Repeat-Free Codes

Ohad Elishco, *Member, IEEE*, Ryan Gabrys, *Member, IEEE*, Eitan Yaakobi, *Senior Member, IEEE*, and Muriel Médard, *Fellow, IEEE*

Abstract

In this paper we consider the problem of encoding data into *repeat-free* sequences in which sequences are imposed to contain any k-tuple at most once (for predefined k). First, the capacity and redundancy of the repeat-free constraint are calculated. Then, an efficient algorithm, which uses a single bit of redundancy, is presented to encode length-n sequences for $k = 2 + 2\log(n)$. This algorithm is then improved to support any value of k of the form $k = a\log(n)$, for $1 < a$, while its redundancy is $o(n)$. We also calculate the capacity of repeat-free sequences when combined with local constraints which are given by a constrained system, and the capacity of multi-dimensional repeat-free codes.

Index Terms

Information theory, DNA sequences, Error-correcting codes, Constrained coding, capacity, Encoder construction

I. Introduction

Repeat-free sequences represent a generalization of the well-known De-Bruijn sequences in which every length-k substring appears exactly once. De-Bruijn sequences have found applications in areas as diverse as cryptography, pseudo-randomness, and information hiding in wireless communications [1]. However, one potential drawback to adopting De-Bruijn sequences for representing information is that De-Bruijn sequences have rate at most $1/2$. In this work, we show that by relaxing the condition in which every k-tuple appears *exactly* once to appear *at most* once, we can generate codes of asymptotic rate 1 with efficient encoders and decoders for a variety of parameters.

One motivating application for this work is DNA storage, and, in particular, the reading process of a DNA string. The reading process of a DNA string is as follows. At first, the long string is fragmented into substrings of a shorter length which may be read properly. Then, a multiset of all the short strings is obtained in a form of their frequency. Then, the long DNA string should be reconstructed using only the knowledge of the shorter length substrings.

There are two common lines of work on DNA storage systems. The first assumes that the data is stored in a living organism. In this case, the major concern is to correct errors which are made by naturally occurring mutations. For analysis of the capacity of mutation strings, see [2]–[6] and [7]–[11] for coding and algorithms related works. The second line of work focuses on data storage outside a living organism and is called coding for string reconstruction. The goal of coding for the string reconstruction problem is to encode arbitrary strings into ones that are uniquely reconstructible. This problem is motivated by the reading process of DNA-based data storage, where the stored strings are to be reconstructed from information about substrings appearing in the stored string. This problem motivated a series of papers regarding decoding of sequences from partial information on their substrings [12]–[20].

In order to ensure unique reconstruction, studies were made on reconstruction of encoded sequences [13], [21], [22]. One method that guarantees a unique reconstruction is to encode the information sequence to a codeword that does not contain any k-tuple more than once. For two positive integers $k < n$, we say that a length-n word w is a *k-repeat free word* if every subword of w of length k appears at most once. It is already known that k-repeat free words are uniquely reconstructible from their length-r substrings multiset if $r \geqslant k+1$ [23]. Furthermore, an encoding scheme that exploits this property has been recently proposed in [20]; however, the encoded words are not strictly repeat free. Thus, studying the *repeat-free constraint* and designing respective efficient encoding and decoding schemes is still an open research problem, which is the primary focus of this paper.

Another important characteristic of the k-repeat free sequences is the growth rate of the number of sequences as a function of the length of the sequence. Arguably, one of the most well known families of k-repeat free sequences are De-Bruijn sequences of span k which play an important role in this paper. A De-Bruijn sequence of span k is a sequence over a finite alphabet, in which every k-tuple appears exactly once. It is clear that every De-Bruijn sequence of span k over an alphabet of size q (which implies that the sequence is of length $q^k + k - 1$) is k-repeat free [24]. For the case of De-Bruijn sequences, a closed formula for the number of De-Bruijn sequences of length $q^k + k - 1$ exists [1], [24]. Unfortunately, there is no such formula for the general set of k-repeat free sequences. It is clear that a sequence of length n over an alphabet of size q cannot be k-repeat free if $k < \log_q n$. However, the size of k-repeat free sequences with $k = a\log_q n$ with $a > 1$ has not been fully determined.

O. Elishco and M. Médard are with Massachusetts Institute of Technology, Cambridge, MA, 02139 (e-mail: {ohadeli,medard}@mit.edu).

R. Gabrys is with Spawar Systems Center, San Diego, San Diego, CA, 92115 (e-mail: ryan.gabrys@navy.mil).

E. Yaakobi is with the Department of Computer Science, Technion — Israel Institute of Technology, Haifa 32000, Israel (e-mail: yaakobi@cs.technion.ac.il).

This paper was presented in part at the IEEE International Symposium on Information Theory (ISIT 2019), Paris, France.

Using union bound arguments it is straightforward to show that the growth rate of the number of k-repeat free sequences is q^n when $k = \lceil a \log_q n \rceil$ and $a \geqslant 2$. On the other hand, from the known enumeration results of De-Bruijn sequences [1], [24] it follows that the growth rate in the binary case is at least $2^{n/2}$ for $a = 1$. Therefore, it is left to find the growth rate for $1 < a < 2$. By carefully calculating the probability that a word has two identical length-k subsequences, we show in this paper that the growth rate is 2^n for all $a > 1$.

Motivated by several previous works [20], [25], [26], we address the problem of calculating the capacity of k-repeat free sequences of length n where $k = a\log(n)$ with $a > 1$. We provide an efficient encoding algorithm that encodes into k-repeat free binary words for $k = 2\lceil \log(n) \rceil + 2$ with only a single redundancy bit. We also extend this algorithm to the setup where $\log(n) < k < 2\log(n)$ with asymptotically rate-one algorithm. Both algorithms operate in two phases; in the first phase the information sequence is compressed into some shorter sequence that satisfies the constraint and afterwards this compressed sequence is expanded to ensure that the final output is of length n and yet satisfies the constraint. We also study the capacity of k-repeat free sequences which satisfy local constraints as well. For example, a combination of the k-repeat free constraint and the no-adjacent-zeros constraint (i.e., the $(0,1)$-run-length-limited constraint). Perhaps surprisingly, we show that the k-repeat free constraint does not impose a rate penalty in either case.

The capacity results are also generalized to the multidimensional case. While the number of binary De-Bruijn sequences of span k is known, in the multidimensional case, the situation is much more complicated. The analog definition of a De-Bruijn sequence to a multidimensional scenario is called a *De-Bruijn torus*. Not only that the number of De-Bruijn tori is not known, it is not known for which sizes there exists a De-Bruijn torus [27]–[29].

The rest of the paper is organized as follows. In Section II, we present the notation and definitions which are used throughout the paper together with the definition of k-repeat free sequences. In Section III, we present our first result which asserts that the capacity of k-repeat free sequences for $k = a\log(n)$ is 1 whenever $a > 1$. In Section IV, we present an encoding algorithm for binary sequences of length n with $k = 2\lceil \log(n) \rceil + 2$ and a single bit of redundancy. Next, an encoding algorithm for $k = a\log(n)$ with $1 < a \leqslant 2$ is presented in Section V. In Section VI, we calculate the capacity of k-repeat free sequences which also satisfy local constraints. In Section VII, we generalize the capacity result for d-dimensional k-repeat free arrays. We conclude in Section VIII.

II. Preliminaries

Let $\mathbb{N}$ denote the set of natural numbers. For $n \in \mathbb{N}$, we denote by $[n]$ the set $[n] = \{0, 1, \ldots, n-1\}$ and by $[-n]$ the set $[-n] = \{-1, -2, \ldots, -n\}$. For a set A we use $|A|$ to denote the size of A. If A is a subset of a group with a group operation $\bullet$, and if b is any group member, we define $b \bullet A \triangleq \{b \bullet a \ : \ a \in A\}$.

Example 1. *Let $A = \{1, 3, 5, 6\} \subseteq \mathbb{Z}$ and let $b = -1$. Then,*

$$A + b = \{0, 2, 4, 5\}, \qquad A \cdot b = \{-1, -3, -5, -6\}.$$

□

Throughout the paper, we use Σ to denote a finite alphabet. A word of length n over Σ, $w = (w_0, \ldots, w_{n-1})$ is a sequence of n symbols from Σ and is defined as a function from $[n]$ to Σ. We denote by Σ^n the set of all functions from $[n]$ to Σ and by $\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$. For a word $w \in \Sigma^*$, $|w|$ denotes the length of w (i.e., the domain of the function w) and $w_i = w(i)$ is simply the ith symbol in w.

Definition 2. *Let A, B be two sets and let $f : A \to B$ be any function. For a subset $A' \subseteq A$, we denote by $f_{A'} : A' \to B$ the restriction of f to A'.*

Since we consider words as functions, for a word $w \in \Sigma^n$ and for a set $A \subseteq [n]$, w_A denotes the restriction of w to the set A. In other words, w_A is a word created by taking the symbols from w that appear in the positions in A. We say that u is a subword or subsequence of w if there exists $i \in \mathbb{N}$ such that $w_{i+[|u|]} = (w_i, \ldots, w_{i+|u|-1}) = u$. If $w, u \in \Sigma^*$ we denote by $wu \in \Sigma^{|w|+|u|}$ the concatenation of w and u. We will also use the symbol $w \circ u$ when we would like to emphasize the distinct parts. For $w \in \Sigma^*$ we write w^ℓ for the concatenation of w with itself $\ell \in \mathbb{N}$ times. Unless otherwise is mentioned, coordinates of a word $w \in \Sigma^*$ are considered modulo $|w|$. Thus, if $w, u \in \Sigma^*$, we have $wu_{[|w|]} = w$ and $wu_{[-|u|]} = u$.

The main object studied in this paper is a set of words which we call a *system*. Specifically, we focus on systems which are defined using global constraints. One of the main characterizations of a system is given by the number of feasible words of length n. To be more specific, we would like to estimate the rate at which the number of length-n words grows with n. This value is called the *capacity* of the system and is defined as follows.

Definition 3. *Let $\mathcal{L} \subseteq \Sigma^*$ be a system. The **capacity** of $\mathcal{L}$ is denoted by $\mathsf{cap}(\mathcal{L})$ and is defined as*

$$\mathsf{cap}(\mathcal{L}) \triangleq \limsup_{n \to \infty} \frac{1}{n} \log_{|\Sigma|} |\mathcal{L} \cap \Sigma^n|.$$

In case $q = 2$ we will sometime simply write log instead of $\log_2$.

The systems we consider will be defined mostly using constraints on the number of subword appearances. To this end, we define the notion of empirical frequency.

Definition 4. *Let $w \in \Sigma^n$ and $k \leqslant n$. The* ***empirical frequency*** *of k-tuples in w is denoted by fr_w^k and is defined as follows. For a k-tuple, $u \in \Sigma^k$,*

$$\mathrm{fr}_w^k(u) \triangleq \frac{1}{(n-k+1)} \sum_{m \in [n-k+1]} \mathbb{1}_u \left(w_{m+[k]} \right),$$

where $\mathbb{1}$ denotes the indicator function defined by $\mathbb{1}_a(b) = 1$ if $b = a$ and 0 otherwise. We will sometimes consider fr_w^k as a vector of length $|\Sigma|^k$ or as a probability distribution.

The *support* of fr_w^k, denoted by $\mathrm{Supp}(\mathrm{fr}_w^k)$, is the set of all k-tuples which appear in w.

Example 5. *Let Σ be the binary alphabet and let $w = (11001010), v = (00111010) \in \Sigma^8$. For $k = 2$, the empirical frequency of the pairs in w, v is given by $\mathrm{fr}_w^2, \mathrm{fr}_v^2$, respectively. We have that*

$$\mathrm{fr}_w^2(01) = \frac{2}{7}, \ \ \mathrm{fr}_w^2(10) = \frac{3}{7}, \ \ \mathrm{fr}_w^2(11) = \mathrm{fr}_w^2(00) = \frac{1}{7}$$

and

$$\mathrm{fr}_v^2(01) = \mathrm{fr}_v^2(10) = \mathrm{fr}_v^2(11) = \frac{2}{7}, \ \ \mathrm{fr}_v^2(00) = \frac{1}{7}.$$

Both w, v have full support, i.e.,

$$\mathrm{Supp}(\mathrm{fr}_w^2) = \mathrm{Supp}(\mathrm{fr}_v^2) = \Sigma^2,$$

but $\mathrm{Supp}(\mathrm{fr}_w^5) = \{11001, 10010, 00101, 01010\}$ *and* $\mathrm{Supp}(\mathrm{fr}_v^5) = \{00111, 01110, 11101, 11010\}$. □

One of the most important sets of words related to this work is the set of (one-dimensional) De-Bruijn sequences [24]. We follow the non-cyclic definition of De-Bruijn sequences and for a finite alphabet Σ, and for $1 \leqslant k \in \mathbb{N}$, we say that a word w is a *De-Bruijn word of span k* if every k-tuple appears in w exactly once. Note that w must be of length $|\Sigma|^k + k - 1$ (where the $(k-1)$-suffix equals to the $(k-1)$-prefix), since there are exactly $|\Sigma|^k$ different k-tuples. Using our notation, we define the following system.

Definition 6. *A word $w \in \Sigma^*$ is called a* ***De-Bruijn sequence of span*** *k if every k-tuple appears exactly once, i.e., for every $u \in \Sigma^k$,*

$$\mathrm{fr}_w^k(u) = \frac{1}{|w| - k + 1}.$$

The ***De-Bruijn system*** *over the alphabet Σ with $|\Sigma| = q$ is denoted by $\mathcal{B}_q$ and is defined as the set of all De-Bruijn sequences (over Σ) of span k for some $1 \leqslant k \in \mathbb{N}$. In a notational form, a De-Bruijn system over Σ is the set*

$$\mathcal{B}_q = \left\{ w \in \Sigma^* \ : \ \exists k \in \mathbb{N} \ s.t. \ \forall u \in \Sigma^k, \ \ \mathrm{fr}_w^k(u) = \frac{1}{|w| - k + 1} \right\}.$$

Note that by definition, a De-Bruijn system contains all the De-Bruijn sequences of span k, for some $k \in \mathbb{N}$. In fact, a De-Bruijn system contains words of lengths $|\Sigma|^k + k - 1$ for some k.

The number of binary De-Bruijn sequences of span k is known due to De-Bruijn himself who used the doubling process to calculate the exact number [24]. Later, his result was generalized to any alphabet [1]. For a finite alphabet Σ with $|\Sigma| = q$, the number of De-Bruijn sequences of span k is given by

$$((q-1)!)^{q^{k-1}} \cdot q^{q^{k-1} - k}.$$

Using this formula, the capacity of the De-Bruijn system can be calculated as follows.

$$\begin{aligned}
\mathsf{cap}(\mathcal{B}_q) &= \limsup_{n \to \infty} \frac{1}{n} \log_q |\mathcal{B}_q \cap \Sigma^n| \\
&\leqslant \limsup_{k \to \infty} \frac{1}{q^k + k - 1} \log_q \left(((q-1)!)^{q^{k-1}} \cdot q^{q^{k-1} - k} \right) \\
&= \frac{1}{q} (\log_q (q!)).
\end{aligned}$$

On the other hand, for lengths $n \neq q^k + k - 1$ for some k, we have that $|\mathcal{B}_q \cap \Sigma^n| = 0$ which implies $\frac{1}{n} \log_q |\mathcal{B}_q \cap \Sigma^n| = -\infty$. Hence, $\mathsf{cap}(\mathcal{B}_q) = \frac{1}{q}(\log_q (q!))$. Note that when $q = 2$, $\mathsf{cap}(\mathcal{B}_2) = 1/2$ but using Stirling's approximation we obtain that $\lim_{q \to \infty} \mathsf{cap}(\mathcal{B}_q) = 1$.

III. Capacity of k-Repeat Free Systems

In this section we introduce the first system we will consider in this work and calculate the capacity of the system. One may regard this system as a generalization of De-Bruijn systems. Throughout this section and unless stated otherwise, we let Σ be a fixed alphabet of size q.

Definition 7. *A sequence $w \in \Sigma^n$ is said to be **k-repeat free** (or, interchangeably, **weak De-Bruijn of span** k) if every k-tuple appears at most once as a subword in w. The set of length-n k-repeat free sequences is denoted by*

$$\mathcal{W}_k(n) \triangleq \left\{ w \in \Sigma^n \ : \ \forall u \in \Sigma^k,\ \mathrm{fr}_w^k(u) \leqslant \frac{1}{n-k+1} \right\}.$$

*For any k, we define the **k-repeat free system (weak De-Bruijn system)** as $\mathcal{W}_k = \bigcup_{n\in\mathbb{N}} \mathcal{W}_k(n)$.*

Note that if $n = q^k + k - 1$ then $\mathcal{W}_k(n)$ is exactly the set of all De-Bruijn sequences of span k. On the other hand, if $n > q^k + k - 1$ then $\mathcal{W}_k(n) = \varnothing$ since there are more subwords than k-tuples. This implies that for any fixed k we have

$$\mathsf{cap}(\mathcal{W}_k) = \limsup_{n\to\infty} \frac{1}{n} \log_q |\mathcal{W}_k(n)| = -\infty.$$

Therefore, a more natural question to ask is how the size $|\mathcal{W}_k(n)|$ behaves when k and n grow together. Namely, we are interested in the set $\mathcal{W}_k(n)$ where $k > \log(n-k+1)$ and is a function of n. We will calculate the capacity of a k-repeat free system for $k = \left\lfloor a \log_q(n) \right\rfloor$ with $a > 1$. Under this scenario, we will also denote $\mathcal{W}_k(n)$ as $\mathcal{W}_a(n)$ and $\mathcal{W}_k$ as $\mathcal{W}_a$. That is, $\mathcal{W}_a(n) = \mathcal{W}_{\left\lfloor a\log_q(n)\right\rfloor}(n)$ and $\mathcal{W}_a = \bigcup_{n\in\mathbb{N}} \mathcal{W}_{\lfloor a \log(n)\rfloor}(n)$.

The size $|\mathcal{W}_a(n)|$ will be estimated by a probabilistic approach. Consider the uniform distribution over all length-n sequences, then $|\mathcal{W}_a(n)| = |\Sigma|^n \cdot \Pr(\mathcal{W}_a(n))$. Then, the capacity in this case is given by

$$\mathsf{cap}(\mathcal{W}_a) = 1 + \limsup_{n\to\infty} \frac{1}{n} \log_q \left(\Pr(\mathcal{W}_a(n))\right). \tag{1}$$

Using simple union bound arguments, it is possible to show that for $a \geqslant 2$, $\mathsf{cap}(\mathcal{W}_a) = 1$. However, in the following theorem we apply a different method which assures that this capacity result holds for all $a > 1$.

Theorem 8. *Let Σ be a finite alphabet of size q then for all $a > 1$, $\mathsf{cap}(\mathcal{W}_a) = 1$.*

Proof: First note that the capacity is upper bounded by 1 and hence we only need to lower bound the capacity. Let w be an infinite sequence such that the symbol in each coordinate is chosen uniformly at random over Σ. Note that in order to estimate $|\mathcal{W}_a(n)|$, we may estimate the probability $\Pr(w_{[n]} \in \mathcal{W}_a(n))$ since $\Pr(w_{[n]} \in \mathcal{W}_a(n)) = \frac{|\mathcal{W}_a(n)|}{q^n}$. For $0 < \ell \in \mathbb{N}$, define the following random variable

$$X_\ell = \sum_{j=(\ell-1)k+1}^{\ell k} \sum_{i=0}^{j-1} \mathbb{1}_{w_{j+[k]}}(w_{i+[k]}),$$

where $k = \left\lfloor a \log_q n \right\rfloor$. For a fixed j, $\sum_{i=0}^{j-1} \mathbb{1}_{w_{j+[k]}}(w_{i+[k]})$ counts the number of times the jth k-tuple appears in w. We have that

$$\Pr\left(w_{[(n+1)k]} \in \mathcal{W}_a\right) = \Pr\left(\sum_{\ell=1}^{n} X_\ell < 1\right) = 1 - \Pr\left(\sum_{\ell=1}^{n} X_\ell \geqslant 1\right).$$

Moreover, for every $0 < \ell \in \mathbb{N}$ we have that

$$\Pr\left(w_{[(\ell+1)k]} \in \mathcal{W}_a \ \middle|\ w_{[\ell k]} \in \mathcal{W}_a\right) = 1 - \Pr\left(X_\ell \geqslant 1\right). \tag{2}$$

We will now bound the probability $\Pr(X_\ell \geqslant 1)$ as follows.

$$\begin{aligned}
\mathbb{E}[X_\ell] &= \sum_{j=(\ell-1)k+1}^{\ell k} \sum_{i\in[j]} \Pr\left(w_{j+[k]} = w_{i+[k]}\right) \\
&= \sum_{j=(\ell-1)k+1}^{\ell k} \sum_{i\in[j]} \frac{1}{q^k} = \frac{1}{q^k} \sum_{j=(\ell-1)k+1}^{\ell k} j \\
&\leqslant \frac{1}{q^k}(\ell k^2 + k) \leqslant \frac{(\ell+1)k^2}{q^k}.
\end{aligned}$$

Using Markov inequality we obtain that $\Pr(X_\ell \geqslant 1) \leqslant \frac{(\ell+1)k^2}{q^k}$. Hence, using (2), we get that

$$\Pr\left(w_{[(\ell+1)k]} \in \mathcal{W}_a \;\middle|\; w_{[\ell k]} \in \mathcal{W}_a\right) \geqslant 1 - \frac{(\ell+1)k^2}{q^k}.$$

This implies that

$$\Pr(\mathcal{W}_k(n)) \geqslant \prod_{\ell=0}^{\lfloor n/k \rfloor} \left(1 - \frac{\ell k^2}{q^k}\right).$$

Since $a > 1$, we get that for n large enough, $\left(1 - \frac{\ell k^2}{q^k}\right)$ is positive for every ℓ. Taking logarithm and exponent from both sides of the inequality we have that

$$\Pr(\mathcal{W}_k(n)) \geqslant q^{\sum_{\ell=0}^{\left\lfloor \frac{n}{k} \right\rfloor} \log_q\left(1 - \frac{\ell k^2}{q^k}\right)}.$$

Using the fact that for every $x \in (0,1)$, $\log_q(1-x) \geqslant -\frac{x}{(1-x)\ln q}$ we deduce

$$\Pr(\mathcal{W}_k(n)) \geqslant q^{-\sum_{\ell=0}^{\left\lfloor \frac{n}{k} \right\rfloor} \frac{\ell k^2}{(q^k - \ell k^2)\ln q}} \geqslant q^{-\left(\frac{n}{k}+1\right)\frac{nk}{(q^k - nk)\ln q}}, \tag{3}$$

where the last inequality follows by taking the largest argument in the sum $\frac{n}{k}+1$ times. Using $k = \lfloor a \log_q n \rfloor$ and writing $a = 1+\epsilon$ we obtain

$$\Pr(\mathcal{W}_k(n)) \geqslant q^{-\left(\frac{n}{k}+1\right)\frac{a\log_q n}{(n^\epsilon - a\log_q n)\ln q}}.$$

Taking logarithm and dividing by n we receive the following inequality

$$\frac{1}{n}\log_q(\Pr(\mathcal{W}_k(n))) \geqslant -\frac{1}{(n^\epsilon - a\log_q n)\ln q} - \frac{a\log_q n}{n(n^\epsilon - 1)\ln q}, \tag{4}$$

which clearly goes to 0 as $n \to \infty$. Thus, plugging it in (1), we obtain that $\mathsf{cap}(\mathcal{W}_a) = 1$ for $a > 1$. ■

Remark 9. *Note that using* (4)*, if we write $a = 1+\epsilon$, we can also obtain that the redundancy is of order $O(n^{1-\epsilon})$. This implies that for $k = 2\log_q n$, the redundancy is some constant number.*

IV. Algorithm For $k = 2\log(n) + 2$

In this section, we provide a coding algorithm for the binary weak De-Bruijn system with $k = 2\log(n)+2$, where for simplicity we assume that n is a power of 2. This will be the basic step towards an algorithm for the case $k = a\log(n)$ with $a > 1$ that will be presented in Section V. First note that according to Remark 9, $a \geqslant 2$ implies that the redundancy is a constant. The input is a binary sequence $w \in \Sigma^{n-1}$, where in this section $\Sigma = \{0,1\}$. The output is a k-repeat-free sequence $\overline{w} \in \mathcal{W}_k(n)$. We first give a short overview of the algorithm, which is divided into two procedures: *elimination* and *expansion*. Given a sequence $w \in \Sigma^{n-1}$, append $10^{1+\log(n)}$ to its end. Then, search for identical subsequences of length $2\log(n)+2$. For every such an occurrence, remove one of them (the first one) and encode at the beginning of the sequence the indices of these two subsequences followed by the bit 0. Note that such an operation reduces the length of the sequence by one, and therefore this procedure is guaranteed to terminate. The second procedure takes this compressed sequence and decompresses it into a longer sequence such that the constraint is not violated. At the end, the output is the first n bits of the decompressed sequence.

Before presenting the algorithm, we need a few more notations. For an integer $i \in [n]$, we let $\boldsymbol{b}(i)$ be its binary representation using $\log(n)$ bits. Let $w \in \Sigma^n$ be any word. Recall that for $i \in \mathbb{N}$, $i \leqslant |w|$, $w_{[-i]}$ is the length-i suffix of w, i.e., $w_{[-i]} = w_{|w|-i+[i]}$. Moreover, the support of fr_w^k, $\mathrm{Supp}(\mathrm{fr}_w^k)$, is the set of all k-tuples that appear in w. For a word $w \in \Sigma^n$ and for $m \in \mathbb{N}$ we denote by $\mathrm{Cr}_m(w)$ the word of length m created by repeatedly concatenating w to itself and taking the length-m prefix, i.e., $\mathrm{Cr}_m(w) = (w^{\mathbb{N}})_{[m]}$. We say that a sequence $w \in \Sigma^*$ is ***ℓ-zero-constrained*** if there are no all-zeros substrings of length ℓ. We say that (i,j) (where $i < j$) is a ***k-identical window*** in w if $w_{i+[k]} = w_{j+[k]}$. If (i,j) is such that for any other k-identical window at (i',j') in w, we have $j \leqslant j'$, we say that (i,j) is a ***primal k-identical window***. The full details appear in Algorithm 1.

We now show the correctness of Algorithm 1. Notice that the first while loop ends since after every iteration either the length of the word $\overline{w}$ decreases by one (case 1) or its Hamming weight increases (case 2). Moreover, in Step 12, the word $\overline{w}$ has no identical length-k windows and has no $0^{\log(n)+1}$-window besides the one at its end, i.e., the word $\overline{w}_{[|w|-1]}$ is $(\log(n))$-zero-constrained. We start with the following lemma.

Algorithm 1 No-Identical Windows Encoding

Input: Sequence $w \in \Sigma^{n-1}$
Output: Sequence $\overline{w} \in \mathcal{W}_k(n)$ with $k = 2\log(n) + 2$
First procedure (elimination):
1: Set $\overline{w} = w \circ 1 \circ 0^{\log(n)+1} \in \Sigma^{n+\log(n)+1}$
2: **while** (i, j) is a k-identical windows in $\overline{w}$ or $\overline{w}_{[|\overline{w}|-1]}$ is not a $\log(n)$-zero-constrained (check the 1st condition first) **do**
3: **Case 1**: (there are identical length-k windows in $\overline{w}$)
4: Let (i, j) be a primal k-identical window in $\overline{w}$
5: Set $\overline{w} = \overline{w}_{[i]} \circ \overline{w}_{i+k+[|\overline{w}|-k-i]}$ (remove the first length-k repeated window from $\overline{w}$)
6: Set $\overline{w} = 0 \circ \boldsymbol{b}(i) \circ \boldsymbol{b}(j) \circ \overline{w}$ (append $0 \circ \boldsymbol{b}(i) \circ \boldsymbol{b}(j)$ to the left of $\overline{w}$)
7: **Case 2**: ($\overline{w}_{[|\overline{w}|-1]}$ is not a $\log(n)$-zero-constrained)
8: Let i be the index of the $0^{\log(n)+1}$-window in $\overline{w}$
9: Set $\overline{w} = \overline{w}_{[i]} \circ \overline{w}_{i+k-1+[|\overline{w}|-i-k]}$ (remove the $0^{\log(n)+1}$-window from $\overline{w}$)
10: Set $\overline{w} = 1 \circ \boldsymbol{b}(i) \circ \overline{w}$ (append $1 \circ \boldsymbol{b}(i)$ to the left of $\overline{w}$)
11: **end while**
12: **if** $|\overline{w}| \geqslant n$ **then**
13: Return $\overline{w}_{[n]}$
14: **end if**
Second procedure (expansion):
15: **while** $|\overline{w}| < n$ **do**
16: Set
$$B = \mathrm{Supp}\left(\mathrm{fr}_{\overline{w}}^{\log(n)}\right) \bigcup \bigcup_{1 \leqslant i \leqslant \log(n)-1} \mathrm{Cr}_n(\overline{w}_{[-i]}).$$
17: Set $S = \Sigma^{\log(n)} \backslash B$ and find $u \in S$
18: Set $\overline{w} = \overline{w} \circ u$ (append u to the right of $\overline{w}$)
19: **end while**
20: Return $\overline{w}_{[n]}$

Lemma 10. *In Step 12, the vector $\overline{w}$ ends with the sequence $1 \circ 0^{\log(n)+1}$.*

Proof: For any iteration of the first while loop for which there are two identical windows of length k in $\overline{w}$, let i and j be their indices, where $i < j$. We claim that the value of i satisfies $i \leqslant |\overline{w}| - 3\log(n) - 2$ and thus the last $\log(n) + 1$ bits of the vector $\overline{w}$ are not removed. Assume in the contrary that $|\overline{w}| - 3\log(n) - 2 < i < j$. Then, the length-$k$ window starting at position i has a 1 in its $(|\overline{w}| - \log(n) - i)$-th position while the length-k window starting at position j has a 0 in this position, which is a contradiction. It is also readily verified that the sequence $1 \circ 0^{\log(n)+1}$ cannot be removed as part of a removal of a $0^{\log(n)+1}$-window. ■

Lemma 11. *If the condition in Step 12 holds, then the returned vector is of length n and has no identical windows of length k.*

Proof: This lemma follows directly from Lemma 10 and Step 13. ■

Lemma 12. *For every iteration of the second while loop, the set S in Step 17 is not empty.*

Proof: Note that the size of the set B is at most $(|\overline{w}| - \log(n) + 1) + (\log(n) - 1) = |\overline{w}| < n$ and hence $B \neq \Sigma^{\log(n)}$. ■

Lemma 13. *For every iteration of the second while loop, in Step 18 the new vector $\overline{w}' = \overline{w} \circ u$ contains the sequence u exactly once at its end.*

Proof: According to the construction of the set B, the sequence u can appear in $\overline{w}' = \overline{w} \circ u$ only as a subsequence starting at positing j, where $|\overline{w}| - \log(n) + 1 \leqslant j \leqslant |\overline{w}| - 1$. Assume in contrary that there exists a value j such that $(\overline{w}')_{j+[\log(n)]} = (\overline{w} \circ u)_{j+[\log(n)]} = u$. But this implies that $u \in \mathrm{Cr}_n(\overline{w}_{[-i]})$ for some $1 \leqslant i \leqslant \log(n) - 1$ which is a contradiction to the construction of the set B in Step 16. ■

Let $\overline{w}_0$ be the value of the vector $\overline{w}$ after Step 14 and $n_0 = |\overline{w}_0|$ is its length. Assume that there are ℓ iterations of the second while loop, so the value of the vector $\overline{w}$ after Step 19 is given by

$$\overline{w} = \overline{w}_0 \circ u_1 \circ u_2 \circ \cdots \circ u_\ell,$$

where $u_1, u_2, \ldots, u_\ell$ are the vectors which were appended to the right of the vector u at each iteration of the while loop.

Lemma 14. *For $1 \leqslant i \leqslant \ell$, the vector $\overline{w}_i = \overline{w}_0 \circ u_1 \circ u_2 \circ \cdots \circ u_i$ has no identical length-k windows.*

Proof: We prove the lemma's statement by induction on the values of i. For the base case, we start with $i = 1$ and show that the vector $\overline{w}_1 = \overline{w}_0 \circ u_1$ has no identical length-k windows.

Assume in the contrary that (i, j) is a k-identical window. We only need to consider the cases where at least one of these two windows overlaps with u_1. This implies that the length-k window starting at position j overlaps with u_1, that is,

$$n_0 - k + 1 \leqslant j \leqslant n_0 + \log(n) - k.$$

In particular, the window $(\overline{w}_1)_{j+[k]}$ contains the $0^{\log(n)+1}$-window at the end of $\overline{w}_0$. If $i \leqslant n_0 - k$, then according to Lemma 10, $(\overline{w}_1)_{i+[k]}$ does not contain a $0^{\log(n)+1}$-window, which is a contradiction. Thus we only need to consider the case $n_0 - k \leqslant i < j \leqslant n + \log(n) - k$. However, this implies that $(\overline{w}_1)_{j+[k]}$ is periodic with period $0 \leqslant j - i \leqslant \log(n) - 1$ which is impossible since it contains the pattern $1 \circ 0^{\log(n)} + 1$.

Next we prove the statement for $\overline{w}_2 = \overline{w}_0 \circ u_1 \circ u_2$. According to the induction assumption we only need to consider values of i and j such that there is an overlap with u_2. Hence,

$$j \leqslant n_0 + 2\log(n) - k = n_0 - 2,$$
$$j \geqslant n_0 + \log(n) - k + 1 = n_0 - \log(n) - 1.$$

In particular, the window $(\overline{w}_2)_{j+[k]}$ contains u_1 as a subsequence. However, since $(\overline{w}_2)_{j+[k]} = (\overline{w}_2)_{i+[k]}$ we get that the sequence u_1 appears one more time in $\overline{w}_0 \circ u_1$, which is a contradiction to Lemma 13.

Next we assume that the lemma's statement holds for $\overline{w}_i$ and prove that it holds for $\overline{w}_{i+1}$, where $1 \leqslant i < \ell$. According to the induction assumption we only need to consider values of i and j such that there is an overlap with u_{i+1}. Hence,

$$n_0 + i\log(n) - k + 1 \leqslant j \leqslant n_0 + (i+1)\log(n) - k.$$

In particular, the window $(\overline{w}_{i+1})_{j+[k]}$ starting at index j contains the sequence u_i. However, since $(\overline{w}_{i+1})_{j+[k]} = (\overline{w}_{i+1})_{i+[k]}$ we get that the sequence u_i appears one more time in $\overline{w}_i$, which is a contradiction to Lemma 13. ■

Theorem 15. *Algorithm* 1 *successfully returns a k-repeat free sequence.*

Proof: In case the condition in Step 12 holds then according to Lemma 11, Algorithm 1 returns a sequence with no identical length-k windows. Otherwise, this claim holds from Lemma 14. ■

Note that there may be two identical length-k windows which intersect, i.e., (i, j) is a k-identical window with $j - i < k$. In this case, Step 5 in the algorithm suggests to remove the first length-k repeated window. This will not cause any problem since if (i, j) is such a k-identical window, then it implies that $w_{i+[k]}$ is a periodic sequence with period $j - i$ and as such can be obtained from the remaining bits. Nevertheless, this should be taken into account in the decoding process that is described next.

The decoding procedure is relatively simple. Look first for the left most sequence of $1 \circ 0^{\log(n)+1}$. According to Algorithm 1, everything to the right of this sequence was added during the expansion procedure and hence it can be removed. If there is no such $1 \circ 0^{\log(n)}$ window, look for the right-most 1. Since the algorithm returns a sequence which is longer by 1 than the input sequence, the right-most 1 (and the zeros following that 1) is a part of the initial set-up of the algorithm. Next, if the first symbol is 1, let i be the position indicated by the $(\overline{w})_{1+[\log(n)]}$, i.e., $\boldsymbol{b}(i) = (\overline{w})_{1+[\log(n)]}$. Delete the first $\log(n) + 1$ bits and enter $0^{\log(n)+1}$ in the ith position. If the first symbol is 0, let i and j be the positions indicated by $(\overline{w})_{1+[\log(n)]}$ and by $(\overline{w})_{1+\log(n)+[\log(n)]}$, respectively. Let $u = (\overline{w})_{j-1+[k]}$, delete the first $2\log(n) + 1$ bits, and put u in the ith position. Repeat this process until obtaining a sequence of length n.

Example 16. *Let $n = 32$ ($k = 2\log(32) + 2 = 12$) and*

$$w = 0100100110110010010011011100110 \in \Sigma^{31}.$$

The first step of the algorithm appends 1000000 *to the end of w and we obtain*

$$\overline{w} = 0100100110110010010011011100110\;1000000.$$

We now look for identical windows of length 12. *We see that $(\overline{w})_{[12]} = (\overline{w})_{13+[12]}$, i.e., $(0, 13)$ is a k-identical window. We eliminate the first* 12 *bits and we append $0 \circ \boldsymbol{b}(0)\boldsymbol{b}(13) = 00000001101$ to the left of $\overline{w}$. Hence,*

$$\overline{w} = 00000001101\;0010010011011100110\;1000000.$$

There are no more identical length-k windows in $\overline{w}$, but the pattern 000000 *appears in $\overline{w}$ in the* 0*th position. Thus, we eliminate the pattern and append $1\boldsymbol{b}(0) = 100000$ to the left, which yields the sequence*

$$\overline{w} = 100000\;01101\;0010010011011100110\;1000000.$$

Again, there is a sequence of 6 *zeros starting in position* 1 *so we delete this pattern and append $1\boldsymbol{b}(1) = 100001$ to the left, so we get that*

$$\overline{w} = 100001\;1\;1101\;0010010011011100110\;1000000.$$

Now $\overline{w}$ has no identical windows of length k and no $0^{\log(n)+1}$ except the one at the end. Moreover, $|\overline{w}| \geqslant 32$ hence the algorithm output is

$$\overline{w} = 100001\ 1\ 1101\ 0010010011011100110\ 10.$$

We now start the decoding process in order to retrieve w from $\overline{w}$. First, we look for the left most 1000000 subword in $\overline{w}$. Since there is no such sequence, we look for the right-most 1 and we know that this bit with all the following zeros were added in the set-up. That is, the last 10 are not part of w. We eliminate those bits and we obtain

$$\widehat{w} = 100001\ 1\ 1101\ 0010010011011100110 \in \Sigma^{30}.$$

Since $\hat{w} \in \Sigma^{30}$ we know that there was only one identical pair of length-k windows. The first bit in $\widehat{w}$ is 1. Thus, we have $i = \boldsymbol{b}(00001) = 1$. We eliminate the first 6 bits and insert 6 zeros in the first position,

$$\widehat{w} = 1\ 000000\ 1101\ 0010010011011100110.$$

Again, the first bit is 1 so the next 5 bits indicate the position of the 0. We eliminate the first 6 bits and enter 000000 in the 0th position to get the word

$$\widehat{w} = 000000\ 0\ 1101\ 0010010011011100110.$$

We are now having 0 for the first bit and the next 10 bits indicate two positions, $i = 0$, $j = 13$. We denote

$$\boldsymbol{x} = (\widehat{w})_{12+[12]} = 010010011011.$$

We now eliminate the first 11 bits and put $\boldsymbol{x}$ in the ith position and obtain

$$\widehat{w} = 010010011011\ 0010010011011100110.$$

Since $\widehat{w} \in \Sigma^{31}$ we are done. □

V. Algorithm For $k = a\log(n)$ with $1 < a < 2$

In this section, we consider the case of $k = a\log(n)$ where $1 < a < 2$. Similarly to Section IV, our coding scheme consists of two basic procedures: elimination and expansion. For the elimination phase, we compress an input sequence into an output sequence of length at most n. At every step of the compression, we remove identical windows so that at the end of this step, the output sequence does not contain any identical windows. For the elimination phase, we rely on an encoding procedure which is very similar to [20]. The process ensures that the sequence output from our encoder does not contain any all-zeros substrings of length greater than $2\log\log(n)$. Throughout this section we assume for simplicity that $\log(n)$ and $\log\log(n)$ are integers. Taking $\lfloor\log(n)\rfloor, \lfloor\log\log(n)\rfloor$ will not affect the results.

The expansion phase is the primary difference between the approach outlined here and [20]. The idea behind the expansion phase is to concatenate a zero-constrained De-Bruijn sequence, which we refer to as $v \in \Sigma^*$, with our compressed sequence, and then insert within v, all-zeros markers of length $4\log\log(n)$. These markers will be used to distinguish (or to make different) the length-k windows between v and the compressed sequence. We will explain these ideas in more detail in what follows.

For $m \in \mathbb{N}$, let $S_m(n)$ denote the set of all sequences of length n which are $(2\log\log(m))$-zero constrained, i.e.,

$$S_m(n) = \left\{u \in \Sigma^n \ : \ \mathrm{fr}_u^{2\log\log(m)}(0\ldots0) = 0\right\}.$$

Note that $S_m = \bigcup_{n\in\mathbb{N}} S_m(n)$ is the $(0, 2\log\log(m))$-RLL constrained system. It is well known (see, for example, [30]) that $\lim_{m\to\infty} \mathsf{cap}(S_m) = 1$. Moreover, the function $\log|S_m(n)|$ is subadditive in n which implies, by Fekete's lemma, that the capacity of S_m is obtained by $\inf_{n\in\mathbb{N}} \frac{1}{n}\log|S_m(n)|$. Therefore, there exists a large enough n such that $\left|S_{2\log\log(n)}(\log(n)+1)\right| \geqslant n$ (choose n such that $\mathsf{cap}(S_{2\log\log(n)})$ is close to 1). Let $f : [n] \to \Sigma^{\log(n)+1}$ be a bijection from $[n]$ to $S_{2\log\log(n)}(\log(n)+1)$, i.e., the image of f lies in the set of all $(2\log\log(n))$-zero-constrained sequences.

The elimination encoder $\mathcal{E}_{el}$, described in Algorithm 2 below, takes as input a sequence $w \in \Sigma^{n-(4\log\log(n)+3)}$, where w is $(2\log\log|w|)$-zero-constrained. The output of $\mathcal{E}_{el}$ is a sequence $\overline{w}$ of length at most $n - (4\log\log(n)+3)$ that is $(2\log\log|w|)$-zero-constrained and does not contain any repeated windows of length $k' = \log(n) + 2\log\log(n) + 5$.

Algorithm 2 Elimination Encoder, $\mathcal{E}_{el}$

1: Set $\overline{w} = w$
2: **while** there are identical length-k' windows in $\overline{w}$ **do**
3: Suppose (i, j) is a primal k'-identical window in $\overline{w}$
4: Remove the substring of length k' starting at position j and replace it with the sequence $(1, 0^{2\log\log(n)}, 1, f(i), 1)$, so that

$$\overline{w} = \overline{w}_{[j]} \circ (1, 0^{2\log\log(n)}, 1, f(i), 1) \circ \overline{w}_{\{j+k', j+k'+1, \ldots, |\overline{w}|-1\}}$$

5: **end while**
6: Return $\overline{w}$

Note that since at Step 4 we replace substrings of length k' with substrings of length $k' - 1 = \log(n) + 2\log\log(n) + 4$, so that each time Step 4 is executed, the length of $\overline{w}$ is decremented by one. We have the following result, which follows from [20].

Lemma 17. *(c.f., Claim10, [20]) The sequence $\overline{w}$ has no repeated k'-windows and $\overline{w}$ can be recovered from w.*

In the following, let $k' = \log(n) + 2\log\log(n) + 5$. For simplicity of calculations, we assume that k' is a prime number, and we later show that we can relax this assumption since the result may be generalized to non prime numbers using similar techniques. For our construction, we require the use of Lyndon words and necklaces. For a word w, we say that w is a *Lyndon* word if w is (strictly) smaller (with respect to the lexicographic order) than all of its rotations. A *necklace* of length k is an equivalence class of sequences of length k. Two sequences w, u are equivalent (or, in the same necklace) if and only if they are equivalent under rotation, i.e., there exists ℓ such that $(w_0, w_1, \ldots, w_{k-1}) = (u_\ell, u_{\ell+1}, \ldots, u_{k-1}, u_0, \ldots, u_{\ell-1})$. The next lemma follows from a well-known result on generating De-Bruijn sequences from Lyndon words [31], [32].

Lemma 18. *The lexicographic concatenation of Lyndon words of length k' which are greater (with respect to the lexicographic order) than or equal to the string*

$$\left((0^{2\log\log(n)-1} \circ 1)^{k'}\right)_{[k']}$$

generates a sequence of length greater than n which does not contain any repeated windows of length k' and also is $(4\log\log(n))$-zero-constrained.

Before proving the previous lemma, we provide an example, which illustrates the idea behind the construction.

Example 19. *Suppose $k' = 5$. Then the Lyndon words of length k' are:*

$$(0,0,0,0,0), (0,0,0,0,1), (0,0,0,1,1), (0,0,1,0,1),$$
$$(0,0,1,1,1), (0,1,0,1,1), (0,1,1,1,1), (1,1,1,1,1).$$

Concatenating these words together produces the De-Bruijn sequence

$$(0,0,0,0,0,1,0,0,0,1,1,0,0,1,0,1,0,0,1,1,1,0,1,0,1,1,0,1,1,1,1,1)$$

of length 32. The key property to notice here is that the length of the runs of zeros is smaller towards the end of the sequence than at the beginning. For example, the longest run of zeros (of length 5) appears in the first position and the last 8 bits of the sequence contains only a single zero. □

We now turn to the proof of Lemma 18.

Proof: It was established that the lexicographic concatenation of Lyndon words generates a De-Bruijn sequence [31], [32], and so it follows that our approach does not have any repeated windows of length k'. Let w be the string which results by concatenating Lyndon words greater than or equal to $\left((0^{2\log\log(n)-1} \circ 1)^{k'}\right)_{[k']}$ as stated in the lemma. We now show w is $(4\log\log(n))$-zero-constrained, which implies the statement in the lemma.

First, we recall a simple procedure from [33] which generates all Lyndon words of length k'. Let $\gamma : \{0,1\}^{k'} \to \{0,1\}^{k'}$ be such that given $x \in \Sigma^{k'}$, $\gamma(x) = y = (y_0, y_2, \ldots, y_{k'-1})$ where $y = \left((x_{[j]} \circ 1)^{\mathbb{N}}\right)_{[k']}$ where j is the largest index such that $x_{\{j,j+1,\ldots,(k'-1)\}} = (011\ldots1)$. Let

$$(0,0,\ldots,0), \gamma(0,0,\ldots,0), \gamma(\gamma(0,0,\ldots,0)), \ldots$$

be a sequence of sequences, and let V denote the result of removing non-necklaces from this sequence. It is known that V is a lexicographic (increasing) sequence of necklaces [33]. The string w (mentioned two paragraphs above) is the result of concatenating the sequences (in order) from V.

We show that if any $x \in V$ is $(2\log\log(n))$-zero-constrained, then the longest run of zeros in $\gamma(x)$ is $2\log\log(n) - 1$, which implies that w does not have any runs of length $4\log\log(n)$. Assume in the contrary that it does not hold, so that $\gamma(x)$ contains an all-zero substring of length $2\log\log(n)$. Let j be the largest index that $x_{\{j,j+1,\ldots,(k'-1)\}} = (011\ldots1)$. Then according to the procedure from the previous paragraph, the all-zero substring of length $2\log\log(n)$ occurs after index j in $\gamma(x)$, since $x_{[j]} = \gamma(x)_{[j]}$ and $\gamma(x)_j = 1$. However, this is also not possible since $\gamma(x)_{[k']}$ comprises of repeated concatenations of $x_{[j]} \circ 1$, and so we arrive at a contradiction to the assumption that x does not contain the all-zeros substring of length $2\log\log(n)$.

We have left to show that $|w| > n$. To see this, note that since k' is a prime, we can bound the length of the $|w|$ as follows.

$$
\begin{aligned}
|w| &\overset{(a)}{\geqslant} \left(\frac{2^{k'}-2}{k'} - 2^{k'-2\log\log(n)}\right)k' \\
&\overset{(b)}{=} \left(\frac{n\cdot(\log(n))^2\cdot 2^5 - 2}{\log(n)+2\log\log(n)+5} - \frac{n(\log(n))^2\cdot 2^5}{(\log(n))^2}\right)\cdot k' \\
&= n2^5\left((\log(n))^2 - \log(n) - 2\log\log(n) - 5\right) - 2 \\
&\geqslant n,
\end{aligned}
$$

where (a) follows since there are exactly $\frac{2^{k'}-2}{k'}$ necklaces of length k' and there are at most $2^{k'-2\log\log(n)}$ words which are smaller than $\left((0^{2\log\log(n)-1}\circ 1)^{k'}\right)_{[k']}$, (b) follows by plugging in the value of k' and the last inequality holds for large enough n. ■

Remark 20. *Note that the assumption that k' is prime affects only the calculation of $|w|$. For a non prime k', the calculation of $|w|$ is more involved (the expression for the number of necklaces is $\frac{1}{k'}\sum_{d|k'}\mu(d)2^{\frac{k'}{d}}$ where μ is the möbius function and the summation is over all divisors of k'). This results in the desired inequality for larger values of n.*

Let v' be the string of length at least n generated from Lemma 18. Let v be the result of inserting the all-zeros substring of length $4\log\log(n)$ periodically into v' as follows:

$$
v = \left(v'_{[k']}\ 10^{4\log\log(n)}1\ v'_{k'+[k']}\ 10^{4\log\log(n)}1\ \ldots v'_{\frac{|v'|}{k'}\cdot(k'-1)+[k']}\right). \tag{5}
$$

We have the following lemma.

Lemma 21. *Let*

$$
\hat{w} = \left(\overline{w}, 1, 0^{4\log\log(n)+1}, 1, v\right)_{[n]}
$$

be the substring of length n which results by concatenating $\overline{w}$ and v. Then $\hat{w}$ does not contain any repeated windows of length $k = \log(n) + 10\log\log(n) + 10 = k' + 8\log\log(n) + 5$.

Proof: Suppose, on the contrary, that there is a repeated k-window at (i, j). The proof is done on a case-by-case basis and we show that for all options of $j > i$, $\hat{w}_{i+[k]} \neq \hat{w}_{j+[k]}$.

If $|\overline{w}| + 4\log\log(n) + 2 - k \leqslant i \leqslant |\overline{w}| + 1$ or $|\overline{w}| + 4\log\log(n) + 2 - k \leqslant j \leqslant |\overline{w}| + 1$, then $\hat{w}_{i+[k]} \neq \hat{w}_{j+[k]}$ since the all-zeros substring of length $4\log\log(n) + 1$ appears only once in $\hat{w}$.

If $j \leqslant |\overline{w}| + 4\log\log(n) + 3 - k$, then the result follows immediately from Lemma 17.

If $i > |\overline{w}| + 1$, then we know that both $\hat{w}_{i+[k]}$ and $\hat{w}_{j+[k]}$ each contain the substring $(1, 0^{4\log\log(n)}, 1)$. Suppose for now that there is only one occurrence of $(1, 0^{4\log\log(n)}, 1)$ in $\hat{w}_{i+[k]}$. From (5), we know we can write:

$$
\hat{w}_{i+[k]} = \left(\hat{w}^{(i,1)}, 1, 0^{4\log\log(n)}, 1, \hat{w}^{(i,2)}\right).
$$

If $|\hat{w}^{(i,2)}| \geqslant 4\log\log(n) + 3$, then from (5), we can recover a substring of v' of length $k - 4\log\log(n) - 2$ by deleting the substring $(1, 0^{4\log\log(n)}, 1)$ from $\hat{w}_{i+[k]}$. Otherwise if $|\hat{w}^{(i,2)}| = t < 4\log\log(n) + 3$, then we can recover a substring of v' of length $k - 4\log\log(n) - 2 - (4\log\log(n) + 3 - t) = k' + t$ by first deleting the substring $(1, 0^{4\log\log(n)}, 1)$ from $\hat{w}_{i+[k]}$ followed by deleting the first $4\log\log(n) + 3 - t$ bits of the resulting string. The only case left to consider is where $\hat{w}_{i+[k]}$ contains two occurrences of the substring $(1, 0^{4\log\log(n)}, 1)$. Suppose the first occurrence of the substring $(1, 0^{4\log\log(n)}, 1)$ appears in position ℓ where it is clear from (5) that $\ell \in \{0, 1\}$. If $\ell = 1$, then we remove the first $4\log\log(n) + 3$ bits from $\hat{w}^{(i,2)}$ followed by the last $4\log\log(n) + 2$ bits. Otherwise, if $\ell = 0$ we remove the first $4\log\log(n) + 2$ bits from $\hat{w}^{(i,2)}$ followed by the last $4\log\log(n) + 3$ bits to obtain a substring of v' of length k' from $\hat{w}_{i+[k]}$.

From the previous paragraph, we know we can recover distinct substrings of length at least k' from v' in $\hat{w}_{i+[k]}$ and $\hat{w}_{j+[k]}$ provided $i > |\overline{w}| + 1$. Since these substrings are unique from Lemma 18, it follows that $\hat{w}_{i+[k]} \neq \hat{w}_{j+[k]}$.

We have left to consider the case where $j > |\overline{w}| + 1$ and $i < |\overline{w}| + 4\log\log(n) + 2 - k$. In this case, there are three possibilities for $\hat{w}_{i+[k]}$: a) $\hat{w}_{i+[k]}$ ends with the substring $0^{4\log\log(n)+1}$, b) $\hat{w}_{i+[k]}$ ends with the substring $0^{4\log\log(n)}$, or c) $\hat{w}_{i+[k]}$ does not contain the substring $0^{4\log\log(n)}$. If a) holds, then clearly $\hat{w}_{i+[k]} \neq w_{j+[k]}$, since by assumption $j > |\overline{w}| + 1$

and $0^{4\log\log(n)+1}$ only appears once in $\hat{w}$. If b) holds, then from (5), $\hat{w}_{j+[k]}$ contains two occurrences $0^{4\log\log(n)}$, and $\hat{w}_{i+[k]}$ only has one occurrence so that $\hat{w}_{i+[k]} \neq \hat{w}_{j+[k]}$. Finally, if c) holds, then $\hat{w}_{i+[k]}$ does not contain the substring $0^{4\log\log(n)}$ but $\hat{w}_{j+[k]}$ does and so $\hat{w}_{i+[k]} \neq \hat{w}_{j+[k]}$ in this case as well. ■

We now present our main result, which follows from the previous discussion.

Theorem 22. *There exists a rate-1 polynomial-time encoder which generates sequences with no-identical k-windows for any $k > a\log(n)$ where $a > 1$.*

Proof: The fact that our algorithm has polynomial-time encode complexity follows from the observation that $\mathcal{E}_{el}$ runs in polynomial time along with the fact that generating a lexicographic ordering of Lyndon words can be accomplished in time at most $\mathcal{O}(2^{k'})$ which is polynomial in n. Suppose $\hat{w} = (\overline{w}, 1, 0^{4\log\log(n)+1}, 1, v)_n$ is a codeword from Lemma 21. Then to recover w from $\hat{w}$, we simply remove the suffix $(1, 0^{4\log\log(n)+1}, 1, v)$ from $\hat{w}$, which is the first suffix of $\hat{w}$ that begins with the substring $(1, 0^{4\log\log(n)+1}, 1)$, to recover $\overline{w}$. The result then follows immediately from Lemma 17 since w can be recovered from $\overline{w}$.

Next, we verify the statement on the rate. From Claim 7 in [20], we have that there are at least

$$\left(\frac{n}{4}\cdot\left(1-\frac{\log(n)}{(\log(n))^2}\right)\right)^{\lfloor\frac{n-(4\log\log(n)+3)}{\log(n)}\rfloor},$$

possible input sequences for Algorithm 2 since we can divide up the input sequence of length $n-(4\log\log(n)+3)$ into blocks of length $\log(n)$ that begin and end with the symbol 1, and then constrain each block to have runs of zeros of length at most $2\log\log(n)-1$. Then,

$$\lim_{n\to\infty}\frac{1}{n}\log\left(\frac{n}{4}\cdot\left(1-\frac{\log(n)}{(\log(n))^2}\right)\right)^{\lfloor\frac{n-(4\log\log(n)+3)}{\log(n)}\rfloor} = 1,$$

which completes the proof. ■

VI. k-Repeat Free Sequences With Combinatorial Constraints

In this section we study the combination of k-repeat free sequences and combinatorial constraints. As mentioned previously, the number of De-Bruijn sequences of span k can be calculated using several combinatorial methods such as the doubling process, the BEST theorem, and using shift registers. Unfortunately, calculating the exact number of De-Bruijn sequences which also satisfy other constraints is not an easy problem [34]. Here, we calculate the capacity of k-repeat free systems with local constraints. For convenience, throughout this section we restrict Σ to the binary alphabet but the same method can be used for larger alphabets.

Before stating the main result of this section, we remind the reader some known definitions and basic results on constrained systems. We follow the lines of [30]. Let $G = (V, E, L)$ be a labeled (directed) graph where V is the set of vertices, E is the set of edges and $L : E \to \Sigma$ is a labeling of the edges. We say that a graph G is *deterministic* if from every vertex, the outgoing edges have different labels. For each graph G, we denote by A_G the adjacency matrix of G. The adjacency matrix is a $|V| \times |V|$ matrix such that the u, v entry of A_G is the number of edges which start at the vertex u and end at v.

A constrained system $S \subseteq \Sigma^{\mathbb{N}}$ is the set of all words obtained by reading the labels of paths in a labeled directed graph. If S is obtained by a graph G, we say that G presents the system S (or G is a presentation of S). A constrained system which is presented by a graph G is said to be *irreducible* if G is strongly connected. For a system S, we denote by $\mathcal{B}_n(S)$ the set of all length-n blocks that appear in words in S. The language of S is denoted by $\mathcal{B}(S) = \bigcup_{n\in\mathbb{N}} \mathcal{B}_n(S)$. It is well known that every constrained system can be presented by a deterministic graph [30, Prop. 2.2]. Therefore, we will assume that all presentations are deterministic. The capacity of a constrained system S is defined as $\mathsf{cap}(S) = \limsup_{n\to\infty} \frac{1}{n}\log_2|\mathcal{B}_n(S)|$. The adjacency matrix is highly related to the capacity of the system. If S is irreducible, the Perron-Frobenius theorem states that A_G has a largest, real, simple eigenvalue λ, with strictly positive left and right eigenvectors. If S is irreducible, it is well known that $\mathsf{cap}(S) = \log_2\lambda$ [30, Th. 3.4].

In this section we are interested in constrained systems which are also repeat free. In other words, if S is a constrained system, we are interested in the following set of words.

Definition 23. *Let S be an irreducible deterministic constrained system with language $\mathcal{B}(S)$ and let $\mathcal{W}_k(n)$ denote the k-repeat free sequences. The (S,k)-repeat free system with is defined by the following sets,*

$$\mathcal{X}_{S,k}(n) = \{w \in \Sigma^n \ : \ w \in \mathcal{W}_k(n) \cap \mathcal{B}_n(S)\}.$$

We define the system $\mathcal{X}_{S,k} = \bigcup_{n\in\mathbb{N}} \mathcal{X}_{S,k}(n)$.

We are interested in the capacity of the system $\mathsf{cap}(\mathcal{X}_{S,k})$, where $k = k(n)$ grows with n. In order to estimate this capacity we need the following useful characterization of the capacity of a constrained system given by Markov chains. For a graph

$G=(V,E)$, a Markov chain is given by a transition probability matrix $P\in[0,1]^{|V|\times|V|}$ such that $P\cdot\mathbf{1}=\mathbf{1}$, where $\mathbf{1}$ is the all ones vector. For an edge $e\in E$, we denote by e^b the starting vertex of e and by e^t the terminal vertex, i.e., if $e=(u,v)\in E$ then $e^b=u$ and $e^t=v$. Thus, from a vertex u, the (u,v) entry of P corresponds to the transition probability from vertex u to vertex v. If for every $u,v\in V$ there exists $n\in\mathbb{N}$ such that $(P^n)_{u,v}>0$ then we say that P is *irreducible*. For an irreducible Markov chain P, there is a unique positive stationary vector μ^T such that $\mu^TP=\mu^T$. For a Markov chain P on a graph $G=(V,E)$ with stationary distribution μ, the entropy of the Markov chain is defined as

$$H(P)=-\sum_{u\in V}\mu_u\sum_{(u,v)\in E}P_{u,v}\log_2 P_{u,v}.$$

We may now state the known relation between the capacity and Markov chains [30, Th. 3.23].

Theorem 24. *[30, Th. 3.23] Let S be an irreducible constrained system presented by G with Perron eigenvalue λ. Then*

$$\sup_P H(P)=\log_2\lambda=\mathsf{cap}(S),$$

where the supremum is taken over all Markov chains on G.

In order to use Theorem 24, we need to find the entries of P that maximize the entropy. Note that although in Theorem 24 we take supremum over all Markov chains, the set on which we take the supremum is a compact set, which means that the supremum is in fact a maximum. Moreover, a closer look on the proof of Theorem 24 (as in [30, Th. 3.23]), reveals exactly the maximizing transition probabilities and the corresponding stationary vector. Indeed, let A_G be the adjacency matrix of G and denote by η^T,ν the normalized left and right eigenvectors of the Perron eigenvalue λ such that $\eta^T\nu=\mathbf{1}$. Then, the Markov chain which maximizes the entropy is given by

$$P_{u,v}=\frac{(A_G)_{u,v}\nu_v}{\lambda\nu_u},$$

and the corresponding stationary vector is given by $\mu_u=\eta_u^T\nu_u$. This means that all the edges from u to v are prescribed with the same probability which is $\frac{\nu_u/\lambda\nu_v}{(A_G)_{u,v}}$. In the next lemma we show that in a constrained system, the probability of two k-tuples to be identical is upper bounded by a constant times λ^{-k}.

Lemma 25. *Let S be an irreducible constrained system presented by $G=(V,E)$ with an entropy maximizing Markov chain P. Let $x\in\Sigma^{\mathbb{N}}$ be a sequence obtained by reading the labels of a path evolving according to P with the initial state chosen according to the stationary distribution $\mu=(\eta_v^T\nu_v)_{v\in V}$. Then for every $i\in\mathbb{N}$ and $k\in\mathbb{N}$,*

$$\Pr\left(x_{[k]}=x_{i+[k]}\right)\leqslant\frac{|V|d^2}{\lambda^k},$$

where $d=\max_{v,u\in V}\frac{\nu_v}{\nu_u}$ and λ is the Perron eigenvalue of the adjacency matrix A_G.

Proof: Recall that a path γ is a sequence of edges $\gamma=(e_0,\dots,e_{k-1})$ such that for every $i\in[k-1]$, $e_i^t=e_{i+1}^b$. First we note that the probability of a specific path over the graph depends only on the start vertex, end vertex, and the length of the path. Indeed,

$$\begin{aligned}\Pr((e_0,\dots,e_{k-1}))&=\mu_{e_0^b}\frac{P_{e_0^b,e_0^t}}{(A_G)_{e_0^b,e_0^t}}\frac{P_{e_1^b,e_1^t}}{(A_G)_{e_1^b,e_1^t}}\cdots\frac{P_{e_{k-1}^b,e_{k-1}^t}}{(A_G)_{e_{k-1}^b,e_{k-1}^t}}\\&=\mu_{e_0^b}\frac{\nu_{e_0^b}}{\lambda\nu_{e_0^t}}\frac{\nu_{e_1^b}}{\lambda\nu_{e_1^t}}\cdots\frac{\nu_{e_{k-1}^b}}{\lambda\nu_{e_{k-1}^t}}\overset{(a)}{=}\mu_{e_0^b}\frac{\nu_{e_0^b}}{\nu_{e_{k-1}^t}}\frac{1}{\lambda^k},\end{aligned}\tag{6}$$

where (a) follows since $e_i^t=e_{i+1}^b$. Since the system is irreducible, μ,ν,η are all positive. If we denote by d the value $d=\max_{v,u\in V}\frac{\nu_v}{\nu_u}\geqslant 1$ we obtain

$$\Pr((e_0,\dots,e_{k-1}))\leqslant\mu_{e_0^b}\frac{d}{\lambda^k}.$$

For a sequence of edges $\gamma=(e_0,\dots,e_{k-1})$ we denote $L(\gamma)\triangleq L(e_0)L(e_1)\cdots L(e_{k-1})$ and denote by γ_0 the vertex e_0^b. We denote by Γ the set of all paths and for $i\in\mathbb{N}$ we denote by Γ^i the set of all paths of length i. Note that for a specific $w\in\Sigma^k$,

$$\Pr(\pi_k(x)=w)=\sum_{\gamma\in\Gamma}\mathbb{1}_w(L(\gamma))\Pr(\gamma)\leqslant\sum_{\gamma\in\Gamma}\mathbb{1}_w(L(\gamma))\mu_{\gamma_0}\frac{d}{\lambda^k}.$$

Since the graph is deterministic, if $\gamma=(e_0,\dots,e_{k-1})$ is a path with $L(\gamma)=w$ then it is the only path with this labeling which starts at the vertex e_0^b. Thus,

$$\Pr(x_{[k]}=w)\leqslant\sum_{v\in V}\mu_v\frac{d}{\lambda^k}\leqslant\frac{d}{\lambda^k}.$$

Since μ is the stationary probability vector, it is shift invariant, i.e., for $w \in \Sigma^k$ and $i \in \mathbb{N}$

$$\Pr(x_{i+[k]} = w) = \Pr(x_{[k]} = w).$$

Assume $i \in \mathbb{N}$ and write

$$\begin{aligned}\Pr\left(x_{[k]} = x_{i+[k]}\right) &= \sum_{w\in\Sigma^{i+k}} \mathbb{1}_{x_{[k]}}(x_{i+[k]}) \Pr(x_{[k+i]} = w) \\ &\leqslant \frac{d}{\lambda^{i+k}} \sum_{w\in\Sigma^{i+k}} \mathbb{1}_{x_{[k]}}(x_{i+[k]}) \leqslant \frac{d}{\lambda^{i+k}} \cdot |\mathcal{B}_i(S)|.\end{aligned} \tag{7}$$

We now need to estimate the value $|\mathcal{B}_i(S)|$. Note that $|\mathcal{B}_i(S)| \leqslant |\Gamma^i|$. Since η^T, ν are left and right eigenvectors of A_G, respectively, for $i \in \mathbb{N}$ we may write

$$\sum_{u\in V}\sum_{v\in V} (A_G^i)_{u,v}\nu_v = \mathbf{1} \cdot A_G^i \cdot \nu = \lambda^i \|\nu\|_1.$$

Since $\|\nu\|_1 \leqslant |V| \max_{v\in V}\{\nu_v\}$ and since $\forall v \in V,\ \nu_v \geqslant \min_{v\in V}\{\nu_v\}$ we obtain

$$\mathbf{1}^T \cdot A_G^i \cdot \mathbf{1} = \sum_{u\in V}\sum_{v\in V} (A_G^i)_{u,v} \leqslant |V| d\lambda^i.$$

Since $|\Gamma^i| = \mathbf{1}A_G^i\mathbf{1}$, plugging it in (7) concludes the proof. ∎

We now state and prove the main result of this section.

Theorem 26. *Let S be an irreducible constrained system presented by the graph G with Perron eigenvalue λ. For every $n \in \mathbb{N}$ let $k = \lfloor a \log_\lambda(n) \rfloor$ with $a > 1$. Then*

$$\mathsf{cap}(\mathcal{X}_{S,k}) = \mathsf{cap}(S).$$

Proof: We again use probability to calculate the capacity. First, let $\mathbb{P}(\cdot)$ denote the uniform probability over the length-n sequences and note that $\frac{|\mathcal{X}_{S,k}(n)|}{2^n} = \mathbb{P}(\mathcal{X}_{S,k}(n))$. Thus, $\mathsf{cap}(\mathcal{X}_{S,k}) = 1 + \limsup_{n\to\infty} \frac{1}{n} \log_2 \mathbb{P}(\mathcal{X}_{S,k}(n))$.

First note that $\mathcal{X}_{S,k}(n) \subseteq \mathcal{B}_n(S)$ which means that $\mathsf{cap}(\mathcal{X}_{S,k}(n)) \leqslant \mathsf{cap}(S)$. So we only need to show that $\mathsf{cap}(\mathcal{X}_{S,k}(n)) \geqslant \mathsf{cap}(S)$. Assume that S is presented by a graph $G = (V, E, L)$ with Perron eigenvalue λ and an entropy maximizing transition probability P with left Perron eigenvector η^T and right Perron eigenvector ν normalized such that $\eta^T \cdot \nu = 1$. Every sequence obtained according to P belongs to $\mathcal{B}(S)$. Denote by μ the stationary distribution of P. Let $\gamma = (e_0, e_1, \dots)$ be a path on G evolving according to P with initial vertex chosen according to μ. Denote by w the sequence obtained by reading the labels of the path γ, i.e., $w \triangleq L(\gamma)$. For $\ell > 0$, define the random variable

$$\mathrm{X}_\ell = \sum_{j=(\ell-1)k+1}^{\ell k} \sum_{i=0}^{j-1} \mathbb{1}_{w_{j+[k]}}(w_{i+[k]}).$$

Taking expectation we obtain

$$\begin{aligned}\mathbb{E}\left[\mathrm{X}_\ell\right] &= \mathbb{E}\left[\sum_{j=(\ell-1)k+1}^{\ell k} \sum_{i=0}^{j-1} \mathbb{1}_{w_{j+[k]}}(w_{i+[k]})\right] \\ &\overset{(a)}{\leqslant} \sum_{j=(\ell-1)k+1}^{\ell k} \sum_{i=0}^{j-1} \frac{|V|d^2}{\lambda^k} \leqslant \frac{|V|d^2(\ell+1)k^2}{\lambda^k},\end{aligned}$$

where (a) follows since μ is stationary, P is shift invariant and by Lemma 25. Thus, by the Markov inequality we obtain $\Pr(X_\ell \geqslant 1) \leqslant \frac{|V|d^2(\ell+1)k^2}{\lambda^k}$. Plugging in this result yields

$$\Pr\left(\mathcal{W}_k(n)\right) \geqslant \prod_{\ell=0}^{n/k} \left(1 - \frac{|V|d^2\ell k^2}{\lambda^k}\right).$$

For n large enough, $\left(1 - \frac{|V|d^2\ell k^2}{\lambda^k}\right)$ is positive for every ℓ. This holds since $|V|d^2$ is a constant, $k = O(\log(n))$, ℓ is upper bounded by $\left(\frac{n}{k} + 1\right)$ and $\lambda = n^a$ where $a > 1$. Taking logarithm and exponent we obtain

$$\Pr\left(\mathcal{W}_k(n)\right) \geqslant \lambda^{\sum_{\ell=0}^{n/k} \log_\lambda\left(1 - \frac{|V|d^2\ell k^2}{\lambda^k}\right)}.$$

For every $x \in (0,1)$, $\log_\lambda(1-x) \geqslant -\frac{x}{(1-x)\ln\lambda}$ and thus

$$\Pr\left(\mathcal{W}_k(n)\right) \geqslant \lambda^{\sum_{\ell=0}^{n/k} \frac{|V|d^2\ell k^2}{(\lambda^k - |V|d^2\ell k^2)\ln\lambda}} \geqslant \lambda^{-(\frac{n}{k}+1)\frac{|V|d^2nk}{(\lambda^k-|V|d^2nk)\ln\lambda}},$$

where the last inequality follows by taking the largest argument in the sum ${}^{n}/{}_{k}+1$ times. Substituting $k = \lfloor a\log_\lambda(n)\rfloor$ and writing $a = 1+\epsilon$ we obtain

$$\begin{aligned}\frac{1}{n}\log_2\Pr\left(\mathcal{W}_k(n)\right) \geqslant &-\frac{|V|d^2\log_2\lambda}{(n^\epsilon - |V|d^2a\log_\lambda n)\ln\lambda} \\ &- \frac{|V|d^2a\log_\lambda n\log_2\lambda}{n(n^\epsilon-|V|d^2a\log_\lambda n)\ln\lambda}.\end{aligned}$$

Taking $n\to\infty$ we obtain

$$\limsup_{n\to\infty}\frac{1}{n}\log_2\Pr(\mathcal{W}_k(n)) \geqslant 0. \tag{8}$$

Next, let $\mathbb{P}$ denote the uniform distribution over Σ^n. Note that in order to use the probability argument in order to estimate $|\mathcal{W}_k(n)|$, we need to use the uniform distribution (indeed, $|\mathcal{X}_{S,k}(n)| = |\Sigma|^n\cdot\mathbb{P}(\mathcal{X}_{S,k}(n))$ since $\mathbb{P}$ is the uniform distribution). We have

$$\frac{1}{n}\log_2\left|\mathcal{X}_{S,k}(n)\right| = 1 + \frac{1}{n}\log_2\mathbb{P}\left(\mathcal{X}_{S,k}(n)\right).$$

Now note that the probability denoted by $\Pr(\cdot)$ in (8) is not the uniform distribution but a distribution obtained by the Markov chain P with stationary initial distribution μ. To finish the proof we need to show that for n large enough, $\Pr(\cdot)$ is almost uniform on the set $\mathcal{B}_n(S)$. By the definition of $\mathcal{X}_{S,k}(n)$ we have that $\mathbb{P}\left(\mathcal{X}_{S,k}(n)\right) = \mathbb{P}\left(\mathcal{B}_n(S)\right)\mathbb{P}\left(\mathcal{W}_k(n)\mid\mathcal{B}_n(S)\right)$. Hence, we obtain

$$\frac{1}{n}\log_2\left|\mathcal{X}_{S,k}(n)\right| = 1+\frac{1}{n}\log_2\mathbb{P}\left(\mathcal{B}_n(S)\right) + \frac{1}{n}\log_2\mathbb{P}\left(\mathcal{W}_k(n)\mid\mathcal{B}_n(S)\right).$$

Note that

$$\mathsf{cap}(S) = \log_2\lambda = 1+\limsup_{n\to\infty}\frac{1}{n}\log_2\mathbb{P}\left(\mathcal{B}_n(S)\right).$$

Therefore, we have

$$\begin{aligned}\limsup_{n\to\infty}\frac{1}{n}\log_2\left|\mathcal{X}_{S,k}(n)\right| = \log_2\lambda& \\ +\limsup_{n\to\infty}\frac{1}{n}\log_2\mathbb{P}\left(\mathcal{W}_k(n)\mid\mathcal{B}_n(S)\right)&.\end{aligned} \tag{9}$$

We claim now that

$$\limsup_{n\to\infty}\frac{1}{n}\log_2\mathbb{P}\left(\mathcal{W}_k(n)\mid\mathcal{B}_n(S)\right) = \limsup_{n\to\infty}\frac{1}{n}\log_2\Pr\left(\mathcal{W}_k(n)\right). \tag{10}$$

Showing this will finish the proof since plugging (10) to (9), together with (8) yields

$$\mathsf{cap}(\mathcal{X}_{S,k}) \geqslant \log_2\lambda = \mathsf{cap}(\mathcal{B}(S)).$$

Note that (10) follows directly from Lemma 25. Indeed,

$$\Pr\left(\mathcal{W}_k(n)\right) = \sum_{w\in\mathcal{W}_k(n)}\Pr(\{w\}) \leqslant |\mathcal{W}_k(n)\cap\mathcal{B}_n(S)|\,\frac{|V|d^2}{\lambda^n}.$$

On the other hand, denoting $d' = \min_{u,v\in V}\frac{\nu_v}{\nu_u}$ we obtain from (6) that

$$\Pr(\mathcal{W}_k(n)) \geqslant |\mathcal{W}_k(n)\cap\mathcal{B}_n(S)|\min_{v\in V}\mu_v\frac{d'}{\lambda^k}.$$

Thus,

$$\limsup_{n\to\infty}\frac{1}{n}\log_2\Pr\left(\mathcal{W}_k(n)\right) = \limsup_{n\to\infty}\frac{1}{n}\log_2|\mathcal{W}_k(n)\cap\mathcal{B}_n(S)| - \log_2\lambda. \tag{11}$$

Since $\mathbb{P}$ is the uniform probability, we have that $\mathbb{P}\left(\mathcal{W}_k(n)\mid\mathcal{B}_n(S)\right) = \frac{|\mathcal{W}_k(n)\cap\mathcal{B}_n(S)|}{|\mathcal{B}_n(S)|}$ which means that

$$\limsup_{n\to\infty}\frac{1}{n}\log_2\mathbb{P}\left(\mathcal{W}_k(n)\mid\mathcal{B}_n(S)\right) = \limsup_{n\to\infty}\frac{1}{n}\log_2|\mathcal{W}_k(n)\cap\mathcal{B}_n(S)| - \log_2\lambda. \tag{12}$$

Combining (11) with (12) we obtain the wanted equality which concludes the proof. ■

Example 27. *In this example we consider the (inverted)* $(0,1)$*-RLL constrained* k*-repeat free sequences (the constrained system is denoted by* S*). Hence, we are interested in sequences for which every* k*-tuple appears at most once and also there are no consecutive ones. We define accordingly the set*

$$\mathcal{X}_{S,k}(n) = \left\{ w \in \Sigma^n \;:\; \mathrm{fr}_w^2(11) = 0 \text{ and } w \in \mathcal{W}_k(n) \right\}.$$

We start by considering the adjacency matrix of the $(0,1)$*-RLL system which is given by*

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

The Perron eigenvalue is $\lambda = \frac{1+\sqrt{5}}{2}$ *and the corresponding eigenvectors are* $\eta = \frac{1}{\sqrt{\lambda+2}} \begin{bmatrix} \lambda & 1 \end{bmatrix}, \nu = \frac{1}{\sqrt{\lambda+2}} \begin{bmatrix} \lambda \\ 1 \end{bmatrix}$ *(note that* $\lambda^2 = \lambda + 1$*). The transition probabilities that maximize the entropy are given by*

$$P = \begin{bmatrix} \frac{1}{\lambda} & \frac{1}{\lambda^2} \\ 1 & 0 \end{bmatrix},$$

with stationary distribution $\mu = \begin{bmatrix} \frac{\lambda+1}{\lambda+2} & \frac{1}{\lambda+2} \end{bmatrix}$*. By Lemma 25 we obtain that for every word* $w \in S$ *of length* n*,* $\Pr(w) \leqslant \frac{2\lambda^2}{\lambda^n}$*. By Theorem 24 the capacity* $\mathsf{cap}(\mathcal{X}_{S,k}(n)) = \log_2(\lambda)$ *when* $k = \lfloor a \log_\lambda(n) \rfloor$ *with* $a > 1$*.* □

VII. Multidimensional k-Repeat Free Patterns

In this section we generalize the capacity results of Section II to the multidimensional case. First, we generalize the relevant notations. Let $\mathbb{N}^d$ be the d-dimensional grid. For a vector $v = (v_0, \ldots, v_{d-1}) \in \mathbb{N}^d$, we define $[v] = [v_0] \times [v_1] \times \cdots \times [v_{d-1}]$. We also use $\mathbf{e}_i$ the unit vector of direction i. For $n \in \mathbb{N}$, we denote by $[n]^d$ the d-dimensional cube of length n, i.e., $[n]^d = \otimes_{i=0}^{d-1}[n]$. Let $w \in \Sigma^{[n]^d}$ and let $v \in [n]^d$, we denote by w_v the symbol located in the v location. We also denote by $\Sigma^{*d} = \bigcup_{n\in\mathbb{N}} \Sigma^{[n]^d}$ the set of all d-dimensional finite cubes.

We now define the d-dimensional capacity and the empirical frequency.

Definition 28. *Let* $\mathcal{L} \subseteq \Sigma^{*d}$ *be a system. The capacity of* $\mathcal{L}$ *is denoted by* $\mathsf{cap}(\mathcal{L})$ *and is defined as*

$$\mathsf{cap}(\mathcal{L}) \triangleq \limsup_{n\to\infty} \frac{1}{n^d} \log_{|\Sigma|} |\mathcal{L} \cap \Sigma^{[n]^d}|.$$

For a pattern $w \in \Sigma^{[n]^d}$ and for a set of coordinates $A \subseteq [n]^d$, we denote by w_A the restriction of w to the set A. We also denote by $|w|$ the side-length n of w.

Definition 29. *Let* $w \in \Sigma^{[n]^d}$ *and* $k \leqslant n$*. The empirical frequency of* k*-patterns in* w *is denoted by* fr_w^k *and is defined as follows. For a* k*-pattern* $u \in \Sigma^{[k]^d}$*,*

$$\mathrm{fr}_w^k(u) \triangleq \frac{1}{(n-k+1)^d} \sum_{v\in[n-k+1]^d} \mathbb{1}_u \left(w_{v+[k]^d} \right).$$

For the measure fr_w^k, the support of fr_w^k, $\mathrm{Supp}(\mathrm{fr}_w^k)$, is the set of all k-patterns that appear in w.

Example 30. *Let* Σ *be the binary alphabet and let*

$$w_1 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}, w_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Let $k = 2$ *and let*

$$u = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Note that $\mathrm{fr}_{w_1}^2$ *is the empirical frequency of* 2×2 *matrices in* w_1*. We have that*

$$\mathrm{Supp}\,\mathrm{fr}_{w_1}^k = \left\{ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \right.$$
$$\left. \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \right\}.$$

Also, $\mathrm{fr}^k_{w_1}(u) = 0$ *and* $\mathrm{fr}^k_{w_2}(u) = \frac{2}{9}$. □

A d-dimensional De-Bruijn system over the alphabet Σ with $|\Sigma| = q$ is denoted by $\mathcal{B}^d_q$ and is defined as the set of all De-Bruijn patterns (over Σ) of span $[k]^d$ for all $k \in \mathbb{N}$. In a notational form, a d-dimensional De-Bruijn system is the set

$$\mathcal{B}^d_q = \left\{ w \in \Sigma^{*d} : \exists k \in \mathbb{N} \; s.t. \; \forall u \in \Sigma^{[k]^d}, \mathrm{fr}^k_w(u) = \frac{1}{(|w| - k + 1)^n} \right\}.$$

In a similar fashion, we define the d-dimensional k-repeat free patterns.

Definition 31. *A pattern $w \in \Sigma^{[n]^d}$ is said to be length-n k-repeat free if every $[k]^d$-tuple appears at most once. The set of length n-length k-repeat free patterns is denoted by*

$$\mathcal{W}^d_k(n) \triangleq \left\{ w \in \Sigma^{[n]^d} \;:\; \forall u \in \Sigma^{[k]^d}, \; \mathrm{fr}^k_w(u) \leqslant \frac{1}{(n-k+1)^d} \right\}.$$

Note that $\mathcal{W}^1_k(|\Sigma|^k + k - 1)$ is exactly the set of all De-Bruijn sequences of span k. Moreover, if $k < \left(d \log_{|\Sigma|}(n-k+1) \right)^{1/d}$ then it holds that $\mathcal{W}^d_k(n) = \emptyset$. Therefore, we are interested in studying the size of the set $\mathcal{W}^d_k(n)$ where $k > \left(d \log_{|\Sigma|}(n-k+1) \right)^{1/d}$. Consider the uniform distribution over all d-dimensional patterns of length n, then

$$|\mathcal{W}^d_k(n)| = |\Sigma|^{n^d} \cdot \Pr\left(\mathcal{W}^d_k(n) \right).$$

For $a > 1$, we define the d-dimensional k-repeat free system as $\mathcal{W}^d_a = \bigcup_{n \in \mathbb{N}} \mathcal{W}^d_k(n)$, where $k = \left\lfloor a \left((2d-1) \log_{|\Sigma|} n \right)^{1/d} \right\rfloor$. The capacity, in this case, is given by

$$\mathsf{cap}(\mathcal{W}^d_a) = 1 + \limsup_{n \to \infty} \frac{1}{n} \log_{|\Sigma|} \Pr(\mathcal{W}^d_k(n)). \tag{13}$$

Our main result in this section is stated in the following theorem, which is a generalization of Theorem 8 for the d-dimensional case.

Theorem 32. *Let Σ be a finite alphabet of size q then for all $a > 1$, $\mathsf{cap}(\mathcal{W}^d_a) = 1$.*

Proof: Let $w \in \Sigma^{\mathbb{N}^d}$ be a random word in which each coordinate is chosen uniformly and independently over Σ. Let $n \in \mathbb{N}$ and $k = a \log_q n$ (we assume for simplicity that $a \log_q n$ is an integer). For a number $1 < \ell \in \mathbb{N}$ we denote by $F_{k\ell}$ the d-dimensional cube of length $k\ell$,

$$F_{k\ell} \triangleq \left\{ v \in \mathbb{N}^d \;:\; \|v\|_\infty \leqslant k\ell - 1 \right\}.$$

We denote by $\partial F_{k\ell}$ the following set of coordinates

$$\partial F_{k\ell} \triangleq F_{k\ell+1} \backslash F_{k(\ell-1)+1}.$$

We will use the set $\partial F_{k(\ell-1)}$. One may think of $\partial F_{k(\ell-1)}$ as k-thick boundary which is defined as a union of boundaries $\partial F_{k(\ell-1)} = \bigcup_{j=1}^{\ell} \left(F_{k(\ell-2)+1+j} \backslash F_{k(\ell-2)+j} \right)$. Assume also that the set $\partial F_{k(\ell-1)}$ is ordered according to the lexicographic order and that every summation is in accordance with that order. For a coordinate $v \in \partial F_{k(\ell-1)}$, we denote by $[v]_{\partial F_{k(\ell-1)}}$ the set $[v]_{\partial F_{k(\ell-1)}} \triangleq \left\{ u \in \partial F_{k(\ell-1)} \;:\; u < v \right\}$ where $u < v$ is with respect to the lexicographic order on $\partial F_{k(\ell-1)}$. Define the random variable

$$X_\ell = \sum_{v \in \partial F_{k(\ell-1)}} \; \sum_{u \in [(\ell-2)k+1]^d \cup [v]_{\partial F_{k(\ell-1)}}} \mathbb{1}_{w_{v+[k]^d}}(w_{u+[k]^d}).$$

Note that since $\partial F_{k(\ell-1)}$ is a k-thick boundary,

$$|\partial F_{k(\ell-1)}| = ((\ell-1)k)^d - ((\ell-2)k)^d \leqslant dk^d \, (\ell-1)^{d-1}.$$

Moreover, we have that

$$\left| [(\ell-2)k+1]^d \cup [v]_{\partial F_{k(\ell-1)}} \right| \leqslant (\ell k)^d,$$

and for every v, u,

$$\mathbb{E}\left[\mathbb{1}_{w_{v+[k]^d}}(w_{u+[k]^d}) \right] = \frac{1}{q^{k^d}}.$$

Thus
$$\mathbb{E}[X_\ell] \leqslant \frac{dk^d \ell^{d-1}(\ell k)^d}{q^{k^d}} = \frac{dk^{2d}\ell^{2d-1}}{q^{k^d}}.$$
Applying Markov inequality we obtain $\Pr(X_\ell \geqslant 1) \leqslant \frac{dk^{2d}\ell^{2d-1}}{q^{k^d}}$ which implies that
$$\Pr(X_\ell < 1) \geqslant 1 - \frac{dk^{2d}\ell^{2d-1}}{q^{k^d}}.$$
Following similar steps as in the proof of Theorem 8 we obtain
$$\Pr(w_{[n]} \in \mathcal{W}_a) \geqslant \prod_{\ell=1}^{\lfloor n/k \rfloor} \left(1 - \frac{dk^{2d}\ell^{2d-1}}{q^{k^d}}\right).$$
This, in turn, implies that
$$\Pr(w_{[n]} \in \mathcal{W}_a) \geqslant q^{-\left(\frac{n}{k}+1\right)\frac{dkn^{2d-1}}{(q^{k^d}-dkn^{2d-1})\ln q}}.$$
Taking $k = a\sqrt[d]{(2d-1)\log_q n}$ with $a = 1+\epsilon$, $\epsilon > 0$ implies that $\mathsf{cap}\left(\mathcal{W}_a^d\right) \to 1$ as $n \to \infty$. Note that if d is large enough so that $(1+\epsilon)^d \geqslant 2d-1$, it is enough to take $k = a\log_q n$ with $a > 1$ and obtain $\mathsf{cap}\left(\mathcal{W}_a^d\right) \to 1$ as $n \to \infty$. ■

VIII. Conclusion

In this paper we consider k-repeat free sequences over a general alphabet, which generalize the well-known De-Buijn sequences. We calculate the capacity of the sequences for k which is a function of the sequence's length. We also study the capacity of k-repeat free sequences with local constraints, imposed by a given irreducible constrained system, and the capacity of d-dimensional k-repeat free patterns. For the binary case, we also provide an efficient encoding and decoding scheme that achieves the capacity.

As a future work, it will be interesting to find an efficient encoding and decoding scheme for k-repeat free sequences with local constraints. We believe that it is possible to modify our coding technique and to adjust it to this case. It is also interesting to find an efficient coding algorithm for the d-dimensional k-repeat free patterns.

References

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

[2] F. Farnoud, M. Schwartz, and J. Bruck, "A stochastic model for genomic interspersed duplication," in *IEEE Int. Symp. Inf. Theory (ISIT)*. IEEE, 2015, pp. 904–908.

[3] ——, "The capacity of string-duplication systems," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 811–824, 2016.

[4] O. Elishco, F. Farnoud, M. Schwartz, and J. Bruck, "The capacity of some pólya string models," in *IEEE Int. Symp. Inf. Theory (ISIT)*. IEEE, 2016, pp. 270–274.

[5] ——, "The capacity of some pólya string models," *CoRR*, vol. abs/1808.06062, 2018. [Online]. Available: http://arxiv.org/abs/1808.06062

[6] H. Lou, F. Farnoud, M. Schwartz, and J. Bruck, "Evolution of k-mer frequencies and entropy in duplication and substitution mutation systems," *arXiv preprint arXiv:1812.02250*, 2018.

[7] Y. M. Chee, J. Chrisnata, H. M. Kiah, and T. T. Nguyen, "Deciding the confusability of words under tandem repeats," *arXiv preprint arXiv:1707.03956*, 2017.

[8] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck, "Duplication-correcting codes for data storage in the dna of living organisms," *IEEE Trans. Inf. Theory*, vol. 63, no. 8, pp. 4996–5010, 2017.

[9] L. Dolecek and V. Anantharam, "Repetition error correcting sets: Explicit constructions and prefixing methods," *SIAM Journal on Discrete Mathematics*, vol. 23, no. 4, pp. 2120–2146, 2010.

[10] A. Lenz, A. Wachter-Zeh, and E. Yaakobi, "Bounds on codes correcting tandem and palindromic duplications," *arXiv preprint arXiv:1707.00052*, 2017.

[11] A. Wachter-Zeh, "List decoding of insertions and deletions," *IEEE Trans. Inf. Theory*, vol. 64, no. 9, pp. 6297–6304, 2018.

[12] D. Margaritis and S. S. Skiena, "Reconstructing strings from substrings in rounds," in *36th Annual Symp. Foundations of Computer Science*. IEEE, 1995, pp. 613–620.

[13] 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, 2016.

[14] B. Manvel, A. Meyerowitz, A. Schwenk, K. Smith, and P. Stockmeyer, "Reconstruction of sequences," *Discrete Mathematics*, vol. 94, no. 3, pp. 209–219, 1991.

[15] A. D. Scott, "Reconstructing sequences," *Discrete Mathematics*, vol. 175, no. 1-3, pp. 231–238, 1997.

[16] T. Batu, S. Kannan, S. Khanna, and A. McGregor, "Reconstructing strings from random traces," in *15th Annual ACM-SIAM Symp. Disc. Algorithms*. Society for Industrial and Applied Mathematics, 2004, pp. 910–918.

[17] M. Dudık and L. J. Schulman, "Reconstruction from subsequences," *Journal of Combinatorial Theory, Series A*, vol. 103, no. 2, pp. 337–348, 2003.

[18] J. Acharya, H. Das, O. Milenkovic, A. Orlitsky, and S. Pan, "On reconstructing a string from its substring compositions," in *IEEE Int. Symp. Inf. Theory (ISIT)*. IEEE, 2010, pp. 1238–1242.

[19] ——, "String reconstruction from substring compositions," *SIAM Journal on Discrete Mathematics*, vol. 29, no. 3, pp. 1340–1371, 2015.

[20] R. Gabrys and O. Milenkovic, "Unique reconstruction of coded sequences from multiset substring spectra," in *IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 2540–2544.

[21] Z. Chang, J. Chrisnata, M. F. Ezerman, and H. M. Kiah, "Rates of dna sequence profiles for practical values of read lengths," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7166–7177, 2017.

[22] V. I. Levenshtein, "Efficient reconstruction of sequences from their subsequences or supersequences," *J. Combin. Theory Ser. A*, vol. 93, no. 2, pp. 310–332, Feb. 2001.

[23] E. Ukkonen, "Approximate string-matching with q-grams and maximal matches," *Theoretical computer science*, vol. 92, no. 1, pp. 191–211, 1992.

[24] N. G. D. Bruijn, "A combinatorial problem," *Koninklijke Nederlandse Akademie v. Wetenschappen*, vol. 49, no. 49, pp. 758–764, 1946.

[25] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for dna storage," in *IEEE Int. Symp. Inf. Theory (ISIT)*, 2017, pp. 3115–3119.

[26] ——, "Mutually uncorrelated codes for dna storage," *IEEE Trans. Inf. Theor.*, vol. 65, no. 6, pp. 3671–3691, 2019.

[27] G. Hurlbert and G. Isaak, "On the de bruijn torus problem," *Journal of Combinatorial Theory, Series A*, vol. 64, no. 1, pp. 50–62, 1993.

[28] ——, "New constructions for de bruijn tori," *Designs, Codes and Cryptography*, vol. 6, no. 1, pp. 47–56, 1995.

[29] S. Ma, "A note on binary arrays with a certain window property (corresp.)," *IEEE Trans. Inf. Theory*, vol. 30, no. 5, pp. 774–775, 1984.

[30] B. H. Marcus, R. M. Roth, and P. H. Siegel, "An introduction to coding for constrained systems," *Lecture notes*, 2001.

[31] H. Fredricksen and I. J. Kessler, "An algorithm for generating necklaces of beads in two colors," *Discrete Math.*, vol. 61, pp. 181–188, 1986.

[32] E. Moreno, "On the theorem of fredricksen and maiorana about de bruijn sequences," *Advances in Applied Mathematics*, vol. 33, no. 2, pp. 413–415, 2004.

[33] F. Ruskey, C. D. Savage, and T. M. Y. Wang, "Generating necklaces," *J. Algorithms*, vol. 13, no. 3, pp. 414–430, 1992.

[34] O. Kupferman and G. Vardi, "Eulerian paths with regular constraints," in *mfcs16*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 58, 2016, pp. 1–62.