since it starts with $\ell-1$ *zeros*), that S contains only edges of the form $v_i \to v_j$ where $i < j$. Thus, given any vertex v_i, we can construct a unique path $v_i \to v_{j_1} \to v_{j_2} \cdots v_{j_{m-1}} \to v_{j_m}$ where $i < j_1 < j_2 < \cdots < j_{m-1} < j_m$, by iteratively traversing the single outgoing edge of each vertex. Since S contains no cycles, this path must end at the root $\boldsymbol{r}$.

Thus, $\boldsymbol{s}$ can be obtained by Algorithm 2 using the reverse spanning tree S. This tree is uniquely defined by the sequence and therefore $\boldsymbol{s}$ it is obtained exactly once by the algorithm. ■

Lemma 18: From each reverse spanning tree T, there are $2\binom{2\mu-1}{\mu-1}^{2^{\ell-1}}$ distinct sequences which are constructed by Algorithm 2.

Proof: By Lemma 16, each vertex v of $G_{\ell,\mu}$ is visited 2μ times during the algorithm, one time for each of the 2μ edges going out of v. On the last visit of each $v \neq \boldsymbol{r}$, the algorithm leaves the vertex on the unique starred edge going out of v. Let $v \to u_1$, $v \to u_2$ be the two types of edges going out of v, where $u_1 \neq u_2$. W.l.o.g. assume that $v \to u_1$ is a starred edge. Hence, on the first $2\mu-1$ times when v is visited, $\mu-1$ times the traversed edge is of the type $v \to u_1$ and μ times the traversed edge is of the type $v \to u_2$. Each different order in which these two types of edges are traversed, will form a different sequence since each edge corresponds to an ℓ-tuple and each such different order of traversed edges, induces a different order of the associated ℓ-tuples in the sequence. There are $\binom{2\mu-1}{\mu-1}$ different orders for these two types of edges to be traversed. When $v = \boldsymbol{r}$, all the 2μ edges going out of $\boldsymbol{r}$ are not starred, and hence there are $\binom{2\mu}{\mu} = 2\binom{2\mu-1}{\mu-1}$ different orders for its edges to be traversed. Finally, there are $2^{\ell-1}$ vertices in the graph which implies the statement of the lemma. ■

Combining Lemmas 17 and 18, and since by Lemma 14 the number of reversed spanning trees rooted at $\boldsymbol{r}$ in $G_{\ell,\mu}$ is $2^{2^{\ell-1}-\ell}$, the following result is immediately inferred.

Corollary 19: The total number of distinct sequences of $\mathcal{B}(\ell,\mu)$ formed by Algorithm 2 is $2^{2^{\ell-1}-\ell+1}\binom{2\mu-1}{\mu-1}^{2^{\ell-1}}$.

By Lemma 17, the enumerated value of Corollary 19 accounts only for sequences of $\mathcal{B}(\ell,\mu)$ that start with $\ell-1$ *zeros*. The next theorem gives the exact size of $\mathcal{B}(\ell,\mu)$.

Theorem 20: For every integers ℓ,μ, the size of $\mathcal{B}(\ell,\mu)$ is

$$|\mathcal{B}(\ell,\mu)| = 2^{2^{\ell-1}}\binom{2\mu-1}{\mu-1}^{2^{\ell-1}}.$$

Proof: Let $\boldsymbol{x} \in \mathcal{B}(\ell,\mu)$ be a sequence that starts with $\ell-1$ *zeros*, and consider its period $p(\boldsymbol{x})$. By the definition of a BdB sequence, the period $p(\boldsymbol{x})$ is $\lambda 2^\ell$ for some integer λ that divides μ. Note that there are exactly 2λ sequences which are cyclic shifts of $\boldsymbol{x}$, start with $\ell-1$ *zeros*, and are generated by Algorithm 2 by Lemma 17. These 2λ sequences are associated with exactly $\lambda 2^\ell$ sequences of $\mathcal{B}(\ell,\mu)$, since choosing each starting point in the $\lambda 2^\ell$-prefix of $\boldsymbol{x}$ corresponds to a different sequence. Therefore, multiplying the number of cyclic shifts of $\boldsymbol{x}$ that are outputs of the algorithm by $\lambda 2^\ell/2\lambda = 2^{\ell-1}$ yields the number of its cyclic shifts in $\mathcal{B}(\ell,\mu)$. Since by Corollary 19 the algorithm generates $2^{2^{\ell-1}-\ell+1}\binom{2\mu-1}{\mu-1}^{2^{\ell-1}}$ sequences that start with $\ell-1$ *zeros*, it follows that the number of sequences

in $\mathcal{B}(\ell,\mu)$ is multiplied by $2^{\ell-1}$. Thus,

$$|\mathcal{B}(\ell,\mu)| = 2^{2^{\ell-1}}\binom{2\mu-1}{\mu-1}^{2^{\ell-1}}.$$

■

Corollary 21: Let $\mu \in \mathbb{N}$ be a constant integer. The asymptotic rate of $\mathcal{B}(\ell,\mu)$ is

$$\mathbb{R}(\mu) = \frac{1+\log\binom{2\mu-1}{\mu-1}}{2\mu}.$$

Proof: We have,

$$\begin{aligned}
\mathbb{R}(\mu) &= \limsup_{\ell\to\infty}\frac{\log|\mathcal{B}(\ell,\mu)|}{\mu 2^\ell} && \text{(Definition of } \mathbb{R}(\mu)\text{)}\\
&= \limsup_{\ell\to\infty}\frac{\log\left(2^{2^{\ell-1}}\binom{2\mu-1}{\mu-1}^{2^{\ell-1}}\right)}{\mu 2^\ell} && \text{(Theorem 20)}\\
&= \limsup_{\ell\to\infty}\frac{2^{\ell-1}+2^{\ell-1}\log\binom{2\mu-1}{\mu-1}}{\mu 2^\ell} && \text{(Algebraic manipulation)}\\
&= \frac{1+\log\binom{2\mu-1}{\mu-1}}{2\mu}. && \text{(Algebraic manipulation)}
\end{aligned}$$

■

Next, we use the result of Theorem 20 in order to derive the redundancy of $\mathcal{B}(\ell,\mu)$ for different values of ℓ,μ.

Theorem 22: The redundancy of $\mathcal{B}(\ell,\mu)$ satisfies

$$\text{red}(\mathcal{B}(\ell,\mu)) = 2^{\ell-1}\left(0.5\log\mu + 0.5\log\pi - \log(1-\frac{1}{c_\mu\mu})\right),$$

where $8 \leqslant c_\mu \leqslant 9$. In particular, for a constant $\ell \in \mathbb{N}$ and $n = \mu 2^\ell$ for some integer μ, the redundancy of $\mathcal{B}(\ell, n/2^\ell)$ is

$$\text{red}(\mathcal{B}(\ell,n/2^\ell)) = 2^{\ell-2}\log n + \mathcal{O}(1).$$

Proof: Using Stirling's central binomial approximation formula [38], we have that

$$\binom{2\mu}{\mu} = \frac{2^{2\mu}}{\sqrt{\mu\pi}}(1-\frac{1}{c_\mu\mu}),$$

for $8 \leqslant c_\mu \leqslant 9$. Additionally, since $\binom{2\mu-1}{\mu-1} = \frac{1}{2}\binom{2\mu}{\mu}$, we can write

$$\log\binom{2\mu-1}{\mu-1} = 2\mu - 1 - 0.5\log\mu - 0.5\log\pi + \log(1-\frac{1}{c_\mu\mu}). \tag{4}$$

Thus,

$$\begin{aligned}
&\text{red}(\mathcal{B}(\ell,\mu))\\
&= \mu 2^\ell - \log|\mathcal{B}(\ell,\mu)| && \text{(Definition of redundancy)}\\
&= \mu 2^\ell - \log\left(2^{2^{\ell-1}}\binom{2\mu-1}{\mu-1}^{2^{\ell-1}}\right) && \text{(Theorem 20)}\\
&= \mu 2^\ell - 2^{\ell-1}\left(2\mu - 0.5\log\mu - 0.5\log\pi + \log(1-\frac{1}{c_\mu\mu})\right) && \text{(Eq. (4))}\\
&= 2^{\ell-1}\left(0.5\log\mu + 0.5\log\pi - \log(1-\frac{1}{c_\mu\mu})\right). && \text{(Algebraic manipulation)}
\end{aligned}$$

■

TABLE III
ASYMPTOTIC RATES OF $\mathcal{B}(\ell,\mu)$ FOR FIXED μ

Multiplicity	Rate
μ = 1 (de Bruijn)	0.5
μ = 2	0.646
μ = 4	0.766
μ = 8	0.853
μ = 16	0.911
μ = 32	0.947
μ = 64	0.971

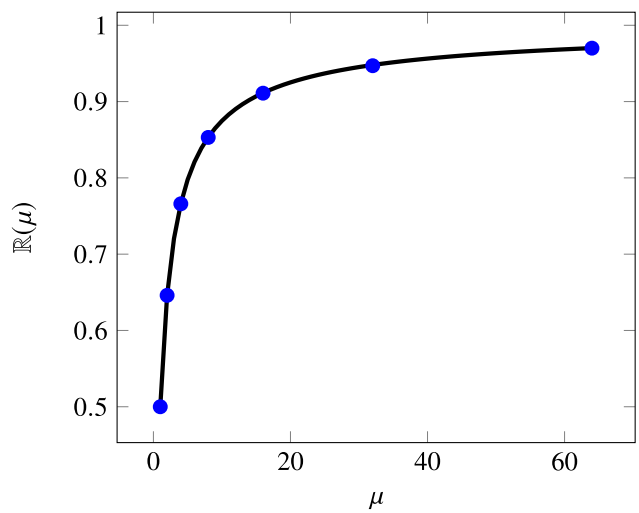


Fig. 1. Asymptotic rate of $\mathcal{B}(\ell,\mu)$ for fixed μ.

Table III and Figure 1 summarize the asymptotic rate results of $\mathcal{B}(\ell,\mu)$ for some chosen values of μ, compared with the known rates of de Bruijn sequences. Observe that when μ increases beyond $\mu = 1$, which coincides with de Bruijn sequences that have a rate of $1/2$, the asymptotic rate grows quickly and approaches 1.

Throughout this paper, it is assumed that the sequences of $\mathcal{B}(\ell,\mu)$ are cyclic where two sequences that are cyclic shifts of each other are considered distinct. For a sequence $\boldsymbol{s} \in \mathcal{B}(\ell,\mu)$, the number of its distinct cyclic shifts in $\mathcal{B}(\ell,\mu)$ is a divisor of μ, and is based on the period of $\boldsymbol{s}$. Let $\mathcal{CB}(\ell,\mu)$ denote the set of (ℓ,μ)-BdB sequences where two sequences that are cyclic shifts of each other are considered the same sequence, and let $\mathcal{PB}(\ell,\mu) \subseteq \mathcal{CB}(\ell,\mu)$ denote the set of (ℓ,μ)-BdB sequences with a period of $\mu 2^\ell$. In the sequel, we analyze the cardinality of these two sets. We use in our analysis Burnside's Lemma [7] and Möbius inversion formula [35], quoted next.

Lemma 23 (Burnside's Lemma): Let $\mathcal{G}$ be a finite group of operators that acts on a set S. The number of equivalence

classes into which S is partitioned by $\mathcal{G}$ is

$$\frac{1}{|\mathcal{G}|}\sum_{g\in\mathcal{G}} I(g),$$

where $I(g)$ is the number of elements of S that are left fixed by the operator g.

Lemma 24 (Möbius Inversion Formula): If g and h are arithmetic functions satisfying

$$g(n)=\sum_{d|n} h(d) \text{ for every integer } n\geqslant 1,$$

then

$$h(n)=\sum_{d|n}\mu(d)g\left(\frac{n}{d}\right) \text{ for every integer } n\geqslant 1,$$

where the sums extend over all positive divisors d of n (indicated by $d|n$ in the above formulas), and μ is the Möbius function, defined by

$$\mu(n)=\begin{cases} 1 & n=1\\ 0 & a^2|n \text{ for some } a>1\\ (-1)^r & n \text{ has } r \text{ distinct prime factors.}\end{cases}$$

Theorem 25: For every integers ℓ,μ,

$$|\mathcal{CB}(\ell,\mu)|=\frac{2^{2^{\ell-1}-\ell}}{\mu}\sum_{\substack{i=1\\ \lambda=\gcd(\mu,i)}}^{\mu}\binom{2\lambda-1}{\lambda-1}^{2^{\ell-1}}, \tag{5}$$

$$|\mathcal{PB}(\ell,\mu)|=\frac{2^{2^{\ell-1}-\ell}}{\mu}\sum_{d|\mu}\mu(d)\binom{\frac{2\mu}{d}-1}{\frac{\mu}{d}-1}^{2^{\ell-1}}. \tag{6}$$

Proof: First, we analyze $|\mathcal{CB}(\ell,\mu)|$ using Burnside's Lemma. We apply the lemma on the sequences of $\mathcal{B}(\ell,\mu)$, where $\mathcal{G}$ consists of the $\mu 2^\ell$ cyclic permutations on the sequences of $\mathcal{B}(\ell,\mu)$. The number of equivalence classes into which $\mathcal{B}(\ell,\mu)$ is partitioned by $\mathcal{G}$ corresponds to the size of $\mathcal{CB}(\ell,\mu)$. Notice that sequences are left"fixed" only by permutations which are multiples of 2^ℓ. Hence,

$$|\mathcal{CB}(\ell,\mu)|=\frac{1}{\mu 2^\ell}\sum_{i=1}^{\mu}|I(g_i)|,$$

where g_i is a shift of $i2^\ell$ symbols. It follows that the period of the sequences of $I(g_i)$ is $\lambda 2^\ell$ where $\lambda=\gcd(\mu,i)$, and in fact these sequences correspond to the sequences of $\mathcal{B}(\ell,\lambda)$, concatenated repeatedly μ/λ times. Thus,

$$\begin{aligned}|\mathcal{CB}(\ell,\mu)| &= \frac{1}{\mu 2^\ell}\sum_{\substack{i=1\\ \lambda=\gcd(\mu,i)}}^{\mu}|\mathcal{B}(\ell,\lambda)| \quad (|I(g_i)|=|\mathcal{B}(\ell,\lambda)|)\\ &= \frac{1}{\mu 2^\ell}\sum_{\substack{i=1\\ \lambda=\gcd(\mu,i)}}^{\mu}2^{2^{\ell-1}}\binom{2\lambda-1}{\lambda-1}^{2^{\ell-1}} \qquad \text{(Theorem 20)}\\ &= \frac{2^{2^{\ell-1}-\ell}}{\mu}\sum_{\substack{i=1\\ \lambda=\gcd(\mu,i)}}^{\mu}\binom{2\lambda-1}{\lambda-1}^{2^{\ell-1}}. \qquad \text{(Algebraic manipulation)}\end{aligned}$$

TABLE IV
SET $\mathcal{CB}(2,2)$

Sequence	Period
00010111	8
00011011	8
00011101	8
00100111	8
00110011	4

Second, we analyze $|\mathcal{PB}(\ell,\mu)|$. Notice the following connection between $|\mathcal{CB}(\ell,\mu)|$ and $|\mathcal{PB}(\ell,\mu)|$,

$$|\mathcal{CB}(\ell,\mu)|=\sum_{\lambda|\mu}\lambda 2^\ell|\mathcal{PB}(\ell,\lambda)|.$$

Denote $g(\mu)=|\mathcal{CB}(\ell,\mu)|=2^{2^{\ell-1}}\binom{2\mu-1}{\mu-1}^{2^{\ell-1}}$ and $h(\mu)=\mu 2^\ell|\mathcal{PB}(\ell,\mu)|$, and then by Möbius inversion formula,

$$\mu 2^\ell|\mathcal{PB}(\ell,\mu)|=\sum_{d|n}\mu(d)2^{2^{\ell-1}}\binom{\frac{2\mu}{d}-1}{\frac{\mu}{d}-1}^{2^{\ell-1}},$$

and (6) follows by algebraic manipulation. ■

Example 8: Let $\ell=\mu=2$. By Theorem 20,

$$|\mathcal{B}(\ell,\mu)|=36,$$

and by Theorem 25,

$$|\mathcal{CB}(\ell,\mu)|=5 \text{ and } |\mathcal{PB}(\ell,\mu)|=4.$$

The sequences of $\mathcal{CB}(\ell,\mu)$ are shown in Table IV along with their periods.

Each sequence of $\mathcal{CB}(\ell,\mu)$ contributes to $\mathcal{B}(\ell,\mu)$ an amount of sequences which is equal to its period. Hence, we have

$$|\mathcal{B}(2,2)|=8\cdot 4+4\cdot 1=36,$$

as expected. □

V. Encoding BdB Sequences Based on the Generalized De Bruijn Graph

In this section, we present an encoding algorithm that utilizes Algorithm 2 in order to generate (ℓ,μ) -BdB sequences of length $n=\mu 2^\ell$, for $\ell=\mathcal{O}(\log\log n)$.

By Lemma 18, given a reverse spanning tree T of $G_{\ell,\mu}$, Algorithm 2 forms $2\binom{2\mu-1}{\mu-1}^{2^{\ell-1}}$ distinct sequences of $\mathcal{B}(\ell,\mu)$. This value corresponds to $\binom{2\mu}{\mu}$ possible orders to traverse the outgoing edges of the root $\boldsymbol{r}$, and $\binom{2\mu-1}{\mu-1}$ possible orders to traverse the outgoing edges of each $v\neq\boldsymbol{r}$. This immediately yields an efficient enumerative encoding algorithm [11] for these sequences for a given reverse spanning tree T. The series of choices to traverse the outgoing edges of $\boldsymbol{r}$ can be represented by a balanced *edges-choice* sequence $\boldsymbol{z}$ of length 2μ and weight μ. Assume that the outgoing edges of $\boldsymbol{r}$ are e_0,e_1, then at the i-th visit of $\boldsymbol{r}$, the algorithm picks to traverse e_0 if $z_i=0$, and e_1 if $z_i=1$. Similarly, the series of choices to traverse the outgoing edges of any $v\neq\boldsymbol{r}$, with

outgoing edges e_0 (for edges parallel to the starred edge) and e_1, can be represented by a balanced edges-choice sequence $\boldsymbol{z}$ of length $2\mu-1$ and weight μ. In this case, the starred edge is traversed at the last visit of v, otherwise the chosen edge is picked depending if $z_i = 0$ for e_0 or $z_i = 1$ for e_1. Each of these balanced sequences can be encoded using Knuth's algorithm [28] with $\log\mu + \Theta(\log\log\mu)$ redundancy bits.

Algorithm 3 receives an input sequence $\boldsymbol{s} \in \mathbb{F}_2^{n-\rho}$ for redundancy $\rho = 2^{\ell-1}\log\mu + \Theta(2^{\ell-1}\log\log\mu)$. The algorithm first identifies a reverse spanning tree T from the first $2^{\ell-1}-\ell$ bits of $\boldsymbol{s}$. The rest of the input sequence $\boldsymbol{s}$ is partitioned to $2^{\ell-1}$ disjoint sequences and each of them is balanced using Knuth's algorithm to a corresponding edges-choice sequence. Then Algorithm 2 is invoked to generate a unique sequence in $\mathcal{B}(\ell,\mu)$ associated with the tree T and the balanced edges-choice sequences.

Algorithm 3 Encoding (ℓ,μ)-BdB Sequences

Input: A sequence $\boldsymbol{s} \in \mathbb{F}_2^{n-\rho}$ where $n = \mu 2^\ell$
Output: A sequence $\boldsymbol{x} \in \mathcal{B}(\ell,\mu)$
Additional Ingredients: Knuth's algorithm $\mathcal{E}_K$, with redundancy $\rho = 2^{\ell-1}\log\mu + \Theta(2^{\ell-1}\log\log\mu)$.

1: Extract an integer i such that $b(i)$ equals the $(2^{\ell-1}-\ell)$-prefix of $\boldsymbol{s}$. Let T be the i-th reverse spanning tree of G_ℓ.
2: Partition the rest of $\boldsymbol{s}$ to $2^{\ell-1}$ sequences of equal length $\{\boldsymbol{y}_v\}_{v\in\mathbb{F}_2^{\ell-1}}$.
3: Use $\mathcal{E}_K$ to encode $\boldsymbol{y}_0$ to $\boldsymbol{z}_0$, a balanced sequence of length 2μ.
4: For every $v \in \mathbb{F}_2^{\ell-1}$, use $\mathcal{E}_K$ to encode $\boldsymbol{y}_i$ to $\boldsymbol{z}_i$, a sequence of length-$(2\mu-1)$ and weight μ.
5: Apply Algorithm 2 with the tree T and the sequences $\{\boldsymbol{z}_v\}_{v\in\mathbb{F}_2^{\ell-1}}$ as follows:
   a: For every $i \in [2\mu]$, at the i-th visit of $\boldsymbol{r}$, if $(\boldsymbol{z}_{\boldsymbol{r}})_i = 0$ traverse the self loop $\boldsymbol{r} \to \boldsymbol{r}$, and if $(\boldsymbol{z}_{\boldsymbol{r}})_i = 1$, traverse the other outgoing edge of $\boldsymbol{r}$.
   b: For every $i \in [2\mu-1]$, at the i-th visit of $v \neq \boldsymbol{r}$, if $(\boldsymbol{z}_v)_i = 0$ traverse the outgoing starred edge of v, if $(\boldsymbol{z}_v)_i = 1$, traverse the outgoing unstarred edge.
6: Return the unique sequence that is associated with the path generated at Step 5 by Algorithm 2.

Example 9: As in Example 7, we continue with $\ell = 3$, $\mu = 2$. The corresponding graph $G_{3,2}$ was presented in Example 7. Notice that $G_{3,2}$ has only two reverse spanning trees (which we denote only by their edges for simplicity),

$$T_0 = \{011, 110, 100\} \text{ and } T_1 = \{010, 110, 100\}.$$

Indeed, the tree can be encoded using $2^{\ell-1}-\ell = 1$ bits.

For the application of Algorithm 2 at Step 5 we need $|\boldsymbol{z}_{\boldsymbol{r}}| = 2\mu = 4$ and $|\boldsymbol{z}_v| = 2\mu-1 = 3$ for every $v \neq \boldsymbol{r}$. Thus, we require $|\boldsymbol{y}_v| = 1$ for every $v \in V$ and support input sequences of length $|\boldsymbol{s}| = 5$. For simplicity, we use in this example a naive encoding $\mathcal{E}$ of balanced sequences, instead of Knuth's encoding $\mathcal{E}_K$. Assume the following naive encoding of balanced sequences of length 3,

$$\mathcal{E}(0) = 011, \mathcal{E}(1) = 110.$$

A balanced sequence of length 4 is generated by $\mathcal{E}(\sigma) = 0 \circ \mathcal{E}(\sigma)$.
Let $\boldsymbol{s} = 10001$ be the input sequence. The tree T_1 is induced from $s_0 = 1$ at Step 1. Then, the rest of $\boldsymbol{s}$ is partitioned into

$$\boldsymbol{y}_{00} = \boldsymbol{y}_{01} = \boldsymbol{y}_{10} = 0, \boldsymbol{y}_{11} = 1.$$

At Step 3, these sequences are encoded to balanced edge-choice sequences

$$\boldsymbol{z}_{00} = 0011, \boldsymbol{z}_{01} = \boldsymbol{z}_{10} = 011, \boldsymbol{z}_{11} = 110.$$

Next, at Step 5 Algorithm 2 is invoked with the edge-choice sequences $\boldsymbol{z}_{00}, \boldsymbol{z}_{01}, \boldsymbol{z}_{10}, \boldsymbol{z}_{11}$. For example, the outgoing edges of $\boldsymbol{r} = 00$ are traversed in the following order: $000, 000, 001, 001$. The outgoing edges of $v = 01$ are traversed in the following order: $010, 011, 011, 010$, since from T_1 the starred edge going out of v is 010. Similarly, the outgoing edges of $v = 10$ are traversed as the following: $100, 101, 101, 100$, and for $v = 11$ the order is $111, 111, 110, 110$. The path generated from these edge-choice sequences is the same path generated in Example 7, and the output BdB sequence is

$$\boldsymbol{x} = 0000100111101101.$$

□

Note that it is necessary for Algorithm 3 to generate any of the $2^{2^{\ell-1}-\ell}$ reverse spanning trees of $G_{\ell,\mu}$. Unfortunately, there is no enumerative encoding algorithm for these spanning trees for large ℓ, and they can only be generated using exhaustive search [16]. Therefore, in order to have linear time complexity, the algorithm is applicable only for small values of ℓ that satisfy $\ell = \mathcal{O}(\log\log n)$.

Theorem 26: For every $\boldsymbol{s} \in \mathbb{F}_2^{n-\rho}$, Algorithm 3 returns a unique (ℓ,μ)-BdB sequence. The algorithm uses

$$\rho = 2^{\ell-1}\log\mu + \Theta(2^{\ell-1}\log\log\mu)$$

redundancy bits and its time complexity is $\mathcal{O}(n)$.

Proof: It follows from Step 5 and by Lemma 16 that Algorithm 3 outputs an (ℓ,μ)-BdB sequence. The uniqueness of the generated sequence follows from the uniqueness of the tree T that was induced at Step 1 and the uniqueness of the balanced sequences generated by Knuth's algorithm at Steps 3 and 4. The redundancy of the algorithm follows from the redundancy of Knuth's algorithm multiplied by $2^{\ell-1}$. The time complexity of the algorithm follows from the complexity of Knuth's algorithm and since $\ell = \mathcal{O}(\log\log n)$. ■

An efficient decoder for Algorithm 3 simply reverses the encoder. First, it identifies the path in the graph that corresponds to the sequence $\boldsymbol{x}$, and finds the tree T according to the last outgoing edge in the path for every vertex in $V \setminus \{\boldsymbol{r}\}$. Next, the decoder travels the path and reveals the sequences $\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{2^{\ell-1}-1}$ according to the series of outgoing edges chosen at every visit of each vertex. Finally, it decodes the sequences $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{2^{\ell-1}-1}$ using Knuth's algorithm decoder and combines with the binary representation of the tree T to construct the input sequence $\boldsymbol{s} \in \mathbb{F}_2^{n-\rho}$.

Remark 3: The balancing encoder used in Algorithm 3, Knuth's encoder $\mathcal{E}_K$ can be replaced with any other balanced

sequences encoder $\mathcal{E}$ (like in Example 9). In this case, the redundancy of the algorithm will be $2^{\ell-1} \cdot \mathrm{red}_{\mathcal{E}}(\mu)$, where $\mathrm{red}_{\mathcal{E}}$ is the redundancy of $\mathcal{E}$.

VI. A Hierarchy of Balanced Sequences Based on Their Derivatives

In this section, we study the family of sequences with balanced derivatives; these sequences are highly related to balanced de Bruijn sequences. For a sequence $\boldsymbol{s}$ and an integer k, let $\boldsymbol{s}^{(k)}$ denote its *k-th derivative*, defined recursively by $\boldsymbol{s}^{(k)} \triangleq \left(\boldsymbol{s}^{(k-1)}\right)'$ and $\boldsymbol{s}^{(0)} = \boldsymbol{s}$.

Definition 27: A sequence $\boldsymbol{s} \in \mathbb{F}_2^n$ is called **k-order balanced**, for $k \geqslant 0$, if $\boldsymbol{s}^{(0)}, \boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(k-1)}$ are all balanced. Let $\mathcal{D}(n,k)$ denote the family of k-order balanced sequences over $\mathbb{F}_2^n$. The following lemma presents a connection between k-order balanced sequences and (ℓ,μ)-BdB sequences. This lemma uses a similar technique as the one used in [8, Theorem 8].

Lemma 28: For all admissible integers ℓ, μ, if $\boldsymbol{s}$ is an (ℓ,μ)-BdB sequence, then it is ℓ-order balanced.

Proof: First notice that if a sequence $\boldsymbol{x}$ contains an equal amount of each t-tuple for some integer $t > 0$, then $\boldsymbol{x}$ is clearly balanced. It follows immediately that $\boldsymbol{s}$ is balanced since it contains each ℓ-tuple exactly μ times. Second, note that from the definition of the derivative sequence, if $\boldsymbol{s}_{i,\ell} = \boldsymbol{v}$ then $\boldsymbol{s}'_{i,\ell-1} = \boldsymbol{v}'$ (note that $\boldsymbol{v}$ is acyclic). Notice that every ℓ-tuple $\boldsymbol{v} \in \mathbb{F}_2^\ell$ satisfies that $\boldsymbol{v}' = \bar{\boldsymbol{v}}'$ (and no other ℓ-tuple shares this derivative), and both $\boldsymbol{v}, \bar{\boldsymbol{v}}$ appear exactly μ times in $\boldsymbol{s}$; therefore, each $(\ell-1)$-tuple appears exactly 2μ times in $\boldsymbol{s}'$. By using an induction, we get that for every $t \in [\ell-1]$, $\boldsymbol{s}^{(t)}$ contains an equal amount of each $(\ell-t)$-tuple, and therefore it is balanced. ■

Lemma 28 gives an upper bound on the redundancy of $\mathcal{D}(n,k)$, using the results presented in Theorem 22.

Corollary 29: For every $k \geqslant 2$, the redundancy of $\mathcal{D}(n,k)$ satisfies

$$\mathrm{red}(\mathcal{D}(n,k)) \leqslant 2^{k-2}\log(n) + \mathcal{O}(1).$$

By Lemma 7, when $k = 2$ the converse of Lemma 28 holds as well and thus the bound is tight.

Corollary 30: The redundancy of $\mathcal{D}(n,2)$ satisfies

$$\mathrm{red}(\mathcal{D}(n,2)) = \log n + \mathcal{O}(1).$$

However, when $k > 2$ the converse of Lemma 28 no longer holds; see the following example.

Example 10: Let $\ell = 3, \mu = 3, n = 24$ and denote

$$\boldsymbol{s} = 000101111101001110010010.$$

Note that $\boldsymbol{s}$ is balanced, and similarly

$$\boldsymbol{s}' = 001110000111010010110110$$

and

$$\boldsymbol{s}'' = 010010001001110111011010$$

are balanced as well. Therefore, $\boldsymbol{s} \in \mathcal{D}(n,3)$. However, the 3-tuple 111 appears four times as a window of $\boldsymbol{s}$ and hence $\boldsymbol{s} \notin \mathcal{B}(3,3)$. □

In the rest of this section, we study tighter bounds for $\mathrm{red}(\mathcal{D}(n,k))$ for different values of k, in order to improve the bound presented in Corollary 29. Next, we present a tight bound for the case of $k = 3$ that is obtained by exact enumeration of 3-order balanced sequences.

Assume w.l.o.g that $\boldsymbol{s}$ starts with *zero*, and denote $\boldsymbol{s}$ as a series of runs,

$$\boldsymbol{s} = 0^{a_0}1^{b_0}0^{a_1}1^{b_1}\cdots 0^{a_{m-1}}1^{b_{m-1}}0^{a_m}, \tag{7}$$

where for every $i \in [m]$, $a_i, b_i > 0$ and the last run of *zeros* satisfies $a_m \geqslant 0$. We denote the first $2m$ runs of *zeros* and *ones* as the *primary runs* of $\boldsymbol{s}$, i.e., all the runs besides 0^{a_m} (which can be empty).

Lemma 31: The sequence $\boldsymbol{s} \in \mathbb{F}_2^n$ is 2-order balanced if and only if $\boldsymbol{s}$ is balanced and has $n/4$ primary runs of *zeros* and $n/4$ primary runs of *ones*.

Proof: By Lemma 7, $\boldsymbol{s}$ is 2-order balanced if and only if it is an $(2, n/4)$-BdB sequence. Every transition from a run of *zeros* to a run of *ones* induces an occurrence of the 2-tuple 01, and vice versa; similarly, every transition from a run of *ones* to a run of *zeros* induces an occurrence of the 2-tuple 10, and vice versa. Hence, $\boldsymbol{s}$ is a $(2, n/4)$-BdB sequence if and only if has $n/4$ primary runs of *zeros* and $n/4$ primary runs of *ones*, which proves the lemma. ■

Lemma 32: The sequence $\boldsymbol{s} \in \mathbb{F}_2^n$ is 3-order balanced if and only if $\boldsymbol{s}$ is 2-order balanced and has exactly $n/4$ primary runs of length 1.

Proof: First, assume that $\boldsymbol{s}$ is 3-order balanced. By the structure given by (7) and by Lemma 31 (since $\boldsymbol{s}$ is 2-balanced), we have that

$$\boldsymbol{s}' = 0^{a_0-1}10^{b_0-1}10^{a_1-1}10^{b_1-1}\cdots 0^{b_{n/4-1}-1}10^{a_{n/4}}. \tag{8}$$

Since $\boldsymbol{s}$ is 3-order balanced, $\boldsymbol{s}'$ is 2-order balanced. Hence, by Lemma 31 $\boldsymbol{s}'$ is balanced and has exactly $n/4$ primary runs of each *ones* and *zeros*. Therefore, it follows that $n/4$ of the $n/2$ runs of *zeros* in $\boldsymbol{s}'$ are empty. In other words, $n/4$ of the $n/2$ variables $a_0, \ldots, a_{n/4-1}, b_0, \ldots, b_{n/4-1}$ should be equal to 1.

On the other direction, assume that $\boldsymbol{s}$ is 2-order balanced and has exactly $n/4$ primary runs of length 1. Similarly, by the structure of $\boldsymbol{s}'$ given by (8), $n/4$ of the $n/2$ variables $a_0, \ldots, a_{n/4-1}, b_0, \ldots, b_{n/4-1}$ should be equal to 1. Therefore, $\boldsymbol{s}'$ is balanced and has exactly $n/4$ primary runs of each *ones* and *zeros*. By the other direction of Lemma 31, $\boldsymbol{s}'$ is 2-order balanced and hence $\boldsymbol{s}$ is 3-order balanced. ■

Theorem 33: The redundancy of $\mathcal{D}(n,3)$ satisfies

$$\mathrm{red}(\mathcal{D}(n,3)) = 1.5\log n + \mathcal{O}(1).$$

The proof of Theorem 33 is given in Appendix A.

The next theorem gives an upper bound on the redundancy of $\mathcal{D}(n,k)$ for any fixed $k \in \mathbb{N}$. This bound significantly improves the upper bound given in Corollary 29.

Theorem 34: Let $k \in \mathbb{N}$ be a constant integer. The redundancy of $\mathcal{D}(n,k)$ satisfies

$$\mathrm{red}(\mathcal{D}(n,k)) \leqslant k\log n + \Theta(\log\log n).$$

The proof of Theorem 34 is given by an explicit encoding algorithm, presented next. The correctness of this

encoding algorithm which also validates the correctness of Theorem 34, is given by Theorem 41. This algorithm generalizes Algorithm 1 and uses Knuth's encoder, $\mathcal{E}_K$, in order to efficiently encode k-order balanced sequences for any fixed k while its redundancy is

$$\rho = k \log n + \Theta(\log \log n).$$

Algorithm 4 receives an input sequence $\boldsymbol{s} \in \mathbb{F}_2^{n-\rho}$ where $\rho = k \log n + \Theta(\log \log n)$, and outputs a sequence $\widehat{\boldsymbol{x}} \in \mathcal{D}(n,k)$. This algorithm follows a similar structure as Algorithm 1. First, Knuth's encoder is used k times in order to generate a sequence $\widehat{\boldsymbol{s}}$ and an indices sequence $\boldsymbol{v}$ of length $k \log n$, such that $\widehat{\boldsymbol{s}}$ is balanced and each of its first $k-1$ derivatives has a constant imbalance. Then, the process is repeated with $\boldsymbol{v}$ as an input to create $\widehat{\boldsymbol{v}}$ and $\boldsymbol{u}$, a second balancing indices sequence of length $k \log(k \log n)$. Since $\boldsymbol{u}$ is short, it can be transformed to a k-order balanced sequence, $\widehat{\boldsymbol{u}}$, using a fixed encoding lookup table, denoted by LT_n^k, which has $2^{k \log(k \log n)} = \Theta(\log n)$ elements. Moreover, from the upper bound on the cardinality of $\mathcal{D}(n,k)$ that follows from Lemma 28, it follows that the length $k \log(k \log n) + \Theta(\log \log \log n)$ is sufficient for $\widehat{\boldsymbol{u}}$ for every n. Later, $\widehat{\boldsymbol{s}}, \widehat{\boldsymbol{v}}, \widehat{\boldsymbol{u}}$ are concatenated with *zeros* to construct $\boldsymbol{x}$, a sequence which itself and each of its first $k-1$ derivatives has a constant imbalance. Finally, manual balancing of $\boldsymbol{x}$ is performed, and $\widehat{\boldsymbol{x}}$, a k-order balanced sequence of length n is obtained.

Algorithm 4 Encoding k-Order Balanced Sequences

Input: A sequence $\boldsymbol{s} \in \mathbb{F}_2^{n-\rho}$
Output: A sequence $\widehat{\boldsymbol{x}} \in \mathcal{D}(n,k)$
Additional Parameters: Knuth's algorithm $\mathcal{E}_K$, redundancy $\rho = k \log n + \Theta(\log \log n)$, constant predetermined values $m_0, \ldots, m_{k-1}$.

1: Compute $\boldsymbol{s}^{(0)}, \ldots, \boldsymbol{s}^{(k-1)}$ and denote $\boldsymbol{s}_{k-1} = \boldsymbol{s}^{(k-1)}$.
2: **for** $t = k-1, k-2, \ldots, 1, 0$ **do**
3: Use $\mathcal{E}_K$ to find i_t, the balancing index of $\boldsymbol{s}_t$, and update $\boldsymbol{s}_t$ by flipping its i_t-prefix.
4: If $t > 0$ obtain $\boldsymbol{s}_{t-1} = \int(\boldsymbol{s}_t, s_0^{(t-1)})$.
5: **end for**
6: Denote $\widehat{\boldsymbol{s}} = \boldsymbol{s}_0$ and construct the indices sequence $\boldsymbol{v} = b(i_0) \circ \cdots \circ b(i_{k-1})$ of length $k \log n$.
7: Repeat Steps 1-6 with $\boldsymbol{v}$ as an input to receive $\widehat{\boldsymbol{v}}$ and the indices sequence $\boldsymbol{u}$ of length $k \log(k \log n)$.
8: Use the lookup table to encode $\widehat{\boldsymbol{u}} = LT_n^k(\boldsymbol{u})$, a k-order balanced sequence of length $k \log \log n + \Theta(\log \log \log n)$.
9: Let $\boldsymbol{x} = 0^{k-1} \circ \widehat{\boldsymbol{s}} \circ \widehat{\boldsymbol{v}} \circ \widehat{\boldsymbol{u}}$.
10: Compute $\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(k-1)}$ and denote $\boldsymbol{x}_{k-1} = \boldsymbol{x}^{(k-1)}$.
11: **for** $t = k-1, k-2, \ldots, 1, 0$ **do**
12: Append at the end of $\boldsymbol{x}_t$ a sequence $\boldsymbol{y}_t$ of length m_t such that:
  a: $\mathrm{Pref}_{k-t-1}(\boldsymbol{y}_t) = \boldsymbol{0}$,
  b: $\boldsymbol{x}_t \circ \boldsymbol{y}_t$ is balanced,
  c: $\boldsymbol{y}_t'$ is $(k-t-1)$-order balanced.
13: If $t > 0$ obtain $\boldsymbol{x}_{t-1} = \int(\boldsymbol{x}_t, x_0^{(t-1)})$.
14: **end for**
15: Return $\widehat{\boldsymbol{x}} = \boldsymbol{x}_0$.

First, we have the following useful lemma, which generalizes Lemma 8 what was presented in Section III.

Lemma 35: Let $\boldsymbol{y}, \boldsymbol{z}$ be sequences such that their derivatives are k-order balanced for some integer $k \geqslant 1$. If $\mathrm{Pref}_k(\boldsymbol{y}) = \mathrm{Pref}_k(\boldsymbol{z})$, then $(\boldsymbol{y} \circ \boldsymbol{z})'$ is k-order balanced as well.

Proof: Let $|\boldsymbol{y}| = n, |\boldsymbol{z}| = m$. We prove the statement using an induction. The case of $k = 1$ was proved in Lemma 8. Assume that this statement holds for $k-1 > 0$. Since $\mathrm{Pref}_k(\boldsymbol{y}) = \mathrm{Pref}_k(\boldsymbol{z})$, the derivatives $\boldsymbol{y}'$ and $\boldsymbol{z}'$ satisfy that $\mathrm{Pref}_{k-1}(\boldsymbol{y}') = \mathrm{Pref}_{k-1}(\boldsymbol{z}')$ and in particular $y_0 = z_0$. Therefore, $(\boldsymbol{y}' \circ \boldsymbol{z}')'$ is $(k-1)$-order balanced. Next, we have

$$\begin{aligned}(\boldsymbol{y} \circ \boldsymbol{z})' &= \mathrm{Pref}_{n-1}(\boldsymbol{y}') \circ (y_{n-1} + z_0) \circ \mathrm{Pref}_{m-1}(\boldsymbol{z}') \circ (z_{m-1} + y_0) \\ &= \mathrm{Pref}_{n-1}(\boldsymbol{y}') \circ (y_{n-1} + y_0) \circ \mathrm{Pref}_{m-1}(\boldsymbol{z}') \circ (z_{m-1} + z_0) \\ &= \boldsymbol{y}' \circ \boldsymbol{z}'.\end{aligned}$$

It follows that $(\boldsymbol{y} \circ \boldsymbol{z})'$ is k-order balanced and the statement holds for k. ■

Next, we provide explanations and lemmas that prove the correctness of Algorithm 4. The loop of Steps 2 to 5 receives $\boldsymbol{s}$ and its first $k-1$ derivatives, where $\boldsymbol{s}^{(k-1)}$ is denoted by $\boldsymbol{s}_{k-1}$. For $t = k-1, \ldots, 0$, the algorithm iteratively uses $\mathcal{E}_K$ to balance the current sequence $\boldsymbol{s}_t$ and sets $\boldsymbol{s}_{t-1}$ as its integral. The result of this process is a sequence $\widehat{\boldsymbol{s}} = \boldsymbol{s}_0$ that is balanced and each of its first $k-1$ derivatives has constant imbalance, as shown in the following lemma.

Lemma 36: At Step 6 of Algorithm 4, $\widehat{\boldsymbol{s}}$ is balanced and for every $t \in [1, k-1]$, $|\delta(\widehat{\boldsymbol{s}}^{(t)})| \leqslant 2^{t+1}$.

Proof: First, $\boldsymbol{s}_t$ is balanced at the t-th iteration of Step 4 and therefore $\widehat{\boldsymbol{s}}$ is balanced. Second, observe that flipping a prefix of a sequence changes at most two bits in its derivative, and changing any bit of a sequence affects exactly two bits in its derivative. We can infer that flipping a prefix of a sequence imbalances its t-th derivative by at most 2^t bits. Thus, we have

$$|\delta(\widehat{\boldsymbol{s}}^{(t)}) - \delta(\boldsymbol{s}_t)| = 2^t + 2^{t-1} + \cdots + 2 \leqslant 2^{t+1},$$

and since $\boldsymbol{s}_t$ is balanced, it follows that $|\delta(\widehat{\boldsymbol{s}}^{(t)})| \leqslant 2^{t+1}$. ■

Later, the algorithm repeats Steps 1 to 6 with $\boldsymbol{v}$ (the indices sequence of $\boldsymbol{s}$) as an input to receive $\widehat{\boldsymbol{v}}$ that satisfies Lemma 36, and $\boldsymbol{u}$, a second balancing indices sequence of length $k \log(k \log n)$. Then, $\boldsymbol{u}$ is encoded to a k-order balanced sequence, $\widehat{\boldsymbol{u}}$, using a fixed encoding lookup table. At Step 9, $\widehat{\boldsymbol{s}}, \widehat{\boldsymbol{v}}, \widehat{\boldsymbol{u}}$ are concatenated with *zeros* to construct $\boldsymbol{x}$, a sequence which itself and each of its first $k-1$ derivatives have a constant imbalance, as shown in the following lemma.

Lemma 37: At Step 9 of Algorithm 4, $\boldsymbol{x}$ satisfies that $\delta(\boldsymbol{x}) \leqslant k-1$ and for every $t \in [1, k-1]$, $|\delta(\boldsymbol{x}^{(t)})| \leqslant 2^{t+2} + 2^{t+1} + \lceil \frac{k-1}{2} \rceil$.

Proof: First, from the construction of $\boldsymbol{x}$ we have,

$$\delta(\boldsymbol{x}) = \delta(0^{k-1}) + \delta(\widehat{\boldsymbol{s}}) + \delta(\widehat{\boldsymbol{v}}) + \delta(\widehat{\boldsymbol{u}}) = \left\lceil \frac{k-1}{2} \right\rceil,$$

since $\widehat{\boldsymbol{s}}, \widehat{\boldsymbol{v}}, \widehat{\boldsymbol{u}}$ are balanced and $\delta(0^{k-1}) = \lceil \frac{k-1}{2} \rceil$ from the definition of imbalance. Next, by Claim 9, when concatenating two sequences $\boldsymbol{y}, \boldsymbol{z}$ it follows that

$$|\delta((\boldsymbol{y} \circ \boldsymbol{z})')| \leqslant |\delta(\boldsymbol{y}')| + |\delta(\boldsymbol{z}')| + 1,$$

and since changing a bit of a sequence affects exactly two bits in its derivative, we have,

$$|\delta((\boldsymbol{y}\circ\boldsymbol{z})^{(t)})| \leqslant |\delta(\boldsymbol{y}^{(t)})| + |\delta(\boldsymbol{z}^{(t)})| + 2^{t-1}.$$

By Lemma 36, which holds for $\widehat{\boldsymbol{v}}$ as well, and since $\widehat{\boldsymbol{u}}$ is k-order balanced, we receive,

$$\begin{aligned}|\delta(\boldsymbol{x}^{(t)})| &\leqslant \delta((0^{k-1})^{(t)}) + |\delta(\widehat{\boldsymbol{s}}^{(t)})| + |\delta(\widehat{\boldsymbol{v}}^{(t)})| + |\delta(\widehat{\boldsymbol{u}}^{(t)})| + 2^{t+1}\\ &= 2^{t+2} + 2^{t+1} + \left\lceil \frac{k-1}{2} \right\rceil.\end{aligned}$$

■

Next, Algorithm 4 performs manual balancing. The loop of Step 11 receives $\boldsymbol{x}$ and its first $k-1$ derivatives, where $\boldsymbol{x}^{(k-1)}$ is denoted by $\boldsymbol{x}_{k-1}$. For $t = k-1, k-2, \ldots, 1, 0$, the algorithm iteratively balances the current sequence $\boldsymbol{x}_t$ by appending $\boldsymbol{y}_t$, a sequence of constant length m_t that satisfies several conditions, and sets $\boldsymbol{x}_{t-1}$ as its integral. It is shown in the following lemma that the imbalance of each $\boldsymbol{x}_t$ is constant.

Lemma 38: At Step 12 of Algorithm 4, for every $t \in [k]$, $|\delta(\boldsymbol{x}_t)| \leqslant 2^{t+2} + 2^{t+1} + \lceil \frac{k-1}{2} \rceil + \sum_{t'=t+1}^{k-1} m_{t'}$.

Proof: Notice that when appending a sequence of length m_{k-1} to $\boldsymbol{x}_{k-1} = \boldsymbol{x}^{(k-1)}$, its integral sequence $\boldsymbol{x}_{k-2} = \int(\boldsymbol{x}_{k-1}, x_0^{(k-2)})$ is a concatenation of $\boldsymbol{x}^{(k-2)}$ with a sequence of length m_{k-1}. It follows that

$$|\delta(\boldsymbol{x}_{k-2}) - \delta(\boldsymbol{x}^{(k-2)})| \leqslant m_{k-1}.$$

Therefore, for every $t \in [k]$, at the t-th invocation of Step 12 it holds that

$$|\delta(\boldsymbol{x}_t) - \delta(\boldsymbol{x}^{(t)})| \leqslant \sum_{t'=t+1}^{k-1} m_{t'},$$

and therefore by Lemma 37 it holds that

$$|\delta(\boldsymbol{x}_t)| \leqslant 2^{t+2} + 2^{t+1} + \left\lceil \frac{k-1}{2} \right\rceil + \sum_{t'=t+1}^{k-1} m_{t'}.$$

■

The existence of suitable strings $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{k-1}$ that satisfy the constraints mentioned in Step 12 is proven in the next lemma, as well as an efficient method to generate them. This method is based on Algorithm 2 which is designed to generate BdB sequences.

Lemma 39: At Step 12 of Algorithm 4, for every $t \in [k]$ there exists a string $\boldsymbol{y}_t$ that satisfies the constraints: (a)

1) $\mathrm{Pref}_{k-t-1}(\boldsymbol{y}_t) = \boldsymbol{0}$,
2) $\boldsymbol{x}_t \circ \boldsymbol{y}_t$ is balanced,
3) $\boldsymbol{y}_t'$ is $(k-t-1)$-order balanced.

Proof: Let $\delta = \lceil \frac{m_t}{2} \rceil - w_H(\boldsymbol{y}_t) = -\delta(\boldsymbol{x}_t)$ denote the required imbalance of $\boldsymbol{y}_t$ such that $\boldsymbol{x}_t \circ \boldsymbol{y}_t$ is balanced. We also notate $k_t = k - t$. The sequence $\boldsymbol{y}_t$ will be constructed from some sequences $\omega_{t,1}, \omega_{t,0}, \omega_{t,-1}$, that satisfy $\delta(\omega_{t,1}) = 1$, $\delta(\omega_{t,0}) = 0, \delta(\omega_{t,-1}) = -1$, start with 0^{k_t-1} and their derivatives are (k_t-1)-order balanced.

We use Algorithm 2 that was presented in Section IV in order to generate a single $(k_t, 2)$-BdB sequence of length 2^{k_t+1} that starts with 0^{k_t-1}, which we denote by $\omega_{t,0}$. From Lemma 28, $\omega_{t,0}$ is k_t-order balanced. The sequence $\omega_{t,1}$ is generated by removing a single *one* from one of the windows of $\omega_{t,0}$ that are equal to 1^{k_t}, and inserting a single *zero* to one of the windows equal to 0^{k_t}. Notice that $\delta(\omega_{t,1}) = 1$, $\mathrm{Pref}_{k_t-1}(\omega_{t,1}) = 0^{k_t-1}$ and that its derivative satisfies that each of the binary (k_t-1)-tuples appears in it as a window 4 times, since $(0^{k_t})' = (1^{k_t})' = 0^{k_t-1}$. Hence, from Lemma 28 $\omega_{t,1}'$ is (k_t-1)-order balanced. Similarly, we generate $\omega_{t,-1}$ with $\delta(\omega_{t,-1}) = -1$ and $\mathrm{Pref}_{k_t-1}(\omega_{t,-1}) = 0^{k_t-1}$. Denote the length of the sequences $\omega_{t,0}, \omega_{t,1}, \omega_{t,-1}$ by $n_0 = 2^{k_t+1}$. Thus, we construct

$$\boldsymbol{y}_t = \begin{cases} \omega_{t,1}^{\delta} \circ \omega_{t,0}^{\frac{m_t}{n_0}-\delta} & \delta > 0 \\ \omega_{t,-1}^{|\delta|} \circ \omega_{t,0}^{\frac{m_t}{n_0}-\delta} & \text{otherwise} \end{cases}$$

The sequence $\boldsymbol{y}_t$ satisfies constraint (a) since the sequences $\omega_{t,0}, \omega_{t,1}, \omega_{t,-1}$ start with 0^{k_t-1}. Constraint (b) is satisfied from $\delta(\boldsymbol{y}_t) = \delta$. Constraint (c) follows from Lemma 35 since $\boldsymbol{y}_t$ is a concatenation of sequences that each of them starts with 0^{k_t-1} and has a derivative that is (k_t-1)-order balanced. ■

It follows from the proof of Lemma 39 that for every $t \in [k]$, the length m_t should satisfy

$$m_t \geqslant n_0|\delta(\boldsymbol{x}_t)| = 2^{k-t+1}|\delta(\boldsymbol{x}_t)|.$$

By Lemma 38,

$$|\delta(\boldsymbol{x}_t)| \leqslant 2^{t+2} + 2^{t+1} + \left\lceil \frac{k-1}{2} \right\rceil + \sum_{t'=t+1}^{k-1} m_{t'}.$$

Hence, the lengths $m_0, \ldots, m_{k-1}$ are constants that can be calculated in advance independently of n.

Finally, the result of the manual balancing loop is $\widehat{\boldsymbol{x}} = \boldsymbol{x}_0$ which is k-order balanced, as proved in the next lemma.

Lemma 40: After Step 15 of Algorithm 4, $\widehat{\boldsymbol{x}}$ is k-order balanced.

Proof: We prove using an induction that for every $t = k-1, \ldots, 0$, after the t-th iteration of Step 12, $\boldsymbol{x}_t \circ \boldsymbol{y}_t$ is $(k-t)$-order balanced. Then, the lemma statement will follow from applying $t = 0$ since $\widehat{\boldsymbol{x}} = \boldsymbol{x}_0$. For the base case, $t = k-1$, the induction statement follows immediately from the fact that $\boldsymbol{y}_t$ satisfies constraint (b), i.e., $\boldsymbol{x}_t \circ \boldsymbol{y}_t$ is balanced. Next, assume that $\boldsymbol{x}^{(t+1)}$ is $(k-t-1)$-order balanced. From Step 9 and from constraint (a), i.e., $\mathrm{Pref}_{k-t-1}(\boldsymbol{y}_t) = \boldsymbol{0}$, we have,

$$\mathrm{Pref}_{k-t-1}(\boldsymbol{y}_t) = \mathrm{Pref}_{k-t-1}(\boldsymbol{x}_t) = 0^{k-t-1}.$$

From constraint (c), $\boldsymbol{y}_t'$ is $(k-t-1)$-order balanced and by Lemma 35 we obtain that $(\boldsymbol{x}_t \circ \boldsymbol{y}_t)'$ is $(k-t-1)$-order balanced as well. Combining this with constraint (b), it follows that $\boldsymbol{x}_t \circ \boldsymbol{y}_t$ is $(k-t)$-order balanced as required and the statement holds for all t. ■

Theorem 41: Algorithm 4 returns a sequence $\widehat{\boldsymbol{x}} \in \mathcal{D}(n,k)$ that can be uniquely decoded to its input sequence $\boldsymbol{s}$. The algorithm uses $\rho = k\log n + \Theta(\log\log n)$ redundancy bits and its time complexity is $\mathcal{O}(n)$.

Proof: It follows immediately from Lemma 40 that the algorithm outputs a sequence $\widehat{\boldsymbol{x}} \in \mathcal{D}(n,k)$. As for the redundancy, the algorithm adds $k\log n$ redundancy bits at Step 6, and additional $k\log\log n + \Theta(\log\log\log n)$ redundancy bits at Step 8. From Step 9 and forward, the algorithm adds a fixed number of supplementary redundancy bits, since $m_0, \ldots, m_{k-1}$ and k are constants.

The complexity of the algorithm follows from the complexity of Knuth's algorithm, since Algorithm 1 invokes $\mathcal{E}_K$ a constant number of times, and from the time complexity of the derivative and integral operations, which is $\Theta(n)$. The rest of the operations invoked throughout the algorithm have constant time complexity.

A decoder for Algorithm 4 reverses Step 12 by iteratively trimming the m_t-suffix of $\widehat{\boldsymbol{x}}^{(t)}$ for every $t = 0, \ldots, k-1$ to recover $\boldsymbol{x}^{(k-1)}$. Then, $\boldsymbol{x}$ is recovered $\boldsymbol{x}^{(k-1)}$ using integrals. Next, the decoder extracts $\widehat{\boldsymbol{s}}, \widehat{\boldsymbol{v}}, \widehat{\boldsymbol{u}}$ and recovers $\boldsymbol{u}$ from the fixed encoding lookup table. Then, it recovers $\boldsymbol{v}$ from $\widehat{\boldsymbol{v}}$ using the balancing indices of $\boldsymbol{u}$ and similarly it recovers $\boldsymbol{s}$ from $\widehat{\boldsymbol{s}}$ using the balancing indices of $\boldsymbol{v}$. Note that this process has the same time complexity as the encoder, $\mathcal{O}(n)$. ■

VII. Conclusion

This paper studied two hierarchies of balanced sequences that are provided by classifying them based on some properties. First, we studied the classification to (ℓ, μ)-BdB sequences, i.e., sequences that contain each ℓ-tuple as a window of length ℓ exactly μ times. For $\ell = 2$, an efficient encoding algorithm, that is based on Knuth's balanced sequences encoder, was presented. Then, an enumeration of this set of sequences for all admissible values of ℓ and μ was presented; this enumeration is based on a generalization of de Bruijn graph of order ℓ, where each edge of the graph is duplicated μ times. Using the result of this enumeration, the cardinality of the set of (ℓ, μ)-BdB sequences was calculated, along with its redundancy and asymptotic rate for some interesting cases. Additionally, a generic encoding scheme for this set of sequences, which is based on the enumeration algorithm and is efficient for $\ell = \mathcal{O}(\log \log n)$, was provided. Moreover, the cardinality of a variation of the set of (ℓ, μ)-BdB sequences, where two sequences that are identical cyclically are considered the same sequence, was induced.

The second classification of balanced sequences is of k-order balanced sequences, for a fixed integer k. These sequences satisfy that their first k derivatives are balanced. The redundancy of this set of sequences was studied for several values of k, and an efficient encoding algorithm, based on Knuth's encoder of balanced sequences, was presented. For $k > 3$, only an upper bound on the redundancy of this set of sequences was given, and a precise result is a subject for future research.

While this work studied the binary case, most of the results in the paper can be extended for alphabet of any size q. Specifically, the results of Section IV can be extended using the q-ary generalization of de Bruijn graph [51].

Appendix A

The goal of this appendix is to provide the proof of Theorem 33, i.e., to show that $\text{red}(\mathcal{D}(n,3))) = 1.5 \log n + \mathcal{O}(1)$. First, we have the next claim which is given by the Central Limit Theorem of binomial variables [29].

Claim 42: For positive integer n and $r \in [-n/2, n/2]$,

$$\frac{1}{2^n}\binom{n}{n/2+r} \approx \sqrt{\frac{2}{n\pi}} \exp\left(\frac{-2r^2}{n}\right),$$

where for two functions $f(n), g(n)$, the notation $f(n) \approx g(n)$ is used to mean that $f(n)/g(n) \to 1$ as $n \to \infty$.

Before presenting the proof Theorem 33, we provide a useful lemma.

Lemma 43: For positive integers n and k,

$$\sum_{v=0}^{n} \binom{n}{v}^k \approx \frac{2^{kn}}{\sqrt{k}} \left(\frac{2}{\pi n}\right)^{\frac{k-1}{2}}.$$

Proof: By applying $r = v - n/2$, we have by Claim 42,

$$\sum_{v=0}^{n} \binom{n}{v}^k = \sum_{r=-n/2}^{n/2} \binom{n}{n/2+r}^k \approx 2^{kn} \left(\frac{2}{\pi n}\right)^{k/2} \sum_{r=-n/2}^{n/2} \exp\left(\frac{-2kr^2}{n}\right). \quad (9)$$

The last sum can be approximated with the following integral,

$$\int_{-\infty}^{\infty} \exp\left(\frac{-2kx^2}{n}\right) dx = \sqrt{\frac{\pi n}{2k}}. \quad (10)$$

and the lemmas statement follows from combining equations (9) and (10). ■

Proof: [Proof of Theorem 33] By the structure given by (7) and by Lemma 32, a sequence $\boldsymbol{s}$ that is 3-order balanced and starts w.l.o.g with *zero* has the form

$$\boldsymbol{s} = 0^{a_0} 1^{b_0} \cdots 0^{a_{n/4-1}} 1^{b_{n/4-1}} 0^{a_{n/4}},$$

where $n/4$ of its primary runs are of length 1. Hence, $\boldsymbol{s}$ has t primary runs of *ones* of length 1 and $n/4 - t$ primary runs of *zeros* of length 1, for some $t \in [1, n/4-1]$. We have $\binom{n/4}{t}$ possibilities to choose the t primary runs of *ones* of length 1, and also $\binom{n/4}{n/4-t} = \binom{n/4}{t}$ possibilities to choose the $n/4 - t$ primary runs of *zeros* of length 1. Since $\boldsymbol{s}$ is balanced, we have

$$\begin{cases} \sum_{i=0}^{n/4} a_i = \frac{n}{2}, \\ \sum_{j=0}^{n/4-1} b_j = \frac{n}{2}. \end{cases} \quad (11)$$

Let $i_0, \ldots, i_{t-1}$ denote the indices of the t runs of *zeros* of length greater than 1, and let $j_0, \ldots, j_{n/4-t-1}$ denote the indices of the $n/4 - t$ runs of *ones* of length greater than 1. By (11), these runs should satisfy

$$\begin{cases} \sum_{\tau=0}^{t-1} a_{i_\tau} + a_{n/4} = \frac{n}{2} - (\frac{n}{4} - t) & \forall_{\tau \in [t]} : a_{i_\tau} \geqslant 2, a_{n/4} \geqslant 0, \\ \sum_{\tau=0}^{n/4-t-1} b_{j_\tau} = \frac{n}{2} - t & \forall_{\tau \in [\frac{n}{4}-t]} : b_{j_\tau} \geqslant 2. \end{cases} \quad (12)$$

These two equations have the same number of solutions as

$$\begin{cases} \sum_{\tau=0}^{t-1} a'_{i_\tau} + a'_{n/4} = \frac{n}{4} - t & \forall \tau \in [t] : a'_{i_\tau} \geqslant 0, a'_{n/4} \geqslant 0, \\ \sum_{\tau=0}^{n/4-t-1} b'_{i_\tau} = t & \forall \tau \in [\frac{n}{4} - t] : b'_{i_\tau} \geqslant 0. \end{cases} \quad (13)$$

The number of solutions to the first equation is $\binom{t+1+n/4-t+1}{n/4-t} = \binom{n/4}{t}$, and the number of solutions to the second equation is $\binom{n/4-t+t-1}{t} = \binom{n/4-1}{t}$. After

multiplying by 2 in order to account for sequences that start with *one*, we receive

$$|\mathcal{D}(n,3)| = 2\sum_{t=1}^{n/4-1}\binom{n/4}{t}^3\binom{n/4-1}{t} = 2\sum_{t=1}^{n/4-1}\frac{n-4t}{n}\binom{n/4}{t}^4,$$

where the last transition follows from $\binom{n/4-1}{t} = \binom{n/4}{t}\frac{n-4t}{n}$. For simplicity of approximating $\mathrm{red}(\mathcal{D}(n,3))$, we can write

$$2\sum_{t=0}^{n/4}\binom{n/4}{t}^4 \leqslant |\mathcal{D}(n,3)| \leqslant 4\sum_{t=0}^{n/4}\binom{n/4}{t}^4.$$

By Lemma 43,

$$2\sum_{t=0}^{n/4}\binom{n/4}{t}^4 \approx 2^n\left(\frac{8}{\pi n}\right)^{1.5},$$

it follows that the size of $\mathcal{D}(n,3)$ satisfies

$$|\mathcal{D}(n,3)| \approx \alpha 2^n\left(\frac{8}{\pi n}\right)^{1.5},$$

for some $1 \leqslant \alpha \leqslant 2$ and hence its redundancy satisfies

$$\mathrm{red}(\mathcal{D}(n,3)) = 1.5\log n + \mathcal{O}(1).$$

■

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor Dr. Xiande Zhang and the two anonymous reviewers for their helpful comments.

REFERENCES

[1] S. Al-Bassam and B. Bose, "On balanced codes," *IEEE Trans. Inf. Theory*, vol. 36, no. 2, pp. 406–408, Mar. 1990.

[2] N. Alon, E. E. Bergmann, D. Coppersmith, and A. M. Odlyzko, "Balancing sets of vectors," *IEEE Trans. Inf. Theory*, vol. IT-34, no. 1, pp. 128–130, Jan. 1988.

[3] N. Alon, J. Bruck, F. F. Hassanzadeh, and S. Jain, "Duplication distance to the root for binary sequences," *IEEE Trans. Inf. Theory*, vol. 63, no. 12, pp. 7793–7803, Dec. 2017.

[4] R. Bar Yehuda, T. Etzion, and S. Moran, "Rotating-table games and derivatives of words," *Theor. Comput. Sci.*, vol. 108, no. 2, pp. 311–329, Feb. 1993.

[5] V. Becher and P. A. Heiber, "On extending de Bruijn sequences," *Inf. Process. Lett.*, vol. 111, no. 18, pp. 930–932, Sep. 2011.

[6] N. G. de Bruijn, "A combinatorial problem," *Koninklijke Nederlandse Akademie Wetenschappen*, vol. 49, pp. 758–764, Jun. 1946.

[7] W. Burnside, "On the continuous group that is defined by any given group of finite order," *Proc. London Math. Soc.*, vols. s1–29, no. 1, pp. 207–224, Nov. 1897.

[8] A. H. Chan, R. A. Games, and E. L. Key, "On the complexities of de Bruijn sequences," *J. Combinat. Theory, A*, vol. 33, no. 3, pp. 233–246, Nov. 1982.

[9] U. Cheng and S. W. Golomb, "On the characteristics of PN sequences (corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 4, p. 600, Jun. 1983.

[10] P. E. C. Compeau, P. A. Pevzner, and G. Tesler, "How to apply de Bruijn graphs to genome assembly," *Nature Biotechnol.*, vol. 29, no. 11, pp. 987–991, Nov. 2011.

[11] T. M. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 1, pp. 73–77, Jan. 1973.

[12] S. Dolinar, T.-M. Ko, and R. McEliece, "Some VLSI decompositions of the de Bruijn graph," *Discrete Math.*, vols. 106–107, pp. 189–198, Sep. 1992.

[13] T. Etzion, "The depth distribution—A new characterization for linear codes," *IEEE Trans. Inf. Theory*, vol. 43, no. 4, pp. 1361–1363, Jul. 1997.

[14] C. T. Fan, S. M. Fan, S. L. Ma, and M. K. Siu, "On de Bruijn arrays," *Ars Combinatoria*, vol. 19A, pp. 205–213, 1985.

[15] H. Fredricksen, "A class of nonlinear de Bruijn cycles," *J. Combinat. Theory, A*, vol. 19, no. 2, pp. 192–199, Sep. 1975.

[16] H. Fredricksen, "A survey of full length nonlinear shift register cycle algorithms," *SIAM Rev.*, vol. 24, no. 2, pp. 195–221, Apr. 1982.

[17] R. A. Games, "A generalized recursive construction for de Bruijn sequences," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 6, pp. 843–850, Nov. 1983.

[18] R. A. Games, "There are no de Bruijn sequences of span N with complexity 2^{n-1}+n+1," *J. Combinat. Theory, A*, vol. 34, no. 2, pp. 248–251, 1983.

[19] T. Goka, "An operator on binary sequences," *SIAM Rev.*, vol. 12, no. 2, pp. 264–266, Apr. 1970.

[20] S. W. Golomb, "On the classification of balanced binary sequences of period 2^{n-1}," *IEEE Trans. Inf. Theory*, vol. IT-26, no. 6, pp. 730–732, Nov. 1980.

[21] S. W. Golomb, *Shift Register Sequences*. Singapore: World Scientific, 2017.

[22] I. J. Good, "Normal recurring decimals," *J. London Math. Soc.*, vols. s1–21, no. 3, pp. 167–169, Jul. 1946.

[23] T. Hasunuma and Y. Shibata, "Embedding de bruijn, Kautz and shuffle-exchange networks in books," *Discrete Appl. Math.*, vol. 78, nos. 1–3, pp. 103–116, Oct. 1997.

[24] H. Hollmann and K. A. S. Immink, "Performance of efficient balanced codes," *IEEE Trans. Inf. Theory*, vol. 37, no. 3, pp. 913–918, May 1991.

[25] K. A. S. Immink, *Codes for Mass Data Storage Systems*. Denver, CO, USA: Shannon Foundation, 2004.

[26] H. M. Kiah, G. J. Puleo, and O. Milenkovic, "Codes for DNA sequence profiles," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3125–3146, Jun. 2016.

[27] D. E. Knuth, "Oriented subtrees of an arc digraph," *J. Combinat. Theory*, vol. 3, no. 4, pp. 309–314, Dec. 1967.

[28] D. Knuth, "Efficient balanced codes," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 1, pp. 51–53, Jan. 1986.

[29] L. L. Cam, "The central limit theorem around 1935," *Stat. Sci.*, vol. 1, no. 1, pp. 78–91, Feb. 1986.

[30] F. T. Leighton, M. Lepley, and G. L. Miller, "Layouts for the shuffle-exchange graph based on the complex plane diagram," *SIAM J. Algebr. Discrete Methods*, vol. 5, no. 2, pp. 202–215, Jun. 1984.

[31] A. Lempel, "On a homomorphism of the de Bruijn graph and its applications to the design of feedback shift registers," *IEEE Trans. Comput.*, vols. C–19, no. 12, pp. 1204–1209, Dec. 1970.

[32] S. Marcovich, T. Etzion, and E. Yaakobi, "Balanced de Bruijn sequences," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 1528–1533.

[33] U. M. Maurer, "Asymptotically-tight bounds on the number of cycles in generalized de Bruijn-good graphs," *Discrete Appl. Math.*, vols. 37–38, pp. 421–436, Jul. 1992.

[34] G. L. Mayhew and S. W. Golomb, "Characterizations of generators for modified de Bruijn sequences," *Adv. Appl. Math.*, vol. 13, no. 4, pp. 454–461, Dec. 1992.

[35] A. F. Möbius, *Über Eine Besondere Art von Umkehrung der Reihen*. 1832.

[36] F. J. Mowle, "Relations between PN cycles and stable feedback shift registers," *IEEE Trans. Electron. Comput.*, vols. EC–15, no. 3, pp. 375–378, Jun. 1966.

[37] F. J. Mowle, "An algorithm for generating stable feedback shift registers of order n," *J. ACM*, vol. 14, no. 3, pp. 529–542, Jul. 1967.

[38] V. Namias, "A simple derivation of Stirling's asymptotic series," *Amer. Math. Monthly*, vol. 93, no. 1, pp. 25–29, Jan. 1986.

[39] M. B. Nathanson, "Derivatives of binary sequences," *SIAM J. Appl. Math.*, vol. 21, no. 3, pp. 407–412, Nov. 1971.

[40] K. G. Paterson, "Perfect maps," *IEEE Trans. Inf. Theory*, vol. 40, no. 3, pp. 743–753, May 1994.

[41] A. Ralston, "A new memoryless algorithm for de Bruijn sequences," *J. Algorithms*, vol. 2, no. 1, pp. 50–62, 1981.

[42] R. M. Roth, P. H. Siegel, and A. Vardy, "High-order spectral-null codes-constructions and bounds," *IEEE Trans. Inf. Theory*, vol. 40, no. 6, pp. 1826–1840, Nov. 1994.

[43] F. Rusky, *Combinatorial Generation*. Victoria, BC, Canada: Univ. of Victoria, 1996.

[44] K. A. S. Immink and J. H. Weber, "Very efficient balanced codes," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 188–192, Feb. 2010.

[45] V. Skachek, T. Etzion, and R. M. Roth, "Efficient encoding algorithm for third-order spectral-null codes," *IEEE Trans. Inf. Theory*, vol. 44, no. 2, pp. 846–851, Mar. 1998.
[46] L. Song, F. Geng, Z. Gong, B. Li, and Y. Yuan, "Super-robust data storage in DNA by de Bruijn graph-based decoding," *bioRxiv*, to be published, doi: 10.1101/2020.12.20.423642.
[47] M. A. Sridhar and C. S. Raghavendra, "Fault-tolerant networks based on the de Bruijn graph," *IEEE Trans. Comput.*, vol. 40, no. 10, pp. 1167–1174, Oct. 1991.
[48] L. G. Tallini and B. Bose, "Balanced codes with parallel encoding and decoding," *IEEE Trans. Comput.*, vol. 48, no. 8, pp. 794–814, Aug. 1999.
[49] L. G. Tallini and B. Bose, "On efficient high-order spectral-null codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 7, pp. 2594–2601, Nov. 1999.
[50] L. G. Tallini, R. M. Capocelli, and B. Bose, "Design of some new efficient balanced codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 3, pp. 790–802, May 1996.
[51] T. van Aardenne-Ehrenfest and N. G. de Bruijn, "Circuits and trees in oriented linear graphs," *Simon Stevin, Wis-en Natuurkundig Tijdschrift*, vol. 28, pp. 203–217, Jan. 1951.
[52] J. H. Weber and K. A. S. Immink, "Knuth's balanced codes revisited," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1673–1679, Apr. 2010.

Sagi Marcovich (Member, IEEE) received the B.Sc. degree in software engineering and the M.Sc. degree in computer science from the Technion—Israel Institute of Technology, Haifa, Israel, in 2016 and 2021, respectively, where he is currently pursuing the Ph.D. degree with the Computer Science Department. His research interests include algorithms, information theory, and coding theory with applications to DNA-based storage.

Tuvi Etzion (Life Fellow, IEEE) was born in Tel Aviv, Israel, in 1956. He received the B.A., M.Sc., and D.Sc. degrees from the Technion—Israel Institute of Technology, Haifa, Israel, in 1980, 1982, and 1984, respectively.

In 1984, he held a position at the Department of Computer Science, Technion, where he is currently the Bernard Elkin Chair in computer science. From 1985 to 1987, he was a Visiting Research Professor at the Department of Electrical Engineering (Systems), University of Southern California, Los Angeles. During the summers of 1990 and 1991, he was visiting Bellcore, Morristown, NJ, USA. From 1994 to 1996, he was a Visiting Research Fellow at the Computer Science Department, Royal Holloway University of London, Egham, U.K. He also had several visits to the Coordinated Science Laboratory, University of Illinois Urbana–Champaign, from 1995 to 1998; two visits to HP Bristol during the summers of 1996 and 2000; a few visits to the Department of Electrical Engineering, University of California at San Diego, from 2000 to 2022; several visits to the Mathematics Department, Royal Holloway University of London, from 2007 to 2022; a few visits to the School of Physical and Mathematical Science (SPMS), Nanyang Technological University, and the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore, from 2016 to 2019, and 2022; and a few visits to Jiaotong University, Beijing, from 2017 to 2019. His research interests include applications of discrete mathematics to problems in computer science and information theory, coding theory, coding for storage, and combinatorial designs.

Dr. Etzion was an Associate Editor for Coding Theory for the IEEE TRANSACTIONS ON INFORMATION THEORY from 2006 to 2009. From 2004 to 2009, he was an Editor for the *Journal of Combinatorial Designs*. Since 2011, he has been an Editor for *Designs, Codes, and Cryptography*. He has also been an Editor for *Advances in Mathematics of Communications* since 2013. He has been an Editor-in-Chief for *Journal of Combinatorial Theory, Series A*, since 2021.

Eitan Yaakobi (Senior Member, IEEE) received the B.A. degree in computer science and mathematics and the M.Sc. degree in computer science from the Technion—Israel Institute of Technology, Haifa, Israel, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of California, San Diego, in 2011. From 2011 to 2013, he was a Post-Doctoral Researcher at the Department of Electrical Engineering, California Institute of Technology, and the Center for Memory and Recording Research, University of California, San Diego. He is currently an Associate Professor with the Computer Science Department, Technion—Israel Institute of Technology. His research interests include information and coding theory with applications to non-volatile memories, associative memories, DNA storage, data storage and retrieval, and private information retrieval. He received the Marconi Society Young Scholar in 2009 and the Intel Ph.D. Fellowship from 2010 to 2011.