Capacity and Expressiveness of Genomic Tandem Duplication

Siddharth Jain, Student Member, IEEE, Farzad Farnoud, Member, IEEE, and Jehoshua Bruck, Fellow, IEEE

Abstract—The majority of the human genome consists of repeated sequences. An important type of repeated sequences common in the human genome are tandem repeats, where identical copies appear next to each other. For example, in the sequence $AGTCTGTCG$, $TGTC$ is a tandem repeat, that may be generated from $AGTCTGC$ by a tandem duplication of length $2$. In this work, we investigate the possibility of generating a large number of sequences from a seed, i.e. a small initial string, by tandem duplications of bounded length. We study the capacity of such a system, a notion that quantifies the system’s generating power. Our results include exact capacity values for certain tandem duplication string systems. In addition, motivated by the role of DNA sequences in expressing proteins via RNA and the genetic code, we define the notion of the expressiveness of a tandem duplication system as the capability of expressing arbitrary substrings. We then completely characterize the expressiveness of tandem duplication systems for general alphabet sizes and duplication lengths. In particular, based on a celebrated result by Axel Thue from 1906, presenting a construction for ternary squarefree sequences, we show that for alphabets of size $4$ or larger, bounded tandem duplication systems, regardless of the seed and the bound on duplication length, are not fully expressive, i.e. they cannot generate all strings even as substrings of other strings. Note that the alphabet of size $4$ is of particular interest as it pertains to the genomic alphabet. Building on this result, we also show that these systems do not have full capacity. In general, our results illustrate that duplication lengths play a more significant role than the seed in generating a large number of sequences for these systems.

Index Terms—Capacity, expressiveness, tandem repeats, tandem duplication, finite automaton, irreducible strings

I. INTRODUCTION

More than $50\%$ of the human genome consists of repeated sequences [9]. Two important types of common repeats are i) interspersed repeats and ii) tandem repeats. Interspersed repeats are caused by transposons. A transposon, also known as a jumping gene, is a segment of DNA that can copy or cut and paste itself into new positions of the genome. Tandem repeats are caused by slipped-strand mispairings [13]. Slipped-strand mispairings occur when one DNA strand in the duplex becomes misaligned with the other.

Tandem Repeats are common in both prokaryote and eukaryote genomes. They are present in both coding and non-coding regions and are believed to be the cause of several genetic disorders. The effects of tandem repeats on several biological processes is understood by these disorders. They can result in generation of toxic or malfunctioning proteins, chromosome fragility, expansion diseases, silencing of genes, modulation of transcription and translation [16] and rapid morphological changes [3].

A process that leads to tandem repeats, e.g. through slipped-strand mispairing, is called tandem duplication, which allows substrings to be duplicated next to their original position. For example, from the sequence $AGTCTGTCG$, a tandem duplication of length $2$ can give $AGTCGTCG$, which, if followed by a duplication of length $3$ may give $AGTCGTCGTCG$. The prevalence of tandem repeats in the human genome motivates us to study the capacity and expressiveness of string systems with tandem duplication, as defined below.

The model of a string duplication system consists of a seed, i.e., a starting string of finite length, a set of duplication rules that allow generating new strings from existing ones, and the set of all sequences that can be obtained by applying the duplication rules to the seed a finite number of times. The notion of capacity, introduced in [4], represents the average number of $m$-ary symbols per sequence symbol that are asymptotically required to encode a sequence in the string system, where $m$ is the alphabet size (for DNA sequences the alphabet size is $4$). The maximum value for capacity is $1$. A duplication system is fully expressive if all strings with the alphabet appear as a substring of some string in the system. As we will show, if a system is not fully expressive, then its capacity is strictly less than $1$.

Before presenting the notation, definitions, and the results more formally, in the rest of this section, we present two simple examples to illustrate the notions of expressiveness and capacity for tandem duplication string systems. Furthermore, we outline some useful tools as well as some of the results of the paper.

Example 1. Consider a string system on the binary alphabet $\Sigma = \{0, 1\}$ with $01$ as the seed that allows tandem duplications of length up to $2$. It is easy to check that the strings generated by this system start with $0$ and end with $1$. In fact, it can be proved that all binary strings of length $n$ which start with $0$ and end with $1$ can be generated by this system. The proof is based on the fact that every such string can be written as $0^{r_1}1^{r_2} \cdots 0^{v-1}1^{r_v}$, where each $r_i \geq 1$ and $v$ is even. A natural way to generate this string is to duplicate $01$ $\frac{v}{2}$ times and then duplicate the $0s$ and $1s$ as needed via duplications of length $1$.

Expressiveness: From the preceding paragraph, every binary
sequence \( s \) can be generated as a substring in this system as 0s1. For example, although 11010 cannot be generated by this system, it can be generated as a substring of 0110101 in the following way:

\[
01 \rightarrow 0101 \rightarrow 010101 \rightarrow 0110101.
\]

Hence this system is fully expressive.

**Capacity:** The number of length-\( n \) strings in this system is \( 2^{n-2} \). Thus, encoding sequences of length \( n \) in this system requires \( n-2 \) bits. The capacity, or equivalently the asymptotic average number of bits (since the alphabet \( \Sigma \) is of size 2) per symbol, is thus equal to 1. This is not surprising as the system generates almost all binary sequences. \( \Box \)

Observing these facts for an alphabet of size 2, one can ask related questions on expressiveness and capacity for other alphabet sizes and duplication lengths. However, counting the number of length-\( n \) sequences for capacity calculation and characterizing fully expressive systems for larger alphabets are often not straightforward tasks. In this paper, we study these questions and develop methods to answer them.

A useful tool in this study is the theory of finite automata. As a simple example note that the string system over the binary alphabet \( \{0, 1\} \) in the preceding example can be represented by the finite automaton given in Figure 1. The regular expression for the language defined by the finite automaton is

\[
R_{01} = (0^+1^+)^+,
\]

which represents all binary strings that start with 0 and end with 1. For definitions of finite automata and regular expression, the reader can refer to Section II.

One can use the Perron-Frobenius theory [7], [12] to count the number of sequences which can be generated by a finite automaton. This enables us to use finite automata as a tool to calculate capacity for some string duplication systems with tandem repeats over larger alphabet.

In our results, we find that the exact capacity of the tandem duplication string system over the ternary alphabet \( \{0, 1, 2\} \) with seed 012 and duplication length at most 3 equals

\[
\log_3 \frac{3^4 \cdot 2^2}{2} \approx 0.876036.
\]

Moreover, we generalize this result by characterizing the capacity of tandem duplication string systems over an arbitrary alphabet and a seed with maximum duplication length of 3. Namely, we show that if the maximum duplication length is 3 and the seed contains \( abc \) as a substring, where \( a, b, \) and \( c \) are distinct symbols, then the capacity

\[
\log_3 |\Sigma| 3.
\]

If such a substring does not exist in the seed, then the capacity is given by \( \log_3 |\Sigma| 2 \), unless the seed is of the form \( a^n \), in which case the capacity is 0. Some of these results are highlighted in Table I.

Our next example presents a system that, unlike that of Example I is not fully expressive.

**Example 2.** Consider a tandem duplication string system over the ternary alphabet \( \{0, 1, 2\} \) with seed 012 and maximum duplication length 3. This system is not fully expressive as it cannot generate 210, 102, or 021, even as a substring. We provide a simple proof.

**Proof:** Let \( z = \alpha \gamma \beta \), where \( \alpha, \gamma \) and \( \beta \) are strings over \( \{0, 1, 2\} \) with \(|\alpha|, |\beta| \geq 0 \), and \( 1 \leq |\gamma| \leq 3 \). Suppose \( z \) does not contain 210, 102 or 021 as a substring. We will now show that \( z^* = \alpha \gamma \gamma \beta \) does not contain 210, 102 or 021 as a substring either. We have three cases:

- For \( \gamma = a_1 \), with \( a_1 \in \{0, 1, 2\} \), the only possible new substrings generated in \( z^* \) which may not occur in \( z \) are the ones with suffix \( a_1 a_1 \) or prefix \( a_1 a_1 \).
- For \( \gamma = a_1 a_2 \), with \( a_1, a_2 \in \{0, 1, 2\} \), the only possible new substrings of length 3 generated in \( z^* \) are \( a_1 a_2 a_1 \) and \( a_2 a_1 a_2 \).
- For \( \gamma = a_1 a_2 a_3 \), with \( a_1, a_2, a_3 \in \{0, 1, 2\} \), there is no substring of length 3 in \( z^* \) which does not occur in \( z \).

Hence, if \( z \) does not contain 210, 102 or 021 as a substring, neither will \( z^* \). Since the seed 012 does not contain 210, 102 or 021 as a substring, neither will any other string in the system.

Therefore, this tandem duplication string system is not fully expressive. \( \Box \)

Based on the previous example, one may ask what happens if we start with a seed that contains one of the strings 210, 102, or 021, e.g., if we let the seed be 01210? Does the system become fully expressive? While this system can generate all strings of length 3 as substrings, the answer is still no as shown in Theorem 2. Regardless of the seed, a ternary system with maximum duplication length of 3 is not fully expressive. We show in Theorem 3 that a maximum duplication length of at least 4 is needed to arrive at a fully expressive ternary system.

While for alphabets of size 2 or 3, increasing the maximum length on duplications turns a system that is not fully expressive into one that is, for alphabets of size 4 or more, duplication systems are not fully expressive regardless of how large the bound on duplication length is. The main tool in constructing quaternary strings that do not appear independently or as substrings in these systems is Thue’s result proving the existence of ternary squarefree sequences of any length. A string is called squarefree if it does not have a tandem repeat of any length (Note that unary and binary squarefree sequences of arbitrarily large length do not exist.). The existence of such sequences underlies the significant shift in the behavior of tandem duplication systems with regards...
<table>
<thead>
<tr>
<th>Σ</th>
<th>s</th>
<th>k</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>{0, 1, 2}</td>
<td>012</td>
<td>3</td>
<td>( \approx 0.870636 )</td>
</tr>
<tr>
<td>arbitrary</td>
<td>xabcy</td>
<td>3</td>
<td>( \approx 0.870636 \log</td>
</tr>
</tbody>
</table>

TABLE I
CAPACITIES VALUES TANDEM DUPLICATION STRING SYSTEMS (\( \Sigma, s, T_{\leq k}^{tan} \)).

Here \( x, y \in \Sigma^* \), and \( a, b, c \in \Sigma \) are DISTINCT.

\[ x, y \in \Sigma^* \]

\[ \text{arbitrary} \]

\[ \text{fully expressive} \]

<table>
<thead>
<tr>
<th>Σ</th>
<th>s</th>
<th>k</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>{0, 1, 2}</td>
<td>arbitrary</td>
<td>≤ 3</td>
<td>No</td>
</tr>
<tr>
<td>{0, 1, 2}</td>
<td>012</td>
<td>≥ 4</td>
<td>Yes</td>
</tr>
<tr>
<td>Size ≥ 4</td>
<td>arbitrary</td>
<td>arbitrary</td>
<td>No</td>
</tr>
</tbody>
</table>

TABLE II
EXPRESSIVENESS OF TANDEM DUPLICATION STRING SYSTEMS (\( \Sigma, s, T_{\leq k}^{tan} \)).

to expressiveness as a function of alphabet size. Some of our results on expressiveness are summarized in Table II.

As part of this paper, we also study regular languages for tandem duplication string systems. In [11], it was shown that the tandem duplication string system is not regular if the maximum duplication length is 4 or more when the seed contains 3 consecutive distinct symbols as a substring. However for maximum duplication length 3, this question remained open. In this paper, we show in Theorem 5 that if the maximum duplication length is 3, a tandem duplication string system is regular irrespective of the seed and the alphabet size. Moreover, we characterize the exact capacity for all these systems.

**Related Work:** Tandem duplications have already been studied in [2], [3], [10]. However the main concern of these works is to determine the place of tandem duplication rules in the Chomsky hierarchy of formal languages. A study related to our work can be found in [4]. String systems with different duplication rules, namely end duplication, tandem duplication, reversed duplication and duplication with a gap, are defined and studied in [4]. In end duplication, a substring of certain length \( k \) is appended to the end of the previous string - for example, \( ACTGT \rightarrow ACTGTCT \). In reversed tandem duplication, the reverse of a substring is appended in tandem in the previous string - for example, \( ACTGT \rightarrow ACTCCTG \).

For tandem duplication string systems, the authors in [4] show that for a fixed duplication length the capacity is 0. Further, they find a lower bound on the capacity of these systems, when duplications of all lengths are allowed. In this paper, we consider tandem duplication string systems, where we restrict the maximum size of the block being tandemly duplicated to a certain finite length.

In the rest of the paper, the term tandem duplication string system refers to string duplication systems with bounded duplication length.

The rest of the paper is organized as follows. In Section II, we present the preliminary definitions and notation. In Section III, we derive our main results on capacity and expressiveness. In Section IV, we show that if the maximum duplication length is 3, then the tandem duplication string system is regular irrespective of the seed and alphabet size. Further, using the regularity of the systems, we extend our capacity results. We present our concluding remarks in Section V.

**II. PRELIMINARIES**

Let \( \Sigma \) be some finite alphabet. An \( n \)-string \( x = x_1x_2 \cdots x_n \in \Sigma^n \) is a finite sequence where \( x_i \in \Sigma \) and \( |x| = n \). The set of all finite strings over the alphabet \( \Sigma \) is denoted by \( \Sigma^* \). For two strings \( x \in \Sigma^n \) and \( y \in \Sigma^m \), their concatenation is denoted by \( xy \in \Sigma^{n+m} \). For a positive integer \( m \) and a string \( s \), \( s^m \) denotes the concatenation of \( m \) copies of \( s \). A string \( v \in \Sigma^* \) is a substring of \( x \) if \( x = uvw \), where \( u, w \in \Sigma^* \).

A string system \( S \subseteq \Sigma^* \) is represented as a tuple \( S = (\Sigma, s, T) \), where \( s \in \Sigma^* \) is a finite string called seed, which is used to initiate the duplication process, and \( T \) is a set of rules that allow generating new strings from existing ones. In other words, the string system \( S = (\Sigma, s, T) \) contains all strings that can be generated from \( s \) using rules from \( T \) a finite number of times.

A tandem duplication map \( T_{i,k} \),

\[ T_{i,k}(x) = \begin{cases} uww, & x = uvw, |u| = i, |v| = k, \\ x, & \text{else,} \end{cases} \]

creates and inserts a copy of the substring of length \( k \) which starts at position \( i+1 \). We use \( T_{i,k}^{tan} : \Sigma^* \rightarrow \Sigma^* \) and \( T_{\leq k}^{tan} \) to denote the set of tandem duplications of length \( k \), and tandem duplications of length at most \( k \), respectively.

\[ T_{i,k}^{tan} = \{ T_{i,k} : i \in \mathbb{N} \cup \{0\} \} \]

\[ T_{\leq k}^{tan} = \{ T_{i,j} : i \in \mathbb{N} \cup \{0\}, j \in \mathbb{N}, j \leq k \} \]

With this notation, the system of Example 1 can be written as \((\{0, 1\}, 01, T_{\leq 2}^{tan})\).

The capacity of the string system \( S = (\Sigma, s, T) \) is defined as

\[ \text{cap}(S) = \limsup_{n \to \infty} \frac{\log |\Sigma| |S \cap \Sigma^n|}{n}. \]

(2)

Furthermore, it is fully expressive if for each \( y \in \Sigma^* \), there exists a \( z \in S \), such that \( y \) is a substring of \( z \).

A useful tool in calculating capacity of tandem duplication string systems is deterministic finite automaton (DFA) which consists of:

- A finite set of states \( Z \).
- Alphabet \( \Sigma \).
- Transition Rule \( \delta : Z \times \Sigma \rightarrow Z \).
- Start state \( z_0 \in Z \).
- A set of accept states \( Y \).

There also exist non-deterministic finite automata. In this paper however, all the automata considered are deterministic. Henceforth, we will be using finite automaton to refer to a DFA. An example of a finite automaton is given in Figure 1.

For this finite automaton we have,

- \( Z = \{ \text{Start}, S_1, S_2 \} \).
- \( \Sigma = \{0, 1\} \).
- \( \delta(\text{Start}, 0) = S_1 \), \( \delta(S_1, 0) = S_1 \), \( \delta(S_1, 1) = S_2 \), \( \delta(S_2, 0) = S_1 \), \( \delta(S_2, 1) = S_2 \).
- \( z_0 = \text{Start} \).
The set of all possible strings that can be generated by a given DFA represent the language described by the finite automaton. This language \( L_R \) can be represented by a regular expression \( R \). Formal definitions of regular expression can be found in [6]. For the purpose of this paper, we define:

- \( R = s^* \): represents the language \( L_R \) which consists of all strings with 0 or more concatenated copies of \( s \in \Sigma^* \), i.e., \( L_R = \{ s^m \mid m \geq 0 \} \).
- \( R = s^+ \): represents the set of all strings with 1 or more concatenated copies of \( s \in \Sigma^* \), i.e., \( L_R = \{ s^m \mid m \geq 1 \} \).
- \( R = R_1 R_2 \): represents the language \( L_R \) formed by the concatenation of \( L_{R_1} \) and \( L_{R_2} \), i.e. \( L_R = \{ s_1 s_2 \mid s_1 \in L_{R_1}, s_2 \in L_{R_2} \} \).

### III. Capacity and Expressiveness

In this section, we present our results on the capacity and expressiveness of tandem duplication system with bounded duplication length. The section is divided into two parts; the first part focuses on capacity and the second on expressiveness.

#### A. Capacity

Our first result is on the capacity of a tandem duplication string system over ternary alphabet.

**Theorem 1.** For the tandem duplication string system \( S = (\{0, 1, 2\}, 012, T_{\leq 3}^{\text{tan}}) \), we have

\[
\text{cap}(S) = \log_3 \frac{3 + \sqrt{5}}{2} \approx 0.876036.
\]

**Proof:** We prove this theorem by showing that the finite automaton given in Figure 2 accepts precisely the strings in \( S \), and then finding the capacity using the Perron-Frobenius theory [7], [12].

The regular expression \( R \) for the language defined by this finite automaton is given by (see [6] for details on how to find a regular expression given a finite automaton)

\[
R = (0^+(1^2)^+ 2^+(1^2 + 2^+))^* (0^+(2^+0^+) 1^+(0^+1^+)^+ 2^+(1^2 + 2^+))^*.
\]

(3)

Let \( L_R \) be the language defined by the regular expression \( R \) (and by the finite automaton). We first show that \( L_R \subseteq S \). The direct way of doing so is to start with 012 and generate all the sequences in \( L_R \) via duplication. For simplicity of presentation, however, we take the reverse route: We show that every sequence in \( L_R \) can be transformed to 012 by a sequence deduplications. A deduplication of length \( k \) is an operation that replaces a substring \( \alpha \alpha \) by \( \alpha \) if \( |\alpha| = k \). For two regular expressions \( R_1 \) and \( R_2 \), we use \( R_1 \xrightarrow{dd\leq k} R_2 \) to denote that each sequence in \( L_{R_1} \) can be transformed into some sequence in \( L_{R_2} \) via a sequence of deduplications of length at most \( k \).

Note that \( R = B_1 B_2^* \), where

\[
B_1 = (0^+(1^2)^+ 2^+(1^2 + 2^+))^*, \quad B_2 = 0^+(2^+0^+) 1^+(0^+1^+)^+ 2^+(1^2 + 2^+)^*.
\]

We have \( B_1 \xrightarrow{dd\leq 3} 012(12)^* \xrightarrow{dd\leq 3} 012 \), since \( a^+ \xrightarrow{dd\leq 3} a \) and \((ab)^+ \xrightarrow{dd\leq 3} ab \) for all \( a, b \in \Sigma \). Furthermore,

\[
B_2 \xrightarrow{dd\leq 3} 0(20)^+ 1(01)^+ 2(12)^* \xrightarrow{dd\leq 3} 0(20)^+ 1(01)^+ 2 \xrightarrow{dd\leq 3} 0(20)^+ 1(01)^+ 2 \xrightarrow{dd\leq 3} 0(20)^+ 1(01)^+ 2 \xrightarrow{dd\leq 3} \{0120212, 0121212\}.
\]

(4)

Note for example that \( 1(01)^+ 2(12)^* \xrightarrow{dd\leq 3} 1(01)^+ 2 \) as the underlined 2 is always preceded by a 1.

We thus have \( R = B_1 B_2^* \xrightarrow{dd\leq 3} \{0120212, 0121212\} \xrightarrow{dd\leq 3} \{0120212, 0121212\} \xrightarrow{dd\leq 3} \{0120212, 0121212\} \), proving that \( L_R \subseteq S \).

To complete the proof of \( L_R \subseteq S \), we now show that \( S \subseteq L_R \). In what follows, we say a finite automaton generates a sequence \( s \), if there is a path with label \( s \) from \( Start \) to an accepting state. If an automaton generates \( uuvw \), with \( u, v, w \in \Sigma^* \), we may use \( v \) to refer both to the string \( v \) itself and to the part of the path that generates \( v \). The meaning will be clear from the context.

We show \( S \subseteq L_R \), by proving the following for the finite automaton in Figure 2:

i) It can generate 012.

ii) If the automaton can generate \( pqr \), with \( p, q, r \in \Sigma^* \) and \( |q| \leq 3 \), it can also generate \( pq^2 r \).

Condition i) holds trivially (see the path \( Start \rightarrow S_1 \rightarrow S_2 \rightarrow S_3 \) in Figure 2). In order to prove ii), we define:

- **Path Label:** Given a path \( a \) in a finite automaton, the path label \( l_a \in \Sigma^* \) is defined as the sequence obtained by concatenating the labels on the edges forming the path.
- **Path Length** is the number of edges of the path.
- **Duplicable Path:** Let \( q \) be a path that ends at state \( C \). The path \( q \) is said to be duplicable if there exists path \( q' \) that \( \text{starts and ends at state } C \) such that the path labels of \( q \) and \( q' \) are the same.
Suppose a finite automaton can generate $pqr$. If $q$ is duplicable, then $pq2r$ can also be generated by the finite automaton. As a result, to prove ii), it suffices to show that for each state $C$ in Figure 2, all paths of length 1, 2 or 3 ending in $C$ are duplicable.

Now, we show that all paths ending in $\{S_1, S_2, S_3, T_2, T_3, T_4\}$ with length $\leq 3$ are duplicable. Note that there are no nontrivial paths ending in the Start state.

Given a state $C$ and $j \in \{1, 2, 3\}$, let $P_C^j$ be the set of all length-$j$ paths ending in $C$ and let $Q_C^j$ be the set of all length-$j$ paths starting and ending in $C$. If

$$\bigcup_{a \in P_C^j} l_a = \bigcup_{a \in Q_C^j} l_a,$$

then all length-$j$ paths ending in $C$ are duplicable.

We prove that (5) holds for all states and all $j \in \{1, 2, 3\}$. This is done by computing $A$, $A^2$ and $A^3$, where $A_1$ is the (labeled) adjacency matrix of the finite automaton given in Figure 2. Here in computing the matrix products, symbols do not commute, e.g. $xy \neq yx$. The adjacency matrix $A$ and its square $A^2$, where $x$ and $y$ represent edges labeled by 0, 1, and 2, respectively, and where rows and columns correspond in order to $S_1, S_2, S_3, T_2, T_3, T_4$, are given by

$$A = \begin{bmatrix}
  x & y & 0 & 0 & 0 & 0 \\
  0 & y & 0 & x & 0 & 0 \\
  0 & 0 & x & y & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix},$$

$$A^2 = \begin{bmatrix}
  x^2 & y^2+xy & yz & 0 & yz & 0 & 0 \\
  0 & y^2+xy & z+yz & xx & x^2+xy & zy & 0 \\
  0 & 0 & z^2+yz & yz & x^2+yz & xz & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}.$$
First, suppose $|v| = 1$, say $v = a_1$. In this case, a string $x$ with the aforementioned properties does not exist as all new substrings contain the square $a_1a_1$.

Second, assume $|v| = 2$, say $v = a_1a_2$. Then $z^* = u_1a_1a_2a_1a_2w$ and $x$ either ends with $a_1a_2a_1$ or starts with $a_2a_1a_2$. Third, suppose $|v| = 3$, say $v = a_1a_2a_3$. So $z^* = u_1a_1a_2a_3a_1a_2a_3w$. Recall that $|x| \geq 4$. The string $x$ either ends with $a_1a_2a_3a_1$ or $a_2a_3a_1a_2$, or starts with $a_2a_3a_1a_2$ or $a_3a_1a_2a_3$.

So for any new 3-irreducible substring $x = x_1 \cdots x_j, x_i \in \Sigma, j \geq 4$, we have $x_1 = x_3, x_1 = x_4, x_j = x_{j-2}$, or $x_j = x_{j-3}$. Now consider the string $(0121)^0, \ell > |s|$. This sequences is 3-irreducible but does not satisfy any of the 4 properties stated for $x$. Since it is not a substring of $s$ and it cannot be generated as a new substring, it is not a substring of any $y \in S$.

Next we consider the system $(\Sigma, s, T^{\text{tan}}_{\leq k})$, $|\Sigma| \geq 4$ in Theorem 3. The proof of the theorem, uses the following lemma, which states that the expressiveness of a system also has a bearing on its capacity.

**Lemma 4.** If a string system $S$ with alphabet $\Sigma$ is not fully expressive, then $\text{cap}(S) < 1$.

**Proof:** Since $S$ is not fully expressive, there exists a $z \in \Sigma^*$ that does not appear as a substring of any $y \in S$. Let $|z| = m$ and $\mu = n - m \left\lceil \frac{n}{m} \right\rceil$. We have

$$|S \cap \Sigma^m| \leq (|\Sigma|^m - 1)^{\left\lceil \frac{n}{m} \right\rceil} |\Sigma|^\mu.$$

Since $m$ is finite, $\text{cap}(S) < 1$.

**Theorem 3.** Consider $S = (\Sigma, s, T^{\text{tan}}_{\leq k})$, where $|\Sigma| \geq 4$, $s$ is any arbitrary seed $\in \Sigma^*$ and $k$ is some finite natural number.

Then $S$ is not fully expressive, which also implies $\text{cap}(S) < 1$.

**Proof:** Suppose $z = wuw \in S$, where $|v| \leq k$, and let $z^* = wuw$ be the result of a duplication applied to $z$. Furthermore, suppose that $x = x_1 \cdots x_j$, where $x_i \in \Sigma$ and $j > k$, is a squarefree substring of $z^*$ but not $z$. Similar to the proof of Theorem 2, $x$ intersects both copies of $v$ but does not contain both. As a result, either $x_1 = x_1+i$ or $x_j = x_{j-1}$, for some $2 \leq i \leq k$.

For definiteness assume $\Sigma$ contains the symbols $\{0, 1, 2, 3\}$. The sequence $000$ is a squarefree sequence over the alphabet $\{0, 1, 2, 3\}$ and $|t| > \max \{|s|, k\}$ is not a substring of $s$ and cannot be generated as a substring since it does not satisfy the conditions stated for $x$ above. Note that such a $t$ exists since as shown by Thue [13], for an alphabet size $\geq 3$, there exists a squarefree string of any length. Hence $S$ is not fully expressive. The second part of the theorem follows from Lemma 4.

**Theorem 4.** Consider $S = ([0, 1, 2], 012, T^{(u)}_{\leq 4})$. Then $S$ is a fully expressive string system.

**Proof:** Let $S' = ([0, 1, 2], 012, T^{(u)}_{\leq 3})$. Clearly, $S' \subseteq S$. From the proof of Theorem 1 we know that the automaton of Figure 2 gives the same language as $S'$. By checking this automaton, we find that all strings of lengths 1, 2, and 3, except 021, 210, and 102, appear as a substring of some string in $S'$ and, as a result, some string in $S$. To generate 021, 210, and 102 as substrings of some string in $S$, we proceed as follows:

$$012 \rightarrow 01212 \rightarrow 012101212$$

$$012 \rightarrow 012012 \rightarrow 01202012 \rightarrow 012021202012$$

$$012 \rightarrow 012012 \rightarrow 01202012 \rightarrow 012021202012$$

where the repeats are underlined.

We have shown that all strings of length 3 appear in $S$ as substrings. Now we show the same for every string $w = w_1w_2w_3w_4$ of length 4. To do so, we study 3 cases based on the structure of $w$:

I) First, suppose that $w_4$ is the same as $w_1$, $w_2$, or $w_3$. For generating such $w$ as a substring, we first generate $w' = w_1w_2w_3$ as a substring of some string and then do a tandem duplication of $w_3$ if $w_4 = w_3$, of $w_2w_3$ if $w_4 = w_2$ and of $w_1w_2w_3$ if $w_4 = w_1$.

II) Suppose I) does not hold but $w_1 = w_2$ or $w_2 = w_3$. If the former holds, first generate $w_1w_2w_4$ and then duplicate $w_1$, and if the latter hold, generate $w_1w_2w_4$ and duplicate $w_2$.

III) If neither I) nor II) holds, then $w = 1210$, up to a relabeling of the symbols. In this case, we first generate $w' = 0121$ and then do a tandem duplication of $w'$ to get $w$. Note that $w'$ is of type considered in I).

Until now, we have shown that all strings $w$ of length at most 4 appear as a substring of some string in $S$. We use induction to complete the proof. Suppose all strings of length at most $m$ appear as a substring of some string in $S$, where $m \geq 4$. We show that the same holds for strings of length $m + 1$.

Consider an arbitrary $w = a_1a_2 \cdots a_m a_{m+1}$. We now consider two cases:

i) If all three letters in the alphabet occur at least once in $a_m a_{m-3} a_{m-2} a_{m-1} a_m$, then $a_m = 1$, and so $a_{m-2}$, $a_{m-1}$, or $a_m$ and $w$ can be generated as a substring by a tandem duplication of some suffix of size $\leq 4$ of $w' = a_1a_2 \cdots a_m$. Note that by the induction hypothesis $w'$ can be generated as a substring of some string.

ii) If at least one letter in the alphabet does not occur in $a_m a_{m-3} a_{m-2} a_{m-1} a_m$, then $a_m = 3$ and $a_{m-3} a_{m-2} a_{m-1} a_m$ is a sequence over binary alphabet and so it has a tandem repeat. Therefore $w$ can be generated as a substring by tandem duplication. Hence, we have proved the Theorem.
proof for $|\Sigma| \geq 4$, we would need to show that the $|\Sigma|-1$-ary string $a_{m-3}a_{m-2}a_{m-1}a_m$ has a tandem repeat. But this is not in general true, since there are squarefree strings over alphabets of size at least 3 per Thue’s result [15] and indeed we showed in Theorem 5 again using Thue’s result, that the system $(\Sigma, s, T^{(t)})$ is not fully expressive for $|\Sigma| \geq 4$ and any $k$.

IV. REGULAR LANGUAGES FOR TANDEM DUPLICATION STRING SYSTEMS

Tandem duplication string systems that define regular languages are easier to study due to the fact that one can use tools from the Perron-Frobenius theory [7, 12] to calculate capacity. It was proved in [11] that for $|\Sigma| \geq 3$ and maximum duplication length $\geq 4$, the language defined by tandem duplication string systems is not regular, if the seed contains $a_b c$ as a substring such that $a, b, c$ are distinct. However, if the maximum duplication length is 3, this question was left unanswered. In Theorem 5 we show that the language resulting from a tandem duplication system with the maximum duplication length of 3 is regular regardless of the alphabet size and seed. Further, in Corollary 5 we characterize the exact capacity of such tandem duplication string systems.

Theorem 5. Let $S = (\Sigma, s, T^{(t)})$, where $\Sigma$ and $s$ are arbitrary. The language defined by $S$ is regular.

Proof: We first assume that $s = a_1 \cdots a_m$, where $a_i$ are distinct. The case in which $a_i$ are not distinct is handled later.

For $3 \leq j \leq m$, let

$$R_{a_1 \cdots a_j} = a_1^j a_2^j \cdots a_j, \quad B_{a_1 a_2 a_3} = a_1^* a_2 a_3 a_1^* B_{a_1 a_2 a_3}$$

$$\vdots$$

$$B_{a_1 a_2 a_3} = a_1^* a_2 a_3 a_1^* B_{a_1 a_2 a_3}$$

where, for $a, b, c \in \Sigma$,

$$B_{abc} = a^* (c^*a^*)^* b^* (a^* b^*)^* c^* (b^* c^*)^*.$$  

We already know from Theorem 1 that $S = (\Sigma, s, T^{(t)})$ with $s = a_1 \cdots a_m$ is a regular language if $m = 3$. We show that for $m \geq 4$, $S$ represents a regular language whose regular expression is given by $R_{a_1 a_2 a_3 \cdots a_m}$. Let $L_R$ be the language defined by $R_{a_1 a_2 a_3 \cdots a_m}$. It suffices to show $L_R = S$.

We first show that $L_R \subseteq S$ by proving $R_{a_1 a_2 \cdots a_m} \xrightarrow{dd < 3} L_R \subseteq S$. To do so, we show by induction that $R_{a_1 a_2 a_3 \cdots a_i} \xrightarrow{dd < 3} \cdots a_j a_{j+1}$.
<table>
<thead>
<tr>
<th>$\Sigma$</th>
<th>$s$</th>
<th>$k$</th>
<th>Capacity</th>
<th>Fully Expressive</th>
</tr>
</thead>
<tbody>
<tr>
<td>${0}$</td>
<td>$0^m$ for some $m \geq 1$</td>
<td>1</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>${0, 1}$</td>
<td>arbitrary but not $a^m$ for some $a \in {0, 1}$</td>
<td>$\geq 2$</td>
<td>0</td>
<td>No</td>
</tr>
<tr>
<td>$</td>
<td>\Sigma</td>
<td>\geq 3$</td>
<td>arbitrary but not $a^m$ for some $a \in \Sigma$</td>
<td>2</td>
</tr>
<tr>
<td>$</td>
<td>\Sigma</td>
<td>\geq 3$</td>
<td>$\forall x, y \in \Sigma^*$, $a, b \in \Sigma$ and $a \neq b \neq c \neq a$</td>
<td>3</td>
</tr>
<tr>
<td>${0, 1, 2}$</td>
<td>012</td>
<td>4</td>
<td>?</td>
<td>No</td>
</tr>
<tr>
<td>$</td>
<td>\Sigma</td>
<td>\geq 4$</td>
<td>arbitrary</td>
<td>$\geq 4$</td>
</tr>
</tbody>
</table>

**TABLE IV**

CAPACITY AND EXPRESSIVENESS FOR DIFFERENT TANDEM Duplication STRING SYSTEMS ($\Sigma, s, T_{\leq k}^{tan}$).

![Finite automaton for $S = (\Sigma, a_1a_2a_3 \cdots a_m, T_{\leq k}^{tan})$](image)

- Label of a path of length 2: i) ending in $S_{3k}$: $a_{k+1}a_k+1$, $a_{k+1}a_{k+1}$, ii) ending in $S_{3k+1}$: $a_{k+2}a_{k+2}$, iii) ending in $T_{3k}$: $a_{k+2}a_{k+2}$.
- Label of a path of length 3: i) ending in $S_{3k-1}$: $a_{k+2}a_{k+2}$, ii) ending in $S_{3k}$: $a_{k+1}a_{k+1}a_{k+2}$, $a_{k+1}a_{k+1}a_{k+2}$, $a_{k+2}a_{k+2}a_{k+2}$, $a_{k+2}a_{k+2}a_{k+2}$, iii) ending in $S_{3k+1}a_{k+1}a_{k+2}$, $a_{k+2}a_{k+2}$, $a_{k+2}a_{k+2}$, iv) ending in $T_{3k}$: $a_{k+2}a_{k+2}a_{k+2}$, $a_{k+2}a_{k+2}a_{k+2}$, $a_{k+2}a_{k+2}a_{k+2}$, v) ending in $T_{3k+1}$: $a_{k+2}a_{k+2}a_{k+2}$.

All the path labels enumerated above are duplicable which can be verified by inspecting $A, A^2, A^3$, for paths of length 1, 2 and 3 respectively. This completes the proof of $S \subseteq L_R$.

We have proved the statement of Theorem 5 assuming all $a_i$’s in the seed $s$ to be distinct. Now assume the symbols of $s$ are not distinct. We color the symbols of $s$ so that they become distinct and obtain the system $\tilde{S} = (\tilde{\Sigma}, \tilde{s}, T_{\leq k}^{tan})$. Applying the preceding proof for distinct symbols to $\tilde{S}$, we find that $\tilde{S}$ is regular. Let $h: \tilde{\Sigma} \to \Sigma$ be a mapping that removes the colors. This mapping is called a morphism. By (14), we have that $S = h(\tilde{S})$ is also regular.

An immediate corollary on the capacity of tandem duplication string system considered in Theorem 5 is stated next.

**Corollary 5.** If for $S$ in Theorem 5 $s$ contains $abc$ as a substring such that $a, b, c \in \Sigma$ are distinct, then $\text{cap}(S) = \log_{|\Sigma|} \left( \frac{3 + \sqrt{13}}{2} \right) \simeq 0.876036 \log_{|\Sigma|} 3$. Otherwise, except for the seed $s$ of the form $a^m$, $\text{cap}(S) = \log_{|\Sigma|} 2$. If $s = a^m$, $\text{cap}(S) = 0$.

**Proof:** If $abc$ occurs as a substring of the seed $s$ such that $a, b$ and $c \in \Sigma$ are distinct, then the adjacency matrix of the finite automaton for $B_{abc}$ (strongly connected component of the finite automaton for $R_{a_1a_2a_3 \cdots a_m}$) has the maximum eigenvalue. Therefore, the $\text{cap}(S) = \log_{|\Sigma|} \left( \frac{3 + \sqrt{13}}{2} \right) \simeq 0.876036 \log_{|\Sigma|} 3$ (see [6] in the proof of Theorem 1).

If no 3 consecutive symbols in the seed $s$ are all distinct and $s \neq a^m$, then the maximum capacity component is a finite automaton only over 2 distinct symbols as in Figure 1. In other words, terms of the form $(a_i^*a_{i+1}^*)^*$ determine the capacity. Hence the capacity is $\log_{|\Sigma|} 2$.

When seed $s = a^m$, there is exactly one sequence of any given length in the system. Hence $\text{cap}(S) = 0$.

The following examples illustrate the statement of Theorem 5 and an application of its proof method.

**Example 6.** The string system $S = (\{0, 1, 2, 3\}, 0123, T_{\leq k}^{tan})$ is regular by Theorem 5 and the regular expression is given by

$$R_{0123} = 0^+1^+ (0^+1^+)^* 2^+ (1^+2^+)^* B_{012} 3^+ (2^+3^+)^* B_{123}.$$  

By Corollary 5 the capacity of this system $\simeq 0.876036 \log_{|\Sigma|} 3 \simeq 0.694242$.

**Example 7.** The string system $S = (\{0, 1, 2\}, 0112, T_{\leq k}^{tan})$ is regular by Theorem 5 and the regular expression is given by

$$R_{0112} = 0^+1^+ (0^+1^+)^* 1^+ (1^+1^+)^* B_{011} 2^+ (1^+2^+)^* B_{112}.$$  

By Corollary 5 the capacity of this system is given by $\log_{|\Sigma|} 2 \simeq 0.63093$.  

When $a_i$’s are assumed to be distinct it can be verified from the regular expression $R_{a_1...a_j}$ in the proof of Theorem 3 that the last occurrence of $a_i$ is before the first occurrence of $a_{i+3}$ for any $i = 1, 2, \ldots, j - 3$ for all $z \in S$.

The following corollary follows for maximum duplication length 2 using the same idea as in Theorem 5.

**Corollary 8.** The capacity for $S = (\Sigma, a_1 a_2 \cdots a_m, T_{2m}^{\text{tand}})$ is given by $\log_{|\Sigma|} 2$, except for the case in which seed $s = a^m$ for $a \in \Sigma$. In that case, the capacity is 0.

**Proof:** The string system $S = (\Sigma, a_1 a_2 \cdots a_m, T_{2m}^{\text{tand}})$ is regular. This can be proved using the same method as used in the proof of Theorem 5. The regular expression $Q_{a_1 a_2 \cdots a_m}$ for $m \geq 2$ is given by

$$Q_{a_1 a_2 \cdots a_m} = a_1^* a_2^* (a_1^+ a_2^*)^* (a_2 a_3)^* \cdots a_m^* (a_m^+ a_m)^*.$$

As in Proof of Corollary 5, the capacity is determined by term(s) of the form $(a_i^+ a_{i+1})^*$, except for the case when seed $s = a^m$. Therefore, the capacity for language represented by $Q_{a_1 a_2 \cdots a_m}$ is $\log_{|\Sigma|} 2$, when $s \neq a^m$ and 0 when $s = a^m$.

The finite automaton for a special case of Corollary 8 with $|\Sigma| = 3$ is given in Figure 4.

Table IV lists the capacity and expressiveness results presented in this paper and also the open question on capacity when $k \geq 4$. The expressiveness results follow from Table III.

V. Conclusion

In this paper, we showed that for tandem duplication string systems with bounded duplication length if the maximum duplication length is 3 or less, the language described by the string system is regular. Further, we computed exact capacities for these systems. Computing the capacities for bounded tandem duplication string systems with maximum duplication length greater than 3 remains an open problem.

Using Thue’s result 13, we showed that a tandem duplication string system cannot be fully expressive if the alphabet size is $\geq 4$. However, for an alphabet of size 3 or less such systems can be fully expressive. Therefore, we have completely characterized fully expressive and non-fully expressive tandem duplication string systems with bounded duplication lengths. As future work, we would like to generalize the notion of expressiveness by counting the asymptotic number of substrings of length $n$ that a string system can generate. Mathematically, we may define the expressiveness $Exp(S)$ of a string system $S$ as

$$Exp(S) = \limsup_{n \to \infty} \frac{\log_{|\Sigma|} E_n(S)}{n}.$$  

Here $E_n(S)$ represents the number of substrings of length $n$ that can be generated by $S$. It is notable here that with this definition of expressiveness, a fully expressive string system $S$ has $Exp(S) = 1$.

In this paper, we studied questions related to the generation of a diverse set of sequences from a seed given a tandem duplication rule. One can also study the minimum number of steps required to generate a given sequence of length $n$ from a squarefree seed and therefore define the notion of distance between a sequence and its seed given a tandem duplication rule. For the special case of binary sequences, we have studied this distance in [1].

It is notable here that the same sequence can be deduplicated to (or equivalently, generated from) more than one squarefree seed given a tandem duplication rule. For example: the sequence 01210212 can be deduplicated to 012 as well as 012102 under bounded tandem duplication with maximum duplication length 4 in the following way

$$0121012 \xrightarrow{dd<4} 0121 \xrightarrow{dd<4} 012.$$  

$$0121012 \xrightarrow{dd<4} 0121012.$$  

Here the underlined portion represents the repeat that is being deduplicated in a given step. This raises the question of the uniqueness of squarefree seeds for strings generated by a given tandem duplication rule. We have studied this question in [8] in the context of duplication errors.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for the thorough reviews and valuable comments. The reviewers helped us make the finite automata and proofs presented in this paper simpler and more clear. This work was supported in part by the NSF Expeditions in Computing Program - The Molecular Programming Project.

REFERENCES


Siddharth Jain (S’15) is a PhD Candidate in the department of Electrical Engineering at Caltech.

His research interests include information and coding theory, machine learning, information theoretic and statistical analysis of genomic data, pattern recognition, data compression and computational biology.

Siddharth received Bachelors and Masters degree from Indian Institute of Technology (IIT) Kanpur, India in 2013. He was awarded the proficiency medal at IIT Kanpur for excellent academic performance in Electrical Engineering.

Farzad Farnoud (Hassanzadeh) (M’13) is an Assistant Professor in the Department of Electrical and Computer Engineering and the Computer Science Department at the University of Virginia. Previously, he was a postdoctoral scholar at the California Institute of Technology.

He received his MS degree in Electrical and Computer Engineering from the University of Toronto in 2008. From the University of Illinois at Urbana-Champaign, he received his MS degree in mathematics and his Ph.D. in Electrical and Computer Engineering in 2012 and 2013, respectively. His research interests include the information-theoretic and probabilistic analysis of genomic evolutionary processes; rank aggregation and gene prioritization; and coding for flash memory and DNA storage.

Dr. Farnoud is the recipient of the 2013 Robert T. Chien Memorial Award from the University of Illinois for demonstrating excellence in research in electrical engineering and the recipient of the 2014 IEEE Data Storage Best Student Paper Award.

Jehoshua Bruck (S’86-M’89-SM’93-F’01) is the Gordon and Betty Moore Professor of computation and neural systems and electrical engineering at the California Institute of Technology (Caltech). His current research interests include information theory and systems and the theory of computation in nature.