Duplication-Correcting Codes for Data Storage in the DNA of Living Organisms

Siddharth Jain*, Farzad Farnoud (Hassanzadeh)*, Moshe Schwartz†, and Jehoshua Bruck*
*Electrical Engineering, California Institute of Technology
Pasadena, CA 91125, U.S.A., {sidjain, farnoud, bruck}@caltech.edu
†Electrical and Computer Engineering, Ben-Gurion University of the Negev
Beer Sheva 8410501, Israel, schwartz@ee.bgu.ac.il

***Abstract*—To be considered for the 2016 IEEE Jack Keil Wolf ISIT Student Paper Award. The ability to store data in the DNA of a living organism has applications in a variety of areas including synthetic biology and watermarking of patented genetically-modified organisms. Data stored in this medium is subject to errors arising from various mutations, such as point mutations, indels, and tandem duplication, which need to be corrected to maintain data integrity. In this paper, we provide error-correcting codes for errors caused by tandem duplications, which create a copy of a block of the sequence and insert it in a tandem manner, i.e., next to the original. In particular, we present a family of codes for correcting errors due to tandem-duplications of a fixed length and any number of errors. We also study codes for correcting tandem duplications of length up to a given constant k, where we are primarily focused on the cases of $k = 2, 3$.**

I. Introduction

Data storage in the DNA of living organisms (henceforth *live DNA*) has a multitude of applications. For example, it can enable in vivo synthetic biology methods and algorithms that need "memory," e.g., to store information about their state or record changes in the environment. Embedding data in live DNA also allows watermarking genetically modified organisms (GMOs) to verify authenticity and to track unauthorized use [1], [7], as well as, labeling organisms in biological studies [14]. DNA watermarking can also be used to tag infectious agents used in research laboratories to identify sources of potential malicious use or accidental release [11]. Furthermore, live DNA can serve as a protected medium for storing large amounts of data in a compact format for long periods of time [2], [14]. An additional advantage of using DNA as a medium is that data can be disguised as part of the organisms original DNA, thus providing a layer of secrecy [3].

While the host organism provide a level of protection to the data-carrying DNA molecules as well as a method for replication, the integrity of the stored information suffers from mutations such as tandem duplications, point mutations, insertions, and deletions. Furthermore, since each DNA replication may introduce new mutations, the number of such deleterious events increases with the number of generations. As a result, to ensure decodability of the stored information, the coding/decoding scheme must be capable of a level of error correction. Motivated by this problem, we study designing codes that can correct errors arising from tandem duplications. In addition to improving the reliability of data storage in live DNA, studying such codes may help to acquire a better understanding of how DNA stores and protects biological information in nature.

Tandem duplication is the process of inserting a copy of a segment of the DNA adjacent to its original position, resulting in a *tandem repeat*. Different approaches to the problem of error-control for data stored in live DNA have been proposed in the literature. In the work of Arita and Ohashi [1], each group of five bits of information is followed by one parity bit for error detection. Heider and Barnekow [7] use the extended (8,4) Hamming code or repetition coding to protect the data. Yachie et al. [15] propose to enhance reliability by inserting multiple copies of the data into multiple regions of the genome of the host organism. Finally, Haughton and Balado [6] present an encoding method satisfying certain biological constraints.studied in a substitution mutation model. None of the aforementioned encodings, with the possible exception of repetition coding, are designed to combat tandem duplications, which is the focus of this paper. While repetition coding can correct duplication errors, it is not an efficient method because of its high redundancy.

It should also be noted that error-control for storage in live DNA is inherently different from that in DNA that is stored outside of a living organism (see [16] for an overview), since the latter is not concerned with errors arising during organic DNA replication. In this work, we ignore the potential biological effects of embedding data into the DNA. Furthermore, constructing codes that in addition to tandem duplication errors can combat other types of errors, such as substitutions, are postponed to a future work.

We also note that tandem duplication, as well as other duplication mechanisms, were studied in the context of information theory [4], [5], [9]. However, these works used duplications as a generative process, and attempted to measure its capacity and diversity. In contrast, we consider duplications as a noise source, and design error-correcting codes to combat it.

When a sequence has been corrupted by a tandem duplication channel, the challenge arises in finding the squarefree

This work was supported in part by the NSF Expeditions in Computing Program (The Molecular Programming Project).

root sequence from which the corrupted sequence could be generated. For example, for the sequence $ACGTGT$, with $GTGT$ as a tandem duplication error, the root sequence would be $ACGT$ since $ACGTGT$ can be generated from $ACGT$ by doing a tandem duplication of length 2 on GT. But there can be sequences which have more than one root. For example, the sequence $\underline{ACGCACGC}G$ can be generated from **ACG** by doing a tandem duplication of CG first, followed by a tandem duplication of $ACGC$ in $A\underline{CGCG}$ and can also be generated from **ACGCACG** by doing a tandem duplication of the suffix CG. Hence, $ACGCACGCG$ has two squarefree roots. But if we restrict the length of duplication to 2 in the previous example, then $ACGCACGCG$ has only one root i.e., $ACGCACG$. This means that the number of roots that a sequence can have depends on the set of duplication lengths that are allowed. In fact, we find that tandem duplication channels which have unique roots are the ones that allow duplications of fixed length k and the other which allow duplications of lengths bounded by 2 or 3. For all other cases, we prove in Section V, that the duplication roots are not necessarily unique. This unique root property for fixed length, 2- bounded and 3- bounded duplication channels allows us to construct error correcting codes for them.

We will first consider the tandem duplication channel with duplications of a fixed length k. For example with $k = 3$, after a tandem duplication, the sequence $ACAGT$ may become $ACAG\underline{CAG}T$, which may further become $ACA\underline{ACA}GCAGT$ where the copy is underlined. In our analysis, we provide a mapping in which tandem duplications of length k are equivalent to insertion of k zeros. Using this mapping, we demonstrate the strong connection between codes that correct duplications of a fixed length and Run-Length Limited (RLL) systems. We present constructions for codes that can correct an unbounded number of tandem duplications of a fixed length and show that our construction is optimal, i.e., of largest size.

We also consider codes for correcting duplications of bounded length. Here, our focus will be on duplication errors of length at most 2 or 3, for which we will present a construction that corrects any number of such errors. In the case of duplication length at most 2 the codes we present are optimal.

The paper is organized as follows. The preliminaries and notation are described in Section II. In Sections III and IV we present the results concerning duplications of a fixed length k and duplications of length at most k, respectively. In Section V, we characterize tandem duplication channels which do not necessarily have a unique root. Due to the page limit, all proofs appear in the full online version [10].

II. Preliminaries

We let Σ denote some finite alphabet, and Σ^* denote the set of all finite strings (words) over Σ. The unique empty word is denoted by ϵ. Given two words $x, y \in \Sigma^*$, their concatenation is denoted by xy, and x^t denotes the concatenation of t copies of x, where t is some positive integer. By convention, $x^0 = \epsilon$. We normally index the letters of a word starting with 1, i.e., $x = x_1 x_2 \dots x_n$, with $x_i \in \Sigma$. With this notation, the t-prefix and t-suffix of x are defined by

$$\mathrm{Pref}_t(x) = x_1 x_2 \dots x_t, \quad \mathrm{Suff}_t(x) = x_{n-t+1} x_{n-t+2} \dots x_n.$$

Given a string $x \in \Sigma^*$, a *tandem duplication of length k* is a process by which a contiguous substring of x of length k is copied next to itself. More precisely, we define the tandem-duplication rules, $T_{i,k} : \Sigma^* \to \Sigma^*$, as

$$T_{i,k}(x) = \begin{cases} uvvw & \text{if } x = uvw,\ |u| = i,\ |v| = k \\ x & \text{otherwise.} \end{cases}$$

Two specific sets of duplication rules would be of interest to us throughout the paper.

$$\mathcal{T}_k = \left\{ T_{i,k} \;\middle|\; i \geqslant 0 \right\},$$
$$\mathcal{T}_{\leqslant k} = \left\{ T_{i,k'} \;\middle|\; i \geqslant 0, 1 \leqslant k' \leqslant k \right\}.$$

Given $x, y \in \Sigma^*$, if there exist i and k such that $y = T_{i,k}(x)$, then we say y is a direct descendant of x, and denote it by $x \Longrightarrow_k y$. If a sequence of t tandem duplications of length k is employed to reach y from x we say y is a t-descendant of x and denote it by $x \Longrightarrow_k^t y$. More precisely, we require the existence of t non-negative integers $i_1, i_2, \dots, i_t$, such that

$$y = T_{i_t,k}(T_{i_{t-1},k}(\dots T_{i_1,k}(x) \dots)).$$

Finally, if there exists a finite sequence of tandem duplications of length k transforming x into y, we say y is a descendant of x and denote it by $x \Longrightarrow_k^* y$. We note that x is its own descendant via an empty sequence of tandem duplications.

Example 1. *Let $\Sigma = \{0,1,2,3\}$ and $x = 02123$. Since, $T_{1,2}(x) = 0212123$ and $T_{0,2}(0212123) = 020212123$, then $02123 \Longrightarrow_2 0212123$ and $02123 \Longrightarrow_2^2 020212123$, where in both expressions, the relation could be replaced with $\Longrightarrow_2^*$.* □

We define the *descendant cone* of x as

$$D_k^*(x) = \left\{ y \in \Sigma^* \;\middle|\; x \overset{*}{\underset{k}{\Longrightarrow}} y \right\}.$$

In a similar fashion we define the *t-descendant cone* $D_k^t(x)$ by replacing $\Longrightarrow_k^*$ with $\Longrightarrow_k^t$ in the definition of $D_k^*(x)$.

The set of definitions given thus far was focused on tandem-duplication rules of substrings of length exactly k, i.e., for rules from $\mathcal{T}_k$. These definitions as well as others in this section are extended in the natural way for tandem duplication rules of length up to k, i.e., $\mathcal{T}_{\leqslant k}$. We denote these extensions by replacing the k subscript with the $\leqslant k$ subscript. Thus, we also have $D_{\leqslant k}^*(x)$ and $D_{\leqslant k}^t(x)$.

Using the notation D_k^*, we restate the definition of the *tandem string-duplication system* given in [4]. Given a finite alphabet Σ, a seed string $s \in \Sigma^*$, the tandem string-duplication system is given by

$$S_k = S(\Sigma, s, \mathcal{T}_k) = D_k^*(s),$$

i.e., it is the set of all the descendants of s under tandem duplication of length k.

The process of tandem duplication can be naturally reversed. Given a string $y \in \Sigma^*$, for any positive integer, $t > 0$, we define the *t-ancestor cone* as

$$D_k^{-t}(y) = \left\{ x \in \Sigma^* \;\middle|\; x \underset{k}{\overset{t}{\Longrightarrow}} y \right\},$$

or in other words, the set of all words for which y is a t-descendant.

Yet another way of viewing the t-ancestor cone is by defining the *tandem-deduplication rules*, $T_{i,k}^{-1} : \Sigma^* \to \Sigma^*$, as

$$T_{i,k}^{-1}(y) = \begin{cases} uvw & \text{if } y = uvvw,\ |u| = i,\ |v| = k \\ \epsilon & \text{otherwise,} \end{cases}$$

where we recall ϵ denotes the empty word. This operation takes an adjacently-repeated substring of length k, and removes one of its copies. Thus, a string x is in the t-ancestor cone of y (where we assume $x, y \neq \epsilon$ to avoid trivialities) iff there is a sequence of of t non-negative integers $i_1, i_2, \ldots, i_t$, such that

$$x = T_{i_t,k}^{-1}(T_{i_{t-1},k}^{-1}(\ldots T_{i_1,k}^{-1}(y)\ldots)).$$

In a similar fashion we define the *ancestor cone* of y as

$$D_k^{-*}(y) = \left\{ x \in \Sigma^* \;\middle|\; x \underset{k}{\overset{*}{\Longrightarrow}} y \right\}.$$

By flipping the direction of the derivation arrow, we let $\Longleftarrow$ denote deduplication. Thus, if y may be deduplicated to obtain x in a single step we write $y \Longleftarrow_k x$. For multiple steps we add $*$ in superscript.

A word $y \in \Sigma^*$ is said to be *irreducible* if there is nothing to deduplicate in it, i.e., y is its only ancestor, meaning $D_k^{-*}(y) = \{y\}$. The set of irreducible words is denoted by Irr_k. We will find it useful to denote the set of irreducible words of length n by $\mathrm{Irr}_k(n) = \mathrm{Irr}_k \cap \Sigma^n$. The ancestors of $y \in \Sigma^*$ that cannot be further deduplicated, are called the *roots* of y, and are denoted by

$$R_k(y) = D_k^{-*}(y) \cap \mathrm{Irr}_k\,.$$

Note that since the aforementioned definitions extend to tandem duplication rules of length up to k, we also have $S_{\leqslant k}$, $D_{\leqslant k}^{-t}(y)$, $D_{\leqslant k}^{-*}(y)$, $\mathrm{Irr}_{\leqslant k}$, $\mathrm{Irr}_{\leqslant k}(n)$, and $R_{\leqslant k}(y)$. In some previous works (e.g., [12]), S_k is called the *uniform-bounded-duplication system*, whereas $S_{\leqslant k}$ is called the *bounded-duplication system*.

Inspired by the DNA-storage scenario, we now define error-correcting codes for tandem string-duplication systems.

Definition 2. *An $(n, M; t)_k$ code C for the k-tandem-duplication channel is a subset $C \in \Sigma^n$ of size $|C| = M$, such that for each $x, y \in C$, $x \neq y$, we have $D_k^t(x) \cap D_k^t(y) = \emptyset$. Here t stands for either a non-negative integer, or $*$. In the former case we say the code can correct t errors, whereas in the latter case we say the code can correct all errors. In a similar fashion, we can define an $(n, M; t)_{\leqslant k}$ by replacing all "k" subscripts by "$\leqslant k$".*

Assume the size of the finite alphabet is $|\Sigma| = q$. We then denote the size of the largest $(n, M; t)_k$ code over Σ by $A_q(n; t)_k$. The capacity of the channel is then defined as

$$\mathsf{cap}_q(t)_k = \limsup_{n\to\infty} \frac{1}{n} \log_q A_q(n; t)_k.$$

Analogous definitions are obtained by replacing k with $\leqslant k$ or by replacing t with $*$.

III. k-TANDEM-DUPLICATION CODES

In this section we consider tandem string-duplication systems where the substring being duplicated is of a constant length k. Such systems were studied in the context of formal languages [12] (also called *uniform-bounded-duplication systems*), and also in the context of coding and information theory [4]. In [12] it was shown that for any finite alphabet Σ, and any word $x \in \Sigma^*$, under k-tandem duplication x has a unique root, i.e., $|R_k(x)| = 1$. In the section that follows we give an alternative elementary proof to the uniqueness of the root. This proof will enable us to easily construct codes for k-tandem-duplication systems, as well as to state bounds on their parameters.

We also mention [4], in which S_k was studied from a coding and information-theoretic perspective. It was shown there that the capacity of all such systems is 0. This fact will turn out to be extremely beneficial when devising error-correcting codes for k-tandem-duplication systems.

Throughout this section, without loss of generality, we assume $\Sigma = \mathbb{Z}_q$. We also use $\mathbb{Z}_q^*$ to denote the set of all finite strings of $\mathbb{Z}_q$ (not to be confused with the non-zero elements of $\mathbb{Z}_q$), and $\mathbb{Z}_q^{\geqslant k}$ to denote the set of all finite strings over $\mathbb{Z}_q$ of length k or more.

We shall require the following mapping, $\phi_k : \mathbb{Z}_q^{\geqslant k} \to \mathbb{Z}_q^k \times \mathbb{Z}_q^*$. The mapping is defined by,

$$\phi_k(x) = (\mathrm{Pref}_k(x), \mathrm{Suff}_{|x|-k}(x) - \mathrm{Pref}_{|x|-k}(x)),$$

where subtraction is performed entry-wise over $\mathbb{Z}_q$. We easily observe that ϕ_k is a bijection between $\mathbb{Z}_q^n$ and $\mathbb{Z}_q^k \times \mathbb{Z}_q^{n-k}$ by noting that we can recover x from $\phi_k(x)$ in the following manner: first set $x_i = \phi_k(x)_i$, for all $1 \leqslant i \leqslant k$, and for $i = k+1, k+2, \ldots$, set $x_i = x_{i-k} + \phi_k(x)_i$, where $\phi_k(x)_i$ denotes the ith symbol of $\phi_k(x)$. Thus, ϕ_k^{-1} is well defined.

Another mapping we define is one that injects k consecutive zeros into a string. More precisely, we define $\zeta_{i,k} : \mathbb{Z}_q^k \times \mathbb{Z}_q^* \to \mathbb{Z}_q^k \times \mathbb{Z}_q^*$, where

$$\zeta_{i,k}(x, y) = \begin{cases} (x, u0^k w) & \text{if } y = uw,\ |u| = i \\ (x, y) & \text{otherwise.} \end{cases}$$

The following lemma will form the basis for the proofs to follow.

Lemma 3. *For all $x \in \mathbb{Z}_q^{\geqslant k}$, $\phi_k(T_{i,k}(x)) = \zeta_{i,k}(\phi_k(x))$.*

Recalling that ϕ_k is a bijection between $\mathbb{Z}_q^n$ and $\mathbb{Z}_q^k \times \mathbb{Z}_q^{n-k}$, together with Lemma 3 gives us the following corollary.

Corollary 4. *For any $x \in \mathbb{Z}_q^{\geqslant k}$, and for any sequence of non-negative integers $i_1, \ldots, i_t$,*

$$T_{i_t,k}(\ldots T_{i_1,k}(x)\ldots) = \phi_k^{-1}(\zeta_{i_t,k}(\ldots \zeta_{i_1,k}(\phi_k(x))\ldots)).$$

Corollary 4 paves the way to working in the ϕ_k-transform domain. In this domain, a tandem-duplication operation of length k translates into an insertion of a block of k consecutive zeros. Conversely, a tandem-deduplication operation of length k becomes a removal of a block of k consecutive zeros.

The uniqueness of the root, proved in [12], now comes for free. In the ϕ_k-transform domain, given $(x, y) \in \mathbb{Z}_q^k \times \mathbb{Z}_q^*$, as long as y contains a substring of k consecutive zeros, we may perform another deduplication. The process stops at the unique outcome in which the length of every run of zeros in y is reduced modulo k.

This last observation motivates us to define the following operation on a string in $\mathbb{Z}_q^*$. We define $\mu_k : \mathbb{Z}_q^* \to \mathbb{Z}_q^*$ which reduces the lengths of runs of zeros modulo k in the following way. Consider a string $x \in \mathbb{Z}_q^*$, $x = 0^{m_0}w_1 0^{m_1} w_2 \ldots w_t 0^{m_t}$, where m_i are non-negative integers, and $w_1, \ldots, w_t \in \mathbb{Z}_q \setminus \{0\}$, i.e., $w_1, \ldots, w_t$ are single non-zero symbols. We define

$$\mu_k(x) = 0^{m_0 \bmod k} w_1 0^{m_1 \bmod k} w_2 \ldots w_t 0^{m_t \bmod k}.$$

Additionally, we define

$$\sigma_k(x) = \left(\left\lfloor \frac{m_0}{k} \right\rfloor, \left\lfloor \frac{m_1}{k} \right\rfloor, \ldots, \left\lfloor \frac{m_t}{k} \right\rfloor\right) \in (\mathbb{N} \cup \{0\})^*$$

and call $\sigma(x)$ the *zero signature* of x. We note that $\mu_k(x)$ and $\sigma(x)$ together uniquely determine x.

Thus, our previous discussion implies the following corollary.

Corollary 5. *For any string $x \in \mathbb{Z}_q^{\geqslant k}$,*

$$R_k(x) = \left\{ \phi_k^{-1}(y, \mu_k(z)) \;\middle|\; \phi_k(x) = (y, z) \right\}.$$

We recall the definition of the $(0, k-1)$-RLL system over $\mathbb{Z}_q$ (for example, see [8], [13]). It is defined as the set of all finite strings over $\mathbb{Z}_q$ that do not contain k consecutive zeros. We denote this set as $C_{\mathrm{RLL}_q(0,k-1)}$. In our notation,

$$C_{\mathrm{RLL}_q(0,k-1)} = \left\{ x \in \mathbb{Z}_q^* \;\middle|\; \sigma_k(x) \in 0^* \right\}.$$

By convention, $C_{\mathrm{RLL}_q(0,k-1)} \cap \mathbb{Z}_q^0 = \{\epsilon\}$.

Given two strings, $x, x' \in \mathbb{Z}_q^{\geqslant k}$, we say x and x' are k-congruent, denoted $x \sim_k x'$, if $R_k(x) = R_k(x')$. It is easily seen that $\sim_k$ is an equivalence relation.

Corollary 6. *Let $x, x' \in \mathbb{Z}_q^*$ be two strings, and denote $\phi_k(x) = (y, z)$ and $\phi_k(x') = (y', z')$. Then $x \sim_k x'$ iff $y = y'$ and $\mu_k(z) = \mu_k(z')$.*

The following lemma appeared in [12, Proposition 2]. We restate it and give an alternative proof.

Lemma 7. *For all $x, x' \in \mathbb{Z}_q^{\geqslant k}$, we have $D_k^*(x) \cap D_k^*(x') \neq \emptyset$ if and only if $x \sim_k x'$.*

We now turn to constructing error-correcting codes. The first construction is for a code capable of correcting all errors.

Construction A. *For all $n \geqslant k \geqslant 1$ we construct*

$$C = \bigcup_{i=0}^{\lfloor n/k \rfloor - 1} \left\{ \phi_k^{-1}(y, z0^{ki}) \;\middle|\; \phi_k^{-1}(y, z) \in \mathrm{Irr}_k(n - ik) \right\}.$$

Theorem 8. *The code C from Construction A is an $(n, M; *)_k$ code, with $M = \sum_{i=0}^{\lfloor n/k \rfloor - 1} q^k M_{\mathrm{RLL}_q(0,k-1)}(n - (i+1)k)$. Here $M_{\mathrm{RLL}_q(0,k-1)}(m)$ denotes the number of strings of length m which are $(0, k-1)$-RLL over $\mathbb{Z}_q$.*

We can say more about the size of the code we constructed.

Theorem 9. *The code C from Construction A is optimal.*

The code C from Construction A also allows a simple decoding procedure, whose correctness follows from Corollary 5. Assume a word $x' \in \mathbb{Z}_q^{\geqslant k}$ is received, and let $\phi_k(x') = (y', z')$. The decoded word is simply $\tilde{x} = \phi_k^{-1}(y', \mu_k(z')0^{n-k-|\mu_k(z')|})$, where n is the length of the code C. In other words, the decoding procedure recovers the unique root of the received x', and in the ϕ_k-transform domain, pads it with enough zeros.

Encoding may be done in any of the many various ways for encoding RLL-constrained systems. The reader is referred to [8], [13] for further reading. After encoding the RLL-constrained string z, a string $y \in \mathbb{Z}_q^k$ is added, and ϕ_k^{-1} employed, to obtain a codeword.

Finally, the asymptotic rate of the code family may also be obtained, thus, obtaining the capacity of the channel.

Corollary 10. *For all $q \geqslant 2$ and $k \geqslant 1$,*

$$\mathsf{cap}_q(*)_k = \mathsf{cap}(\mathrm{RLL}_q(0, k-1)),$$

where $\mathsf{cap}(\mathrm{RLL}_q(0, k-1))$ is the capacity of the q-ary $(0, k-1)$-RLL constrained system.

As a side note, we comment that an asymptotic (in k) expression for the capacity may be given by

$$\mathsf{cap}(\mathrm{RLL}_q(0,k)) = \log_2 q - \frac{(q-1)\log_2 e}{q^{k+2}}(1 + o(1)). \quad (1)$$

IV. $\leqslant k$-TANDEM-DUPLICATION CODES

In this Section, we consider error-correcting codes that correct duplications of length at most k, which correspond to $S_{\leqslant k}$, i.e., bounded tandem string-duplication systems, where the substring being duplicated is of maximum length k. In particular, we present constructions for codes that can correct any number of duplications of length $\leqslant 3$ as well as a lower bound on the capacity of the corresponding channel. In the case of duplications of length $\leqslant 2$ we give optimal codes, and obtain the exact capacity of the channel.

It is worth noting that the systems $S_{\leqslant k}$ were studied in the context of formal languages [12] and also in the context of coding and information theory [9]. In [12], it was shown that $S_{\leqslant k}$, with $k \geqslant 4$, is not a regular language for alphabet size $|\Sigma| \geqslant 3$. However, it was proved in [9] that $S_{\leqslant 3}$ is indeed a regular language irrespective of the starting string and the alphabet size.

In this paper, we will show that strings that can be generated by bounded tandem string-duplication systems with maximum duplication length 3 have a unique deduplication root, a fact that will be useful for our code construction.

Theorem 11. *For any $z \in \Sigma^*$ we have $|R_{\leqslant 3}(z)| = 1$, following which $|R_{\leqslant 2}(z)| = |R_{\leqslant 1}(z)|$.*

In a similar fashion to the previous section, we define the following relation. We say $x, x' \in \Sigma^*$ are $\leqslant 3$-congruent, denoted $x \sim_{\leqslant 3} x'$, if $R_{\leqslant 3}(x) = R_{\leqslant 3}(x')$. Clearly $\sim_{\leqslant 3}$ is an equivalence relation. Having shown any sequence has a unique root when duplicating up to length 3, we obtain the following corollary.

Corollary 12. *For any two words $x, x' \in \Sigma^*$, if $D^*_{\leqslant 3}(x) \cap D^*_{\leqslant 3}(x') \neq \emptyset$ then $x \sim_{\leqslant 3} x'$.*

We note that unlike Lemma 7, we do not have $x \sim_{\leqslant 3} x'$ necessarily imply that their descendant cones intersect. Here is a simple example illustrating this case. Fix $q = 3$, and let $x = 012012$ and $x' = 001122$. We note that $x \sim_{\leqslant 3} x'$, since $R_{\leqslant 3}(x) = R_{\leqslant 3}(x') = \{012\}$. However, $D^*_{\leqslant 3}(x) \cap D^*_{\leqslant 3}(x') = \emptyset$ since all the descendants of x have a 0 to the right of a 2, whereas all the descendants of x' do not.

We are missing a simple operator which is required to define an error-correcting code. For any sequence $x \in \Sigma^+$, we define its k-suffix-extension to be $\xi_k(x) = x(\mathrm{Suff}_1(x))^k$, i.e., the sequence x with its last symbol repeated an extra k times.

For the remainder of the section we denote by $\mathrm{Irr}_{q;\leqslant 3}$ the set of irreducible words with respect to $\Longleftarrow_{\leqslant 3}$ over $\mathbb{Z}_q$, in order to make explicit the dependence on the size of the alphabet.

Construction B. *Let Σ be some finite alphabet. The constructed code is $C_k = \bigcup_{i=1}^{n} \{\xi_{n-i}(x) \mid x \in \mathrm{Irr}_{\leqslant k}(i)\}$.*

Theorem 13. *The code C_3 from Construction* B *is an $(n, M; *)_{\leqslant 3}$ code, where $M = \sum_{i=1}^{n} |\mathrm{Irr}_{\leqslant 3}(i)|$ and $\mathsf{cap}_q(*)_{\leqslant 3} \geqslant \mathsf{cap}(\mathrm{Irr}_{q;\leqslant 3})$.*

For $q = 3$, $\mathsf{cap}(\mathrm{Irr}_{3;\leqslant 3}) = 0.347934$.

Stronger statements can be given when the duplication length is upper bounded by 2 instead of 3.

Lemma 14 *For all $x, x' \in \Sigma^*$, we have $D^*_{\leqslant 2}(x) \cap D^*_{\leqslant 2}(x') \neq \emptyset$ if and only if $x \sim_{\leqslant 2} x'$.*

Theorem 15. *The code C_2 from Construction* B *is an optimal $(n, M; *)_{\leqslant 2}$ code, where $M = \sum_{i=1}^{n} |\mathrm{Irr}_{\leqslant 2}(i)|$ and $\mathsf{cap}_q(*)_{\leqslant 2} \geqslant \mathsf{cap}(\mathrm{Irr}_{q;\leqslant 2})$.*

V. Duplication Roots

In Section III, we stated that if the duplication length is uniform (i.e., a constant k), then every sequence has a unique root. Further in Section IV, we proved in Theorem 11 that if the duplication length is bounded by 3 (i.e. $\leqslant 3$), then again every sequence will have a unique root. In fact, the two cases proved in the paper are the only cases of tandem duplication channels that have a unique root given a sequence, namely, in all other cases, the duplication root is not necessarily unique. The characterization is stated in the following Theorem.

Theorem 16 *Given Σ with $|\Sigma| \geqslant 3$ and a non-empty set of duplication lengths U, there exists a sequence $z \in \Sigma^*$ with $|R_U(z)| > 1$ given $U \neq \{k\}$ for some $k \geqslant 1$, $U \neq \{1, 2\}$ and $U \neq \{1, 2, 3\}$.*

VI. Conclusion and Further Results

We considered codes for correcting tandem duplication errors for DNA storage in living organisms. We addressed the model of fixed-length duplications, and provided an optimal code along with encoding and decoding procedures. We also studied the case of bounded tandem duplication, particularly for duplication lengths of up to 2 or 3. In the former we provided optimal codes, whereas in the latter the construction is not always optimal. We also classified the set of duplication lengths for which there is more than one root. Further results are described in the fulll version [10]. Most notably, we address the problem of $(n, M; t)_k$ codes, i.e., codes for fixed-length duplication that can correct only a bounded number of errors. Many open problems remain. We mention in particular the study of the intricate structure we encounter in $S_{k\leqslant 3}$, where strings with a common (unique) root may still have non-intersecting descendant cones.

References

[1] M. Arita and Y. Ohashi, "Secret signatures inside genomic DNA," *Biotechnology Progress*, vol. 20, no. 5, pp. 1605–1607, 2004.

[2] F. Balado, "Capacity of DNA data embedding under substitution mutations," *IEEE Trans. Inform. Theory*, vol. 59, no. 2, pp. 928–941, Feb. 2013.

[3] C. T. Clelland, V. Risca, and C. Bancroft, "Hiding messages in DNA microdots," *Nature*, vol. 399, no. 6736, pp. 533–534, 06 1999.

[4] F. Farnoud, M. Schwartz, and J. Bruck, "The capacity of string-duplication systems," *IEEE Trans. Inform. Theory*, accepted.

[5] ——, "A stochastic model for genomic interspersed duplication," in *Proceedings of the 2015 IEEE International Symposium on Information Theory (ISIT2015), Hong Kong, China SAR*, Jun. 2015, pp. 1731–1735.

[6] D. Haughton and F. Balado, "BioCode: Two biologically compatible algorithms for embedding data in non-coding and coding regions of DNA," *BMC Bioinformatics*, vol. 14, no. 1, pp. 1–16, 2013.

[7] D. Heider and A. Barnekow, "DNA-based watermarks using the DNA-Crypt algorithm," *BMC Bioinformatics*, vol. 8, no. 1, pp. 1–10, 2007.

[8] K. A. S. Immink, *Coding Techniques for Digital Recorders*. Prentice-Hall, 1991.

[9] S. Jain, F. Farnoud, and J. Bruck, "Capacity and expressiveness of genomic tandem duplication," in *Proceedings of the 2015 IEEE International Symposium on Information Theory (ISIT2015), Hong Kong, SAR China*, Jun. 2015, pp. 1946–1950.

[10] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck, "Duplication correcting codes for data storage in the dna of living organisms," www.paradise.caltech.edu/papers/etr131.pdf.

[11] D. C. Jupiter, T. A. Ficht, J. Samuel, Q.-M. Qin, and P. de Figueiredo, "DNA watermarking of infectious agents: Progress and prospects," *PLoS Pathog*, vol. 6, no. 6, p. e1000950, 06 2010.

[12] P. Leupold, C. Martín-Vide, and V. Mitrana, "Uniformly bounded duplication languages," *Discrete Appl. Math.*, vol. 146, no. 3, pp. 301–310, 2005.

[13] D. Lind and B. H. Marcus, *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, 1985.

[14] P. C. Wong, K.-k. Wong, and H. Foote, "Organic data memory using the DNA approach," *Commun. ACM*, vol. 46, no. 1, pp. 95–98, Jan. 2003.

[15] N. Yachie, Y. Ohashi, and M. Tomita, "Stabilizing synthetic data in the DNA of living organisms," *Systems and Synthetic Biology*, vol. 2, no. 1-2, pp. 19–25, 2008.

[16] S. M. H. T. Yazdi, H. M. Kiah, E. R. Garcia, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: Trends and methods," *arXiv preprint: http://arxiv.org/abs/1507.01611*, 2015.