Reconstruction of a Single String From a Part of Its Composition Multiset

Zuo Ye, Member, IEEE, and Ohad Elishco, Member, IEEE

Abstract—Motivated by applications in polymer-based data storage, we study the problem of reconstructing a string from part of its composition multiset. We give a full description of strings that cannot be uniquely reconstructed up to reversal from their multisets of all the prefix-suffix compositions. Leveraging this description, we prove that for all \( n \geq 6 \), there exists a string of length \( n \) that cannot be uniquely reconstructed up to reversal. Moreover, for all \( n \geq 6 \), we explicitly construct the set consisting of all length \( n \) strings that can be uniquely reconstructed up to reversal. As a byproduct, we obtain that any binary string can be constructed using Dyck strings and Catalan-Bertrand strings. For any given string \( s \), we provide a method to explicitly construct the set of all strings with the same prefix-suffix composition multiset as \( s \), as well as a formula for the size of this set. Furthermore, we construct two classes of composition codes that can respectively correct composition missing errors and mass-reducing substitution errors. In addition, we raise a new problem: reconstructing a string when only given its compositions of substrings of length at most \( r \). We give suitable codes under some conditions.

Index Terms—Polymer-based storage, unique string reconstruction, Dyck strings, composition codes.

I. INTRODUCTION

IN THIS information age, transmission, processing, and storage of data are playing an increasingly important role. As human life becomes more and more dependent on data, exponentially increasing amounts of data are generated every year. According to the IDC, the global datasphere will grow from 33 zettabytes in 2018 to 175 zettabytes by 2025 [1]. Therefore, the requirements for the density and longevity of storage devices are higher and higher. However, existing data storage media are not particularly advantageous in terms of these two requirements. And this is a major stimulus for the search for storage paradigms that offer storage densities at the nanoscale. Recently, several molecular storage techniques, which can provide significantly higher storage densities and longevity, have been proposed [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16]. DNA-based data storage is one such promising data storage media, but the systems come with slow and error-prone read/write platforms and have several constraints including a prohibitively expensive synthesis and sequencing process.

Polymer-based data storage systems [7], [15], as an alternative to DNA-based data storage systems, are emerging as the next-generation data storage media. Compared to DNA-based data storage systems, they offer high storage densities at lower cost and lower readout latency. In such platforms, two molecules (phosphodiesters) with significantly different masses are synthesized, one of which represents the bit 0 while the other represents the bit 1. In the write-in process, a string of binary information is encoded into a string (polyphosphodiester) of these two molecules which are stitched together by means of phosphate bonds. In the readout process, a tandem mass (MS/MS) spectrometer is used to break some of these phosphate bonds, thereby creating fragments of various lengths, and then output the masses of the resulting fragments. Then the information of masses is utilized to reconstruct the whole string. It is worth noting that the problem of reconstructing strings from substrings has been extensively investigated under a totally different setting [17], [18], [19], [20], [21], where the order of symbols in each substring is known, while in our setting, we are not given the order of symbols in substrings.

In our setting, there have been several works. The first one is [22], the results of which are obtained under the following two assumptions.

A1: One can uniquely infer the composition, i.e., the number of 0’s and 1’s, of a fragment string from its mass.

A2: When a polymer is broken down for mass spectrometry analysis, the masses of all its fragments will be observed with identical frequency.

The authors of [22] proved that when \( n \leq 7 \), or \( n \) is one less than a prime or one less than twice a prime, any string of length \( n \) can be uniquely reconstructed up to reversal from its composition multiset, i.e., the multiset consisting of the compositions of all its substrings (fragments). Later, under the same assumptions, the authors of [23] viewed the problem from a coding-theoretic perspective and demonstrated that: 1) one can add at most logarithmic redundancy to an information string so that unique reconstruction is possible, even if the information string itself cannot be uniquely reconstructed up to reversal; 2) one can add at most logarithmic redundancy to an information string so that unique reconstruction is possible,
even in the presence of errors. Following [23], in [24] the authors investigated more types of errors and constructed corresponding codes. Later, a new algorithm for the problem of string reconstruction from its composition multiset was introduced [25]. Furthermore, the authors constructed a set of strings that can be uniquely reconstructed, the size of which is larger by a linear factor than that of the code given in [23].

Recall that all of these works are based on Assumption 2. However, it is often not practical to assume that the masses of all fragments can be observed, since breaking the string in one rather than two positions is easier to perform. In [26], the authors studied the problem of reconstructing strings from the multiset consisting of the compositions of all their prefixes and suffixes. Rather than focusing on the reconstruction of a single string, they investigated a more general problem: reconstruction of a set of at most \(h\) strings from the union of their prefix-suffix composition multisets, where \(h \geq 2\) is a positive integer. By using \(B_n\) codes and Dyck strings, they explicitly constructed a code \(C(n, h)\), in which for any subset of \(C(n, h)\) of size at most \(h\), one can uniquely reconstruct this subset, where \(n\) is the length of the strings. Furthermore, they studied two types of error models, and based on \(C(n, h)\), they constructed several codes that can correct such errors.

Obviously, if no error occurs in the (prefix-suffix) composition multiset, all the works mentioned above can be viewed as special cases of a more general problem: how to reconstruct strings from a part of their composition multisets? This problem is one of the many open problems raised in [22, Section 16]. In this paper, we continue this line of research and we study two subclasses of this problem. The first one is reconstructing a single string from its prefix-suffix composition multiset. Intuitively, one can deem that this is a critical step to go further and fully understand the more general setup considered in [26]. Therefore we only concentrate on the reconstruction of a single string and ask the following questions.

**Question 1:** For which \(n\), all strings of length \(n\) can be uniquely reconstructed up to reversal?

**Question 2:** Given \(n\), can we construct all strings of length \(n\) that cannot be uniquely reconstructed up to reversal in a uniform way? What is the number of such strings?

**Question 3:** Given \(n\), can we construct all the strings of length \(n\) that can be uniquely reconstructed up to reversal in a uniform way? What is the number of such strings?

**Question 4:** Given a string \(s\), how can we find all the strings that share a common prefix-suffix composition multiset with \(s\)? What is the number of such strings?

**Question 5:** Can we construct large codes that enable unique reconstruction of each codestring from its prefix-suffix composition multiset, either in the error-free case or in the erroneous case?

In this paper, we solve these questions for the prefix-suffix composition multiset. Our basic method is the generating polynomial method first introduced in [22]. As a byproduct, we find that all binary strings can be constructed by Dyck strings and Catalan-Bertrand strings (see Corollary 29).

Secondly, we turn our attention to the second subclass of the general problem: how to reconstruct a string if we are only given its compositions of substrings of length at most \(r\). This problem is new and motivated by the reading process of tandem mass spectrometers, together with the assumption that longer strings tend to break with higher probability than short strings. We provide codes suitable for this new setup.

The paper is organized as follows. Section II is devoted to introducing the problem, the relevant terminologies and notations. In Section III, we study strings that cannot be uniquely reconstructed up to reversal. Thus, Question 1 is answered. In Section IV, we study strings that can be uniquely reconstructed up to reversal and give a uniform way to construct them. In this section, Question 2 and Question 3 are answered. In Section V, we present a method to find and enumerate all strings that share a common prefix-suffix composition multiset with a given string. Thus, answering Question 4. Based on the results in previous sections, in Section VI we construct several codes enabling unique reconstruction of each codestring. Some of them can combat errors of the type described in [26, Section VI]. This answers Question 5 affirmatively. In Section VII we present a new model called the \(r\)-length limited composition multiset model, and provide a simple coding scheme for the case \(r\) is a linear function of the length of the words \(n\). In Section VIII, we conclude this paper and discuss future problems.

**II. Preliminaries**

Let \(\Sigma_2 \equiv \{0, 1\}\) be the binary alphabet. For any string \(s \in \Sigma_2^n\), denote by \(c(s) = 0^n w 1^w\) the composition of \(s\), where \(w = \text{wt}_{\Sigma_2}(s)\) is the Hamming weight of \(s\). For any string \(s\) of length \(n\) and any \(1 \leq i \leq n\), the \(i\)-th component of \(s\) is denoted by \(s_i\) (i.e., \(s_i = s_n \cdots s_i\)), unless otherwise specified. If \(s \in \Sigma_2^n\), we denote \(s_i^j = s_i \cdots s_j\) for any \(1 \leq i \leq j \leq n\). If \(j < i \leq 1\), we define \(s_i^j\) to be the empty string \(\epsilon\). In particular, \(s_1^i\) and \(s_i^n\) are called the prefix and suffix of \(s\) respectively.

**Definition 1:** For a string \(s \in \Sigma_2^n\), we denote by \(\mathcal{M}(s)\) the multiset of compositions of all prefixes and suffixes of \(s\), i.e.,

\[
\mathcal{M}(s) := \left\{ (c(s_1^j), c(s_i^n)) : 1 \leq j \leq n, 1 \leq i \leq n \right\}. \]

We call \(\mathcal{M}(s)\) the prefix-suffix composition multiset of \(s\).

For example, if \(s = 01010\), then

\[
\mathcal{M}(s) = \left\{ (0, 0), (0, 0, 01), (0, 0, 01, 0^21), (0, 0, 01, 0^21, 0^21^2), (0, 0, 01, 0^21^2, 0^31^2) \right\}.
\]

For any \(s = s_1 \cdots s_n\), we denote \(s^* = s_n s_{n-1} \cdots s_1\) and call it the reversal of \(s\). It is clear that \(\mathcal{M}(s) = \mathcal{M}(s^*)\) for any string \(s\). We say a string \(s\) can be uniquely reconstructed up to reversal if there is no string \(t\) other than \(s\) and \(s^*\), such that \(\mathcal{M}(t) = \mathcal{M}(s)\).

With a given string \(s\), we can associate a bivariate polynomial \(P_s(x, y)\) with coefficients in \(\{0, 1\}\). In this paper, for a given bivariate polynomial \(P(x, y)\), we always let \((P(x, y))_i\)

\[1\] In this paper, we let \(\emptyset\) denote a multiset.
denote the term of total degree $i$ of $P(x, y)$ if this term is unique. The next polynomial was first introduced in [22].

**Definition 2:** Let $s \in \Sigma_2^n$ be a string. Define the generating polynomial of $s$, denoted $P_s(x, y)$, as $P_s(x, y)_0 \triangleq 1$ and for any $1 \leq i \leq n$,

$$P_s(x, y)_i \triangleq \begin{cases} y \cdot (P_s(x, y))_{i-1}, & \text{if } s_i = 0, \\ x \cdot (P_s(x, y))_{i-1}, & \text{if } s_i = 1. \end{cases}$$

For example, if $s = 010100$, then $P_s(x, y) = 1 + y + xy + xy^2 + x^2y^2 + x^2y^3$. By definition, we can also write $P_s(x, y)$ as

$$P_s(x, y) = \sum_{i=0}^{n} x^{a_i} y^{i-a_i},$$

where $a_0 \triangleq 0$ and $a_i = \text{wt}_H(s_i^1)$ for any $1 \leq i \leq n$. On the other hand, since $\text{wt}_H(s_i^{-1}) \leq \text{wt}_H(s_i^1) \leq \text{wt}_H(s_i^1) + 1$ for each $s \in \Sigma_2^n$ and $1 \leq i \leq n$, we have the following claim.

**Claim 3:** Given a polynomial

$$P(x, y) = \sum_{i=0}^{n} x^{a_i} y^{i-a_i},$$

where $0 \leq a_i \leq i$ for each $0 \leq i \leq n$, it is a generating polynomial of some string if and only if $a_{i-1} \leq a_i \leq a_{i-1} + 1$ for each $1 \leq i \leq n$.

Let $P(x, y)$ be as in Equation (2). Assume $a_n = \max\{a_i : 0 \leq i \leq n\}$ and $n - a_n = \max\{i - a_i : 0 \leq i \leq n\}$. The reciprocal of $P(x, y)$ is defined to be the polynomial $P^*(x, y) = x^{a_n} y^{n-a_n} P(\frac{1}{x}, \frac{1}{y})$. Then we have

$$P^*(x, y) = \sum_{i=0}^{n} x^{a_n-a_{i-1}} y^{i-(a_n-a_{i-1})}.$$  

In particular,

$$P^*(x, y) = \sum_{i=0}^{n} x^{w-a_{i-1}} y^{i-(w-a_{i-1})},$$

where $w = \text{wt}_H(s)$.

The relationship between $P_s(x, y)$ and $P^*_s(x, y)$ is described in the following lemma.

**Lemma 4** [22, Section 7]: For any string $s$, we have $P^*_s(x, y) = P^*_s(x, y)$.

**Remark 5:** It is immediate to verify that $(P_s(x, y))_i = (P^*_s(x, y))_i$ if and only if $(P_s(x, y))_{n-i} = (P^*_s(x, y))_{n-i}$ for every $0 \leq i \leq \lfloor n/2 \rfloor$ and $s \in \Sigma_2^n$. By Lemma 4 and the relationship between a generating polynomial and its corresponding string, this is equivalent to $\text{wt}_H(s_i^1) = \text{wt}_H((s^*)^i_1)$ if and only if $\text{wt}_H(s_i^{-1}) = \text{wt}_H((s^*)^{-1}_i)$.

This, in turn, is equivalent to $\text{wt}_H(\hat{s}_i^1) = \text{wt}_H(\hat{s}_i^{-1})$ if and only if $\text{wt}_H(\hat{s}_i^{n-1}) = 1$.

The following lemma, which can be easily verified, is key to our purpose.

**Lemma 6:** For any two strings $s$ and $t$, $M(s) = M(t)$ if and only if $P_s(x, y) + P^*_s(x, y) = P_t(x, y) + P^*_t(x, y)$, i.e., $P_s(x, y) + P^*_t(x, y) = P_t(x, y) + P^*_s(x, y)$.

In the following section we use Lemma 6 to identify strings that are not uniquely reconstructible up to reversal.

III. **Strings That Cannot Be Uniquely Reconstructed up to Reversal**

Let $s \in \Sigma_2^n$ be a string of length $n$. By Lemma 6, to find a string $t \neq s, s^*$ such that $M(s) = M(t)$, we need to find a generating polynomial $Q(x, y)$ such that

$$P_s(x, y) + P^*_s(x, y) = Q(x, y) + Q^*(x, y)$$

and $Q(x, y) \notin \{P_s(x, y), P^*_s(x, y)\}$. To be specific, our goal includes three aspects:

(G1) Find a polynomial $Q(x, y)$ with coefficients in $\{0, 1\}$ that satisfies Equation (5);

(G2) Make sure that $Q(x, y) \neq P_s(x, y)$, or $P^*_s(x, y)$;

(G3) Make sure that $Q(x, y)$ is a generating polynomial.

We begin our discussion with (G1) and (G2), while (G3) will be discussed in Section III-A and Section III-B.

First, we describe how to find a polynomial $Q(x, y)$ that satisfies (G1). Since the coefficients of $P_s(x, y)$ and $Q(x, y)$ are 0 or 1, Equation (5) holds if and only if $(Q(x, y))_i = (P_s(x, y))_i$ or $(Q(x, y))_i = (P^*_s(x, y))_i$ for each $i$. In other words, the polynomial $Q(x, y)$ must be obtained by swapping some $(P_s(x, y))_i$’s and the corresponding $(P^*_s(x, y))_i$’s. This gives us an idea of how to generate different strings with the same prefix-suffix multisets.

Let us define the set

$$D \triangleq \{i : 1 \leq i \leq n, (P_s(x, y))_i \neq (P^*_s(x, y))_i\}.$$  

Since it is meaningless to swap $(P_s(x, y))_i$ and $(P^*_s(x, y))_i$ if $(P_s(x, y))_i = (P^*_s(x, y))_i$, the set $D$ consists of all the terms in $P_s(x, y)$ that can be swapped with their corresponding terms in $P^*_s(x, y)$ to generate a polynomial as in (G1). Notice that in $D$, the values of $i$ range from 1 to $n-1$ since it always hold that $(P_s(x, y))_n = (P^*_s(x, y))_n$ and $(P_s(x, y))_0 = (P^*_s(x, y))_0$.

If $D = \emptyset$, then $(P_s(x, y))_i = (P^*_s(x, y))_i$ for all $1 \leq i \leq n$, which is equivalent to $s = s^*$ and thus $s$ can be uniquely reconstructed up to reversal. So for our aim, we assume $D \neq \emptyset$. Moreover, by Remark 5, $i \in D$ if and only if $n - i \in D$.

In order to denote the terms in $P_s(x, y)$ that we aim to swap with their corresponding counterparts in $P^*_s(x, y)$, we use subsets of $D$ in the following way. For a subset $S$ of $D$, let $Q_{S_1} (x, y)$ be obtained by replacing all the $(P_s(x, y))_j$’s with $(P^*_s(x, y))_j$’s, and let $Q_{S_2} (x, y)$ be obtained by replacing all the $(P^*_s(x, y))_j$’s with $(P_s(x, y))_j$’s, where $j \in S$. In other words,

$$Q_{S_1} (x, y) = P_s(x, y) - \sum_{j \in S} (P_s(x, y))_j + \sum_{j \in S} (P^*_s(x, y))_j,$$

$$Q_{S_2} (x, y) = P^*_s(x, y) - \sum_{j \in S} (P^*_s(x, y))_j + \sum_{j \in S} (P_s(x, y))_j.$$  

We mention here that one needs to be careful when choosing the subsets $S$. This is because after swapping the terms, the equality $Q_{S_1} (x, y) = Q_{S_2} (x, y)$ needs to hold.

---

2Throughout the paper, polynomial operations are performed in the context of integer polynomials.

3The set $D$ depends on the choice of $s$. We omit $s$ from the notation $D$ since it is always clear from the context which string we are analyzing.
and the resulting polynomials need to be generating polynomials. We first consider the equality \( Q_{S,1}(x, y) = Q_{S,2}(x, y) \), as demonstrated in the following example.

**Example 7:** Let \( s = 1000 \). Then \( P_a^*(x, y) = 1 + x + xy + xy^2 + xy^3 \) and \( Q_a^*(x, y) = 1 + y + y^2 + y^3 + xy^3 \). We have \( D = \{1, 2, 3\} \). Let \( S = \{1\} \) and \( T = \{1, 3\} \). Then

\[
\begin{align*}
Q_{S,1}(x, y) &= 1 + y + xy + xy^2 + xy^3, \\
Q_{S,2}(x, y) &= 1 + x + y^2 + y^3 + xy^3,
\end{align*}
\]

and

\[
\begin{align*}
Q_{T,1}(x, y) &= 1 + y + xy + y^3 + xy^3, \\
Q_{T,2}(x, y) &= 1 + x + y^2 + xy^2 + xy^3.
\end{align*}
\]

Clearly, \( Q_{S,1}(x, y) \neq Q_{S,2}(x, y) \) and \( Q_{T,1}(x, y) = Q_{T,2}(x, y) \). Notice that \( Q_{S,1}(x, y) \) is the only generating polynomial.

The next lemma shows how to select such subsets \( S \) for which \( Q_{S,1}(x, y) = Q_{S,2}(x, y) \). According to \( Q \) for any \( n \) and \( j \), if \( j \in S \) and only if \( j \in S \), for each \( j \).

**Lemma 8:** \( Q_{S,1}(x, y) = Q_{S,2}(x, y) \) if and only if \( S \) satisfies:

- \( j \in S \) if and only if \( n - j \in S \), for each \( j \).

**Proof:** Suppose that \( P_a(x, y) \) is given as in Equation (1). Then \( P_a^*(x, y) \) is given in Equation (4). By the definition of \( Q_{S,1}(x, y) \) and \( Q_{S,2}(x, y) \), we have

\[
Q_{S,1}(x, y) = \sum_{i=0}^{n} x^{i} y^{j-i}, \quad \text{and} \quad Q_{S,2}(x, y) = \sum_{i=0}^{n} x^{i} y^{j-i},
\]

where \( d_i = a_i \), \( d_i' = w - a_{n-i} \) if \( i \notin S \) and \( d_i = w - a_{n-i} \), \( d_i' = a_{j} \) if \( i \in S \).

We first prove the \( \Leftarrow \) direction. By Equation (3), it is enough to show that \( d_i' = d_i - d_{n-i} \) for all \( 0 \leq i \leq n \), which is clear (notice that \( d_n = w \)).

Next, we prove the \( \Rightarrow \) direction. Suppose on the contrary that there is a \( j \in S \) such that \( n - j \notin S \). According to Equation (3), we have \( (Q_{S,1}^{a}(x, y))_{j} = x^{w-d_{n-j}} y^{j-(w-d_{n-j})} \).

Since \( n - j \notin S \subseteq D \), we have \( d_{n-j} = a_{n-j} \). Comparing with Equation (4), we conclude that \( (Q_{S,1}^{a}(x, y))_{j} = (P_{a}^{*}(x, y))_{j} \).

On the other hand, we have \( (Q_{S,2}(x, y))_{j} = (P_{b}^{*}(x, y))_{j} \) and \( (P_{b}(x, y))_{j} \neq (P_{a}^{*}(x, y))_{j} \) since \( j \in S \). Therefore, \( (Q_{S,1}(x, y))_{j} \neq (Q_{S,2}(x, y))_{j} \), which is a contradiction. \( \square \)

By Lemma 8, to find a polynomial \( Q(x, y) \) satisfying Equation (5), it is enough to consider subsets of \( D \cap \{1, \ldots, [n/2]\} \). So in the rest of this section, we always assume that \( S \) is a subset of \( D \cap \{1, \ldots, [n/2]\} \) and, by abuse of notation, denote \( Q_{S,1}(x, y) = Q_{S,2}(x, y) \). Let \( Q_{S,1}(x, y) = Q_{S,2}(x, y) = Q_{S,1}(n-2S,x,y) \), where \( n - S = \{n - j : j \in S\} \). For each such a subset \( S \), we can obtain a polynomial \( Q(x, y) = Q_{S,1}(x, y) \) (or \( Q(x, y) = Q_{S,2}(x, y) \)) which satisfies Equation (5).

Next, we consider (G2). The following claim, whose proof is clear from the definitions of \( Q_{S,1}(x, y) \) and \( Q_{S,2}(x, y) \), describes the property of \( S \) such that \( Q_{S,1}(x, y) \neq P_b(x, y) \) or \( P^*_a(x, y) \). This claim will be frequently used in the next two subsections.

**Claim 9:** Let \( s \in \Sigma_2 \) be a string and \( D \) be the set defined as in Equation (6). Let \( S \) be a subset of \( D \cap \{1, \ldots, [n/2]\} \).

(i) Suppose that \( S \) is nonempty, then \( Q_{S,1}(x, y) \) (and \( Q_{S,2}(x, y) \)) \( \notin \{P_b(x, y), P^*_a(x, y)\} \) if and only if there exists a \( k \in D \cap \{1, \ldots, [n/2]\} \) such that \( k \notin S \).

(ii) The polynomial \( Q_{S,1}(x, y) \) is a generating polynomial if and only if \( Q_{S,2}(x, y) \) is a generating polynomial, since \( Q_{S,1}(x, y) = Q_{S,2}(x, y) \).

Now what remains is to make sure that \( Q(x, y) \) is a generating polynomial. We first analyze the case where \( |S| = 1 \) and then analyze general \( S \).

**A. A Specific Case**

In this subsection, we study the specific case where \( |S| = 1 \). On the one hand, this specific case inspires our discussion of the general case in the next subsection. On the other hand, in this specific case, we can obtain a more accurate description of the structure of strings.

Let \( P_b(x, y) \) and \( P^*_a(x, y) \) be given in Equation (1) and Equation (4) respectively. Let \( S = \{j\} \), where \( 1 \leq j \leq [n/2] \). By Claim 3, \( Q_{S,1}(x, y) \) is a generating polynomial if and only if the following inequalities hold:

\[
\begin{align*}
&aj_{-1} \leq w - a_{n-j} \leq aj_{-1} + 1 \\
&w - a_{n-j} \leq aj_{+1} \leq w - a_{n-j} + 1 \\
&an_{-1} \leq w - aj \leq an_{-1} + 1 \\
&w - aj_{-1} \leq aj_{+1} \leq w - aj_{+1} + 1
\end{align*}
\]

Simplifying these inequalities, we get

\[
\begin{align*}
&w - a_{n-j-1} - 1 \leq aj \leq w - a_{n-j-1} \\
&w - a_{n-j+1} \leq aj \leq w - a_{n-j+1} + 1 \\
&w - aj_{-1} - 1 \leq an_{-j} \leq w - aj_{-1} \\
&w - aj_{+1} \leq an_{-j} \leq w - aj_{+1} + 1
\end{align*}
\]

Putting everything together, we get the following lemma.

**Lemma 10:** Let \( s \in \Sigma_2 \) be a string and let \( P_b(x, y) \) be its generating polynomial. Then there exists a string \( t \) such that \( t \neq s, s^* \) and \( M(t) = M(s) \) if there exists some \( 1 \leq j \leq [n/2] \) such that the following three conditions hold:

(i) Equation (7) holds.

(ii) \( aj \neq w - an_{-j} \), i.e., \( wt_H(s_j') \neq wt_H(s_{n-j+1}') \).

(iii) There exists some \( k \in \{1, \ldots, [n/2]\} \) such that \( (P_b(x, y))_k \neq (P^*_a(x, y))_k \), i.e., \( wt_H(s_{n-k+1}') \neq wt_H(s_{n-k+1}) \).

**Proof:** Let \( S = \{j\} \). Condition (ii) implies \( S \subseteq D \cap \{1, \ldots, [n/2]\} \). So the two polynomials \( Q_{S,1}(x, y) \) and \( Q_{S,2}(x, y) \) are well-defined. Condition (i) ensures that \( Q_{S,1}(x, y) \) is a generating polynomial. Let \( t \) be the string corresponding to \( Q_{S,1}(x, y) \). Then by the definition of \( Q_{S,1}(x, y) \), we have \( P_b(x, y) + P^*_a(x, y) = Q_{S,1}(x, y) + Q_{S,1}(x, y) \), which implies \( M(t) = M(s) \) by Lemma 6. By Condition (iii) and Claim 9, we have \( t \neq s, s^* \).

With Lemma 10, we can now describe the explicit structure of a class of strings that cannot be uniquely reconstructed up to reversal.

**Theorem 11:** Let \( s \in \Sigma_2 \), where \( n \geq 2 \). Then \( s \) satisfies all the conditions in Lemma 10 if and only if there exist distinct
Consider \( k = 3 \), since \( \text{wt}_H(s^7_5) = \text{wt}_H(s^0_6) \). Then by Theorem 11, we get four strings: 101010, 100110, 010101 and 011001. According to Theorem 11, each of these four strings cannot be uniquely reconstructed up to reversal. Furthermore, it can be directly verified that \( M(101010) = M(010101) = M(011001) \).

If \( j = 3 \), we cannot find a desired \( k \neq j \), \( n - j \) for which \( \text{wt}_H(s^k_j) \neq \text{wt}_H(s^n_{k+1}) \), since \( \text{wt}_H(s^1_1) = \text{wt}_H(s^5_6) \) and \( \text{wt}_H(s^2_1) = \text{wt}_H(s^6_7) \) by Theorem 11.

If \( j = 3 \), then we can only consider \( k = 1 \) since \( \text{wt}_H(s^1_3) = \text{wt}_H(s^6_9) \) by Theorem 11. Then by Theorem 11 we get four strings: 101010, 011001, 010101 and 100110. Thus, for \( n = 6 \), there are at least 4 strings that cannot be uniquely reconstructed up to reversal and they share the same prefix-suffix composition multiset.

2) Consider \( n = 7 \), which implies \( 1 \leq j \leq 2 \). If \( j = 1 \), using Theorem 11 we may consider only \( k = 3 \).

Theorem 11 we get eight strings: 1011010, 1010010, 1001110, 1000110, 0101011, 0100101, 0110101 and 0110001. By Theorem 11, each of these eight strings cannot be uniquely reconstructed up to reversal. Furthermore, it can be directly verified that

\[
M(1011010) = M(1001100) = M(1001110) = M(1001001) = M(0111001) = M(0111000),
\]

and that

\[
M(1010010) = M(0100101) = M(0100110) = M(0110001).
\]

If \( j = 2 \), we cannot find a desired \( k \) since \( \text{wt}_H(s^k_1) = \text{wt}_H(s^7_5) \) by Theorem 11. Thus, for \( n = 7 \), there are at least 8 strings that cannot be uniquely reconstructed up to reversal, and among them, four share a common prefix-suffix composition multiset, and the other four share another common prefix-suffix composition multiset.

Later in Corollary 18, we will prove that for \( n = 6 \) (\( n = 7 \)), there are exactly 4 (exactly 8) strings which cannot be uniquely reconstructed up to reversal.

Example 12 shows that when \( n = 6, 7 \), there exist length-\( n \) strings that cannot be uniquely reconstructed up to reversal. The following corollary asserts that this is true for all \( n \geq 6 \).

Corollary 13: For any \( n \geq 6 \), there always exists a string of length \( n \) that cannot be uniquely reconstructed up to reversal.

Proof: Let \( s = 100s_4 \cdots s_{n-3}110 \). Then \( s \) satisfies (i) in Theorem 11 with \( j = 1 \) and \( k = 3 \).

B. The General Case

Theorem 11 only gives a sufficient condition for a string to be non-uniquely reconstructible up to reversal. This condition is not necessary. The following example gives a string that cannot be uniquely reconstructed up to reversal but does not satisfy Theorem 11.

Example 12: 1) Consider \( n = 6 \), which implies \( 1 \leq j \leq 3 \). If \( j = 1 \), using Theorem 11 we can only
Example 14: Let $s = 110100100110 \in \Sigma_n^2$. Then $s^* = 011001001011$ and

$$P_s(x, y) = 1 + x + x^2 + x^2y + x^3y + x^3y^2 + x^3y^3 + x^4y^3 + x^4y^4 + x^4y^5 + x^4y^6 + x^5y^6,$$

$$P_s^*(x, y) = 1 + y + xy + x^2y + x^2y^2 + x^3y^3 + x^3y^4 + x^3y^5 + x^3y^6 + x^4y^6.$$  

Let $t$ be the string corresponding to $Q_{S_1}(x, y)$. Then $t = 01100100100111$ and $M(t) = M(s)$. Clearly, $t \neq s$ and $t \neq s^*$.  

In the above example, if we let $S = \{1\}$, then

$$Q_{S_1}(x, y) = 1 + y + xy + x^2y + x^2y^2 + x^3y^3 + x^4y^3 + x^4y^4 + x^4y^5 + x^4y^6 + x^5y^6,$$

$$Q_{S_2}(x, y) = 1 + x + x^2 + x^2y + x^2y^2 + x^3y^3 + x^4y^3 + x^4y^4 + x^4y^5 + x^5y^5 + x^6y^6.$$  

However, neither $Q_{S_1}(x, y)$ nor $Q_{S_2}(x, y)$ is a generating polynomial. Therefore, we should first find the conditions under which $Q_{S_1}(x, y)$ and $Q_{S_2}(x, y)$ are both generating polynomials.  

Let $I \subseteq \{1, \ldots, n\}$. If there are $1 \leq i \leq j \leq n$ such that $I = \{i, i+1, \ldots, j\}$, we say $I$ is an interval. Any nonempty subset $S$ of $D = S \{\{n/2\}\}$ is a union of maximal intervals, i.e., $S = \bigcup I_i$, where each $I_i$ is an interval and $I_i \cup I_j$ is not an interval whenever $i \neq j$. For example, in Example 14, let $S_1 = \{1, 2\}$ and $S_2 = \{1, 2, 4, 5\}$. Then $S_1$ is a maximal interval and $S_2 = I_1 \cup I_2$, where $I_1 = \{1, 2\}$ and $I_2 = \{4, 5\}$. Both $I_1$ and $I_2$ are maximal intervals since $I_1 \cup I_2$ is not an interval.  

Firstly, we consider the case when $S$ itself is an interval. Lemma 15 below is a generalization of Lemma 10 (i).  

Lemma 15: Let $s \in \Sigma_n^2$ be a string and let $P_s(x, y)$ be its generating polynomial. Let $S = \{i_1, \ldots, i_s\}$ be an interval of $D \cap \{1, \ldots, \lfloor n/2 \rfloor\}$. Then $Q_{S_1}(x, y)$ and $Q_{S_2}(x, y)$ are both generating polynomials if and only if $i_1 - 1, i_2 + 1 \notin D \cap \{1, \ldots, \lfloor n/2 \rfloor\}$.  

Proof: The $\Rightarrow$ direction is clear from Claim 3 and the definition of $Q_{S_1}(x, y)$ and $Q_{S_2}(x, y)$. So we only need to prove the $\Leftarrow$ direction.  

According to the definition of $D$, we have $(P_s(x, y))_{i_j} \neq (P_s^*(x, y))_{i_j}$, i.e., $\text{wt}_H(s^*)_{i_j} \neq \text{wt}_H((s^*)^*_{i_j})$, for all $j \in S$. Suppose $(P_s(x, y))_{i_{j-1}} = x^ay^b$, $(P_s^*(x, y))_{i_{j-1}} = x^cy^d$, $(P_s(x, y))_{i_j} = x^ey^f$ and $(P_s^*(x, y))_{i_j} = x^gy^h$. According to the definition, we have $(Q_{S_1}(x, y))_{i_{j-1}} = (P_s(x, y))_{i_{j-1}}$ and $(Q_{S_2}(x, y))_{i_{j-1}} = (P_s^*(x, y))_{i_{j-1}}$. Since $Q_{S_1}(x, y)$ and $Q_{S_2}(x, y)$ both give a string, by Claim 3, we have

$$a \leq a' \leq a + 1,$$

$$c \leq c' \leq c + 1.$$  

Notice that $a' > c'$ or $a' < c'$ since $\text{wt}_H(s^*)_{i_j} \neq \text{wt}_H((s^*)^*_{i_j})$. Then from Equation (9) we can conclude that $a = c$ and hence $(P_s(x, y))_{i_{j-1}} = (P_s^*(x, y))_{i_{j-1}}$, which means $i_1 - 1 \notin D \cap \{1, \ldots, \lfloor n/2 \rfloor\}$.  

If $i_2 = \lceil n/2 \rceil$, it is clear that $i_2 + 1 \notin D \cap \{1, \ldots, \lfloor n/2 \rfloor\}$.  

If $i_2 < \lfloor n/2 \rfloor$, we have $(Q_{S_1}(x, y))_{i_{j+1}} = (P_s(x, y))_{i_{j+1}}$ and $(Q_{S_2}(x, y))_{i_{j+1}} = (P_s^*(x, y))_{i_{j+1}}$. In a similar way we can also prove $(P_s(x, y))_{i_{j+1}} = (P_s^*(x, y))_{i_{j+1}}$. Therefore, $i_2 + 1 \notin D \cap \{1, \ldots, \lfloor n/2 \rfloor\}$.  

Let $S = \bigcup I_i$ be a nonempty subset of $D \cap \{1, \ldots, \lfloor n/2 \rfloor\}$, where each $I_i$ is a maximal interval. Recall that Claim 3 asserts that a polynomial as in Equation (2) is a generating polynomial if and only if $a_i - 1 = a_i$ for all $1 \leq i \leq n$. Therefore, we have the following claim.  

Claim 16: Let $S$ be as above. Then $Q_{S_1}(x, y)$ (or $Q_{S_2}(x, y)$) is a generating polynomial if and only if for each $I_i$, $Q_{I_1,1}(x, y)$ (or $Q_{I_1,2}(x, y)$) is a generating polynomial.  

Claim 9 and Claim 16 lead to the next lemma.  

Lemma 17: Let $s$ be a string of length $n$ with generating polynomial $P_s(x, y)$. Let $S = \bigcup I_i$ be a nonempty subset of $D \cap \{1, \ldots, \lfloor n/2 \rfloor\}$, where each $I_i$ is a maximal interval. Suppose that $Q_{S_1}(x, y)$ gives a string which is different from $s$ and $s^*$. Then for each $i$, $Q_{I_1,i}(x, y)$ also gives a string which is different from $s$ and $s^*$.  

With Claim 9, Lemma 15 and Lemma 17, we can obtain that for $n \leq 5$, every string of length $n$ is uniquely reconstructible up to reversal.  

Corollary 18: (i) When $n = 6$ or 7, all the strings that are not uniquely reconstructible are listed in Example 12.  

(ii) When $n \leq 5$, each string of length $n$ can be uniquely reconstructed up to reversal.  

Proof: Suppose that $s \in \Sigma_n^2$ is a string that is not uniquely reconstructible up to reversal. Let $t \in \Sigma_n^2$ be such that $M(t) = M(s)$ and $t \neq s, s^*$. Then there exists a nonempty subset $S = \bigcup I_i$ of $D \cap \{1, \ldots, \lfloor n/2 \rfloor\}$, where each $I_i$ is a maximal interval, such that $P_t(x, y) = Q_{S_1}(x, y)$ or $Q_{S_2}(x, y)$. Notice that $S \neq \{1, \ldots, \lfloor n/2 \rfloor\}$ by Claim 9 (ii).  

(i) When $n = 6$ or 7, we have $|S| \leq 2$ since $|n/2| = 3$ and $S \neq \{1, 2, 3\}$. If $S = \{1, 2\}$ or $\{2, 3\}$, then $S$ is an interval. Now by Lemma 15, we cannot find a desired $k$ required in Claim 9 (ii). If $S = \{1, 3\} \cup \{\}$, i.e., $S$ is the union of two maximal intervals, by Lemma 17, $Q_{I_{1,1}}(x, y)$ gives a string different from $s$ and $s^*$. Then $(P_s(x, y))_2 = (P_s^*(x, y))_2$ by Lemma 15. Again, we cannot find a desired $k$ required in Claim 9 (ii) for $S$. Therefore $|S| = 1$ and it suffices to apply Theorem 11. So the first conclusion is proved.  

(ii) We analyze according to different values of $n$.
exists a string \(t\) such that \(\mathcal{M}(t) = \mathcal{M}(s)\) and \(t \neq s, s^*\), one can assume that \(S\) is an interval. Now we are ready to give the main result of this section, which is a generalization of Lemma 10.

**Theorem 19:** Let \(s \in \Sigma^*_2\) be a string and let \(P_s(x, y)\) be its generating polynomial given in Equation (1). Then there exists a string \(t\) such that \(t \neq s, s^*\) and \(\mathcal{M}(t) = \mathcal{M}(s)\) if and only if there exists an (nonempty) interval \(S = \{i_1, \ldots, i_2\}\) such that the following two conditions hold:

(i) \(i_1 - 1, i_2 + 1 \notin D \cap \{1, \ldots, [n/2]\}\).

(ii) There exists some \(k \in \{1, \ldots, [n/2]\} \setminus S\) such that \((P_s(x, y))_k \neq (P_s(x, y))_{k^*}\), i.e., \(\text{wt}_H(s^k) \neq \text{wt}_H(s^k_{n-k+1})\).

**Proof:** We first prove the \(\Rightarrow\) direction. By Lemma 17, we can choose a \(t\) whose generating polynomial corresponds to an interval. Then (i) follows from Lemma 15 and (ii) follows from Claim 9.

Now we prove the \(\Leftarrow\) direction. Condition (i) and Lemma 15 ensure that \(Q_{S, 1}(x, y)\) does give a string. Let \(t\) be the string with generating polynomial \(Q_{S, 1}(x, y)\). Claim 9 and Condition (ii) ensure \(t \neq s, s^*\). \(\square\)

IV. **Strings That Can Be Uniquely Reconstructed up to Reversal**

In the previous section we obtained that for any \(n \geq 6\), there always exists a string of length \(n\) that cannot be uniquely reconstructed up to reversal. Two natural questions arise:

1) For a given \(n\), how many strings of length \(n\) are there which can be uniquely reconstructed up to reversal?

2) Is there a unified method to construct all the uniquely reconstructible strings?

This section is devoted to solving these questions. Let \(U(n) \subseteq \Sigma^*_2\) be the set of all \(n\)-length strings that can be uniquely reconstructed up to reversal.

For any \(s \in \Sigma^*_2\), let \(D\) be as defined at the beginning of Section III. Let \(0 = j_0 < j_1 < \cdots < j_k \leq [n/2]\), where \(\ell \geq 0\), be such that \(\{j_i\}_{i=0}^k = \{1, \ldots, [n/2]\} \setminus D\). Then by the definition of \(D\), we have \((P_s(x, y))_j = (P_s(x, y))_{j^*}\) i.e., \(\text{wt}_H(s^j) = \text{wt}_H(s^{j^*})\), \(\text{wt}_H(s^j) = \text{wt}_H(s^j_{n-j+1})\) if and only if \(j \in \{j_i, n - j_i : 0 \leq i \leq \ell\}\). Then each string \(s\) can be uniquely represented as

\[
s = r^{(i)} \ldots r^{(\ell)} r^{(\ell+1)} (t^{(i)})^* \ldots (t^{(1)})^*,
\]

where \(r^{(i)} = s^{j_i}_{j_i+1}, t^{(i)} = (s^*)^{j_i}_{j_i+1}\) for all \(1 \leq i \leq \ell\) and \(r^{(\ell+1)} = s^{j_{\ell+1}}\). Here and in the rest of this paper, we denote \(j_{\ell+1} \triangleq n - j_{\ell}\). See Figure 1 for an illustration.

**Example 20:**

1) Let \(n = 6\) and \(s = 010111\). Then \(\ell = 0, j_0 = 0\) and \(j_1 = 6\). We have \(r^{(1)} = s\).

2) Let \(n = 7\) and \(s = 1011010\). Then \(\ell = 1, j_1 = 2\) and \(j_2 = 5\). We have \(r^{(1)} = 10, t^{(1)} = 01\) and \(r^{(2)} = 110\).

The relationship between \(\{j_i, \ldots, j_{\ell}\}\) and \(D\) implies the following reformulation of Theorem 19.

**Theorem 21:** Let \(s \in \Sigma^*_2\) and let \(j_0, \ldots, j_{\ell}, j_{\ell+1}\) be defined as above. Then \(s\) cannot be uniquely reconstructed up to reversal if and only if there exist two different integers \(m, m' \in \{1, \ldots, \ell + 1\}\) such that \(j_m - j_{m-1} \geq 2\) and \(j_{m'} - j_{m'-1} \geq 2\). In particular, we have \(\ell \geq 1\).

**Proof:** We first prove the \(\Rightarrow\) direction. Suppose that \(s\) cannot be uniquely reconstructed up to reversal. Then there is an interval \(S = \{i_1, \ldots, i_2\}\) of \([1, \ldots, [n/2]\}\) that satisfies (i) and (ii) of Theorem 19. If \(i_2 < [n/2]\), let \(m\) be such that \(j_m = i_2 + 1\). Then we have \(j_{m-1} = i_1 - 1\). If \(i_2 = [n/2]\), then \(j_{\ell-1} = i_1 - 1\). In this case, let \(m = \ell + 1\). For either case, it holds that \(j_m - j_{m-1} \geq 2\) by (ii) of Theorem 19, there exists \(k \in \{1, \ldots, [n/2]\}\) such that \(\text{wt}_H(s^k) = \text{wt}_H(s^k_{n-k+1})\). Let \(m'\) be such that \(j_{m'} - j_{m'-1} < k < j_{m'}\). Then \(j_{\ell'} - j_{\ell'-1} > 2\). Since \(k \notin S\), we have \(m \neq m'\).

Now we prove the \(\Leftarrow\) direction. If \(m \leq \ell\), we set \(S = \{j_1, \ldots, j_{m-1}\}\). If \(m = \ell + 1\), we set \(S = \{j_1 + 1, \ldots, [n/2]\}\). Since \(j_m - j_{m-1} \geq 2\), we have \(S \neq \emptyset\). Then by the definition of \(j_0, j_1, \ldots, j_{\ell+1}\), condition (i) of Theorem 19 is satisfied. Since \(j_m - j_{m-1} \geq 2\), we can choose a \(k\) with \(j_{m'-1} < k < j_{m'}\). In addition, we have \(k \notin S\) because \(m \neq m'\). Then condition (ii) of Theorem 19 is satisfied. Therefore, the string \(s\) cannot be uniquely reconstructed up to reversal. \(\square\)

Recall that \(\text{wt}_H(s^j) = \text{wt}_H((s^*)^j)\) if and only if \(j \in \{j_i, n - j_i : 0 \leq i \leq \ell\}\). Therefore, the \(2\ell + 1\) strings \(r^{(i)}\) and \(t^{(i)}\) (\(1 \leq i \leq \ell\)) must satisfy

\[
\text{wt}_H(r^{(i)}) = \text{wt}_H(t^{(i)}),
\]

\[
\text{wt}_H(r^{(i)})^{k}_{1} \neq \text{wt}_H(t^{(i)})^{k}_{1}
\]

for all \(1 \leq k < j_i - j_{i-1}\),

\[
\text{wt}_H(r^{(\ell+1)})^{k}_{1} \neq \text{wt}_H(r^{(\ell+1)})^{j_{\ell+1}-j_{\ell}-k}_{j_{\ell+1}-j_{\ell}-k+1}
\]

for all \(1 \leq k < j_{\ell+1} - j_{\ell}\). (11)

Since Equation (10) and Equation (11) hold for arbitrary strings, we obtain the following proposition.

**Proposition 22:** For any \(n \geq 1\), it holds that

\[
\Sigma^{\text{n}}_2 = \left\{r^{(1)} \ldots r^{(\ell)} r^{(\ell+1)} (t^{(i)})^{*} \ldots (t^{(1)})^{*} : \ell \geq 0; j_0 = 0; 1 \leq j_1 < \cdots < j_{\ell} \leq \left\lfloor \frac{n}{2} \right\rfloor \right\},
\]
when \( Ye \) and Elishco: Reconstruction of a Single String from a Part of its Composition Multiset 3929

In Equation (12), we allow \( wt_H(n) \leq n \) for all \( 1 \leq j < n \). For the \( \Rightarrow \) direction, we first assume \( s_1 = 0 \) and \( s_n = 1 \). Let \( I \subseteq \{2, \ldots, n/2\} \) such that \( s_i \neq s_{n-i-1} \) if and only if \( i \in I \cup (n + 1 - I) \cup \{1, n\} \). Since any \( i \notin I \cup (n + 1 - I) \cup \{1, n\} \) does not affect \( wt_H(s_i) - wt_H(s_{n-j-1}) \), together with the assumptions that \( s_1 = 0, s_n = 1 \) and that \( wt_H(s_i) \neq wt_H(s_{n-j-1}) \) for all \( 1 \leq j < n \), we conclude that \( wt_H(s_i) < wt_H(s_{n-j-1}) \) for all \( 1 \leq j < n \). It follows that \( s_{n-j-1} \) is a Catalan-Bertrand string and hence \( s \in S_R(n) \).

The following corollary is a direct result of Lemma 24.

Corollary 25: For a given \( s \), let \( f(n) \) be the number of strings \( s \) of length \( n \) such that \( wt_H(s) \neq wt_H(s_{n-j-1}) \) for all \( 1 \leq j < n \). Then

\[
f(n) = \begin{cases} 2 |S_R(n)| & \text{if } n \geq 2, \\ 2 & \text{if } n = 1. \end{cases}
\]

The value of \( |S_R(n)| \) was given in [23]:

\[
|S_R(n)| = \begin{cases} \sum_{i=0}^{n} \frac{1}{2^i} & \text{if } n \text{ is even,} \\ \sum_{i=0}^{\lfloor n/2 \rfloor} \frac{1}{2^i} & \text{if } n \text{ is odd.} \end{cases}
\]

Here we define \( S_R(0) \triangleq \{\epsilon\} \) and hence \( |S_R(0)| = 1 \). Therefore,

\[
\lim_{n \to \infty} \log_2 \left( |S_R(n) \cup S_R(n)^*| / n \right) = 1,
\]

where \( S_R(n)^* \triangleq \{s : s^* \in S_R(n)\} \).

The following corollary is an immediate result of Proposition 23 (by letting \( \ell = 0 \)) and Lemma 24. We note that it also follows from [23, Lemma 11].

Corollary 26: Let \( s \in \Sigma_2^n \) be a string such that \( wt_H(s) \neq wt_H(s_{n-j-1}) \) for all \( 1 \leq j < n \). Then \( s \) is uniquely reconstructible up to reversal. In other words, any string in \( S_R(n) \cup S_R(n)^* \) can be uniquely reconstructed up to reversal.

B. General \( \ell \)

In this subsection, we consider the case where \( \ell \) is arbitrary. First, since \( r(\ell+1) \) satisfies the third condition in Equation (11), we have \( r(\ell+1) \in S_R(j+1 - j) \cup S_R(j-1 - j) \). Our next aim is to explicitly construct \( r(i) \) and \( t(i) \) so that they satisfy Equation (11), for each \( 1 \leq i \leq \ell \). To this end, let \( u = r(i)(t(i))^* \). Then Equation (11) is equivalent to

\[
wt_H\left( u^m \right) = wt_H\left( u^m_{\ell+1} \right),
\]

where \( m = 2(j_i - j_{i-1}) \).

Now the question boils down to explicitly constructing strings that satisfy Equation (15). To move further, we first

\[
wt_H\left( u_{m-j}^{n} \right) = wt_H\left( u_{m-k+1}^n \right) \quad \text{for all } 1 \leq k < \frac{m}{2},
\]
introduce the well-known Dyck strings [27, Section 1.4]. A string $s \in \Sigma_2^2$ is called a Dyck string if $wt_H(s) = h$ and $wt_H(s_i) \geq \left\lfloor \frac{2}{7} h \right\rfloor$ for all $1 \leq i \leq 2h - 1$. The number of Dyck strings of length $2h$ is given by the Catalan number [27, Theorem 1.4.1]:

$$C_h \triangleq \frac{1}{h + 1} \binom{2h}{h}.$$  

For even $m \geq 4$, define

$$A(m) \triangleq \{ u \in \Sigma_m^n : u_1 = 0, u_{m+1} = 0, u_1 = 1; \exists I \subseteq \{2, \ldots, m/2 - 1\} \text{ such that if}$$

$$1 < i < m/2, \text{ then } u_i \neq u_{m-i+1} \text{ if and}$$

$$\text{only if } i \in I; \overline{u}_I \text{ is a Dyck string} \},$$

where $\overline{u} \triangleq (1 - s_1)(1 - s_2) \cdots (1 - s_n)$ for any string $s \in \Sigma_2^n$. We allow $I = \emptyset$ in the definition of $A(m)$. Note that since we require $u_I$ to be a Dyck string, $|I|$ is necessarily even.

Remark 27: Suppose $u \in A(m)$ and let $I$ be such that $\overline{u}_I$ is a Dyck string. Then $wt_H(u_{j\ell (1, m/2)} = \lfloor |I| + 2 \cdot 2^h \rfloor$. So $u \notin S_R(m)$ and thus $A(m) \cap S_R(m) = \emptyset$.

By definition, we have

$$|A(m)| = \sum_{i=0}^{m-1} \binom{\frac{m-2}{2}}{\frac{i}{2}} 2^{h-2}\frac{1}{2^i} \left(\frac{2^i}{i + 1}\right)$$  \hspace{1cm} (16)

If $m = 2$, we define $D(m) \triangleq \{00, 11\}$. If $m \geq 4$ is even, we define $D(m) \triangleq \{ u : u \in A(m) \text{ or } u^* \in A(m) \}$.

Lemma 28: Let $m \geq 2$ be an even integer. Then a string $u \in \Sigma_2^n$ satisfies Equation (15) if and only if $u \in D(m)$.

Proof: The $\Rightarrow$ direction follows immediately from symmetry around $m/2$ in the construction. Thus, we only need to prove the $\Leftarrow$ direction. If $m = 2$, the conclusion is trivial so we may assume $m \geq 4$.

We first notice that since $m \geq 4$, the second row of Equation (15) has a meaning. From the second row, we deduce $u_1 \neq u_m$. Assume that $u_1 = 0$ and $u_m = 1$ (the case $u_1 = 1$ and $u_m = 0$ is similar). Notice that the second row of Equation (15) implies that $wt_H(u_1^2) < wt_H(u_{m-k+1})$ for all $1 \leq k \leq m/2 - 1$. Thus, the equality in the first row of Equation (15), $wt_H(u_{m/2}^2) = wt_H(u_{m/2+1})$, implies that $u_{m/2} = 1$ and $u_{m/2+1} = 0$.

Let $I = \{ i : 1 < i < m/2, u_i \neq u_{m-i+1} \}$. If $I = \emptyset$, we are done. Otherwise, the first row of Equation (15) implies that $|I|$ is even and $wt_H(u_I) = |I|/2$. This, together with the second row of Equation (15) implies that $\overline{u}_I$ is a Dyck string. \( \square \)

As a result of Proposition 22 and Lemma 28, we obtain the following characterization of the set of all binary sequences.

Corollary 29: For all $n \geq 1$, we have

$$\sum_{2}^{n} = \left\{ r^{(1)} \cdots r^{(\ell)} t^{(\ell+1)} (t^{(\ell+1)})^* \cdots (t^{(1)})^* : \ell \geq 0; j_0 = 0 \right\}$$

$$1 \leq j_1 \cdots j_\ell \leq \left\lfloor \frac{n}{2} \right\rfloor; j_{\ell+1} = n - j_\ell;$$

$$\begin{align*}
\text{r}^{(\ell+1)} &\in S_R(j_{\ell+1} - j_\ell) \cup S_R(j_\ell - j_{\ell+1}); \\
r^{(i)} t^{(i)} &\in \sum_{2}^{j_i-j_{i-1}} \text{ for all } 1 \leq i \leq \ell; \\
r^{(i)} (t^{(i)})^* &\in D(2j_i - 2j_{i-1}) \text{ for all } 1 \leq i \leq \ell.
\end{align*}$$  \hspace{1cm} (17)

Example 30: Let $n = 4$. Then $\ell = 0$, 1 or 2.

- When $\ell = 2$, we have $j_1 = 1$ and $j_3 = j_2 = 2$. So in this case the sequences have the form $r(1)^{r(2)} r(3)^{t(3)} (t(3))^* (t(1))^*$, where $r(3) \in S_R(0) \cup S_R(0)^*$ and $r(1)^*(t(1))^* r(2)^* (t(2))^* \in D(2)$. Since $D(2) = \{00, 11\}$ and $S_R(0) = \{ e \}$, in this case we obtain four sequences: 0000, 0110, 1001, 1111.

- When $\ell = 1$, we have $j_1 = 1$ or $j_2 = 2$. If $j_1 = 1$, then $j_2 = 3$. In this case the sequences have the form $r(1)^* r(2)^* (t(1))^*$, where $r(2) \in S_R(2) \cup S_R(2)^*$ and $r(1)^* (t(1))^* \in D(2)$. Since $S_R(2) = \{01\}$, in this case we obtain four sequences: 0010, 0100, 1011, 1101.

- When $\ell = 0$, we have $j_1 = 4$. In this case the sequences have the form $r(1)^*$, where $r(1) \in S_R(4) \cup S_R(4)^*$. Since $S_R(4) = \{0001, 0111, 0011\}$, in this case we obtain six sequences: 0001, 0111, 0011, 1100, 1110, 1000.

Now we have obtained all length-4 strings.

Combining Proposition 23 and Lemma 28, we obtain the main result of this subsection.

Theorem 31: For all $n \geq 1$, we have

$$\mathcal{U}(n) = \left\{ r^{(1)} \cdots r^{(\ell)} t^{(\ell+1)} (t^{(\ell+1)})^* \cdots (t^{(1)})^* : \ell \geq 0;$$

$$j_0 = 0; 1 \leq j_1 \cdots j_\ell \leq \left\lfloor \frac{n}{2} \right\rfloor; j_{\ell+1} = n - j_\ell;$$

$$r^{(\ell+1)} \in S_R(j_{\ell+1} - j_\ell) \cup S_R(j_\ell - j_{\ell+1});$$

$$r^{(i)} t^{(i)} \in \sum_{2}^{j_i-j_{i-1}} \text{ for all } 1 \leq i \leq \ell;$$

$$r^{(i)} (t^{(i)})^* \in D(2j_i - 2j_{i-1}) \text{ for all } 1 \leq i \leq \ell;$$

there is at most one $i$ such that $j_i - j_{i-1} \geq 2$ \right\}.$$  \hspace{1cm} (18)

It is possible to provide a formula for $|\mathcal{U}(n)|$. For $n \geq 1$, define

$$\mathcal{P}(n) \triangleq \left\{ j = (j_0, j_1, \ldots, j_\ell + 1) :$$

$$\ell \geq 0; j_0 = 0;$$

$$1 \leq j_1 < \cdots < j_\ell \leq \left\lfloor \frac{n}{2} \right\rfloor;$$

$$j_{\ell+1} = n - j_\ell;$$

there is at most one $i$ such that $j_i - j_{i-1} \geq 2$$\right\}.$$  \hspace{1cm} (19)

Then

$$|\mathcal{U}(n)| = \sum_{\ell+1} \prod_{j \in \mathcal{P}(n)} g(j_i - j_{i-1}), \hspace{1cm} (17)$$

where $\ell$ is determined by $j$ and

$$g(j_i - j_{i-1}) =$$

$$\begin{cases}
1, & \text{if } j_i = j_{i-1} \\
2, & \text{if } j_i - j_{i-1} = 1 \\
2 | S_R(n - 2j_i) \rangle, & \text{if } i = \ell + 1 \text{ and } n - 2j_i \geq 2 \\
2 | A(2j_i - 2j_{i-1}) \rangle, & \text{if } j_i - j_{i-1} \geq 2 \text{ and } i \neq \ell + 1. \end{cases} \hspace{1cm} (18)$$

We give two examples below.

Example 32: All needed values of $|S_R(m)|$ and $|A(m)|$ are listed in Table I.
The goal of this subsection is to completely characterize the sets $E(s)$ and $E_n$ for any $s$ and $n$. To that end, we build upon previous ideas. Specifically, we use similar ideas to those that appeared in Theorem 19 and Theorem 21. Before diving into the calculations, we give some simple examples.

Example 34: 1) Let $n = 1$. Then $E(0) = \{0\}$, $E(1) = \{1\}$ and hence $E_1 = 1$.

2) Let $n = 2$. Then $E(00) = \{00\}$, $E(11) = \{11\}$, and $E(01) = \{01, 10\}$. Therefore, $E_2 = 2$.

3) From Corollary 18, we conclude that for $3 \leq n \leq 5$, $|E(s)| = 1$ or 2 for any $s \in \Sigma_n$ and that $E_n = 2$.

In the rest of this section, we always assume that $n \geq 2$, unless otherwise stated.

Recall that for any $s \in \Sigma_n$, we can always find $0 = j_0 < j_1 < \cdots < j_t \leq \lfloor n/2 \rfloor$, where $t \geq 0$, such that $(P_s(x,y))_{j_i} = (P_s^*(x,y))_{j_i}$ if and only if $j_i \in \{j_1, n-j_1 : 1 \leq i \leq t\}$.

In other words, $wt_H(s^*_1) = wt_H((s^*)_1^t)$ if and only if $j_i \in \{j_1, n-j_1 : 1 \leq i \leq t\}$.

Given a string $s$, we define $I_s = \{i : 1 \leq i \leq \ell + 1, j_i - j_{i-1} + 1 \geq 2\}$.

Then Theorem 21 asserts that $s$ cannot be uniquely reconstructed up to reversal if and only if $|I_s| \geq 2$. In Section III, we showed that $t \in E(s)$ if only if $P_t(x,y) = Q_{S,1}(x,y)$ or $P_t(x,y) = Q_{S,2}(x,y)$ for some carefully chosen subset $S$ of $D \setminus \{1, \ldots, \lfloor n/2 \rfloor\} \setminus \{j_1, \ldots, j_t\}$; we have the following natural definition.

Definition 35: Let $A \subseteq I_s$ be a set. We define a string $t_A(s)$ in the following way.

1) If $A = \emptyset$, let $t_A(s) = s$.

2) If $A \neq \emptyset$, let $t_A(s)$ be the string such that $(P_{t_A(s)}(x,y))_j = (P_s^*(x,y))_j$, $(P_{t_A(s)}(x,y))_{n-j} = (P_s^*(x,y))_{n-j}$, for all $i \in A$ and all $j_{i-1} < j < j_i$; and $(P_{t_A(s)}(x,y))_j = (P_s(x,y))_j$ for all other $j$’s.

Define $\text{Swap}(s, A)$ as

$$\text{Swap}(s, A) = \{t_A(s), t_A(s)^*\}.$$ 

In words, $\text{Swap}(s, A)$ is the set of the strings obtained by swapping between $(P_s^*(x,y))_j$ and $(P_s^*(x,y))_{n-j}$, and between $(P_s(x,y))_{n-j}$ and $(P_s^*(x,y))_{n-j}$, for all $j_{i-1} < j < j_i$ and for all $i \in A$, while keeping $(P_s(x,y))_j$ and $(P_s^*(x,y))_j$ intact for all other $j$’s. If $A = \emptyset$, we define $\text{Swap}(s, A) = \{s, s^*\}$.

Remark 36: In fact, let $A$ be as in Definition 35 and $S$ be such that $P_{t_A(s)}(x,y) = Q_{S,1}(x,y)$, then

$$S = \{j : \exists i \in A, \text{ such that } j_{i-1} < j < j_i \cap \{1, \ldots, \lfloor n/2 \rfloor\}.$$ 

On the other hand, suppose that $S = \cup I_t$ is a nonempty subset of $D \cap \{1, \ldots, \lfloor n/2 \rfloor\}$ such that $Q_{S,1}$ gives a string, where $I_t = \{k_{t1}, k_{t1} + 1, \ldots, k_{t2}\}$ is a maximal interval, for each $t$. By Claim 16 and Lemma 15, we conclude that for each $t$, there exists an $i \in \{1, \ldots, \ell + 1\}$ such that $k_{t1} - 1 = j_{i-1}$. Then $A$ is the set consisting of all such $i$’s. Therefore, $t \sim s$ if and only if $t \in \text{Swap}(s, A)$ for some $A \subseteq I_s$.

Notice that $1 \leq |\text{Swap}(s, A)| \leq 2$. Indeed, if $I_s = \emptyset$, then $s \neq s^*$ and $t_A(s) \neq t_A(s)^*$, implying that $|\text{Swap}(s, A)| = 2$. 

| TABLE I |
| --- |
| Values of $|S_R(m)|$ and $|A(m)|$ in Example 32 |
| $m$ | $|S_R(m)|$ | $|A(m)|$ |
| 2 | 1 | |
| 3 | 2 | |
| 4 | 3 | 1 |
| 5 | 6 | |
| 6 | 10 | 2 |
| 7 | 20 | |

| TABLE II |
| --- |
| Values of $\prod_{i=1}^{t+1} g(j_i - j_{i-1})$ in Example 32 |
| $j$ | $\prod_{i=1}^{t+1} g(j_i - j_{i-1})$ |
| (0, 6) | 20 |
| (0, 1, 5) | 12 |
| (0, 3, 3) | 4 |
| (0, 1, 2, 4) | 8 |
| (0, 1, 2, 3, 3) | 4 |
| (0, 1, 2, 3, 3) | 8 |

1) Let $n = 6$. Then

$$\mathcal{P}(6) = \{(0, 6), (0, 1, 5), (0, 3, 3), (0, 1, 2, 4), (0, 1, 3, 3), (0, 2, 3, 3), (0, 1, 2, 3, 3)\}.$$ 

According to Equation (18), we obtain Table II. Then by Equation (17), we have $|U(6)| = 20 + 12 + 4 + 8 + 4 + 4 + 8 = 60$.

2) Let $n = 7$. Then

$$\mathcal{P}(7) = \{(0, 7), (0, 1, 6), (0, 3, 4), (0, 1, 2, 5), (0, 1, 3, 4), (0, 2, 3, 4), (0, 1, 2, 3, 4)\}.$$ 

By a similar process as in 1), we can get $|U(7)| = 120$. The results in Example 32 are consistent with Example 12 and Corollary 18.

Up to this point, we studied the set of strings that can be uniquely reconstructed (up to reversals) from their prefix-suffix composition multiset. However, we did not study how many sequences there are that share the same prefix-suffix composition multiset. This is the focus of the next section.

V. MUTUALLY EQUIVALENT STRINGS

In this section we focus our attention on the study of sets of strings with a common prefix-suffix composition multiset. To clearly state the goals of this subsection, we define the following equivalence relation on the set of strings.

Definition 33: Let $s, t \in \Sigma_n$ be two strings of length $n$. We say that $s$ and $t$ are equivalent, denoted $s \sim t$, if $M(s) = M(t)$, i.e., if $s$ and $t$ have the same prefix-suffix composition multiset.

It is clear that the relation “$\sim$” is indeed an equivalence relation. For a string $s \in \Sigma_n$ of length $n$, we define

$$E(s) = \{t \in \Sigma_n : t \sim s\}.$$ 

to be the set of all strings that are equivalent to $s$, and

$$E_n = \max \{|E(s)| : s \in \Sigma_n\}.$$ 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Example 37: Let us consider the following sequence
\[ s = 110101011101010111 \]
of length 18. We have \( s^* = 110101011101010111 \). The corresponding polynomials are
\[
P_s(x, y) = 1 + x + x^2 + (x^2 y) + x^3 y + (x^3 y^2) + x^4 y^2
\]
\[
+ (x^4 y^3) + x^5 y^3 + x^5 y^2 + x^7 y^2 + (x^7 y^4)
\]
\[
+ x^8 y^4 + (x^8 y^5) + x^9 y^5 + (x^9 y^6)
\]
\[
+ x^{10} y^6 + x^{11} y^6 + x^{12} y^6,
\]
\[
P_s'(x, y) = 1 + x + x^2 + (x^3) + x^3 y + (x^4 y^2)
\]
\[
+ (x^5 y^2) + x^5 y^3 + x^6 y^3 + x^7 y^3 + (x^8 y^3)
\]
\[
+ x^8 y^4 + (x^9 y^4) + x^9 y^5 + (x^{10} y^5)
\]
\[
+ x^{10} y^6 + x^{11} y^6 + x^{12} y^6,
\]

where \((\cdot)\) denotes a difference in the total degree between \(P_s(x, y)\) and \(P_s'(x, y)\). Therefore, we obtain \([18/2] = 9\) and \(j_1 = 1, j_2 = 2, j_3 = 4, j_4 = 6, j_5 = 8, j_6 = 9\) with the additional \(j_{t+1} = j_7 = 18 - j_6 = 9\). This implies that \(I_s = \{3, 4, 5\}\) since \(j_3 - j_2 = j_4 - j_3, j_5 - j_4 = 2\). Let us take \(A = \{3, 4\}\). The generating polynomial of \(t_A(s)\) is
\[
P_{t_A(s)}(x, y) = 1 + x + x^2 + (x^3) + x^3 y + (x^4 y^2)
\]
\[
+ x^4 y^2 + x^5 y^2 + x^6 y^2 + x^7 y^2 + x^8 y^2
\]
\[
+ x^8 y^3 + (x^9 y^3) + x^9 y^4 + (x^{10} y^4)
\]
\[
+ x^{10} y^6 + x^{11} y^6 + x^{12} y^6.
\]

Thus, the corresponding string is \(t_A(s) = 110101011101010111\). So \(\text{Swap}(s, A) = \{t_A(s), t_A(s)^*\} = \{110101011101010111, 110101011101010111\}\).

By Remark 36, we know that \(t \sim s\) if and only if \(t \in \text{Swap}(s, A)\) for some \(A \subseteq I_s\). Therefore, to fully understand \(E(s)\), we need to study the relationship between \(\text{Swap}(s, A)\) and \(\text{Swap}(s, B)\) for distinct \(A, B \subseteq I_s\).

Lemma 38: Let \(A, B\) be two subsets of \(I_s\) such that \(A \neq B\).

(i) If \(A \cup B \neq I_s\), then \(\text{Swap}(s, A) \cap \text{Swap}(s, B) = \emptyset\).

(ii) If \(A \subseteq B\), then \(\text{Swap}(s, A) \cap \text{Swap}(s, B) = \emptyset\).

(iii) If \(A \cap B = I_s\), then \(\text{Swap}(s, A) \cap \text{Swap}(s, B) = \emptyset\).

Proof: (i) Since \(A \cup B \neq I_s\), there exists \(i_0 \in I_s \setminus \{A \cup B\}\) which implies that there is \(j_{i_0-1} < k < j_{i_0}\) and \((P_s(x, y))\) for all \(j_{i_0-1} < k < j_{i_0}\) such that \(i \in A \cup B\) where \(\Delta\) is the symmetric difference. This is possible only if \(A \cup B = \emptyset\) which contradicts the assumption. A similar contradiction arises if we assume \((P_s(x, y))\) for all \(j_{i_0-1} < k < j_{i_0}\). Since \(A \neq B\), we have \(A \setminus B \neq \emptyset\) or \(B \setminus A \neq \emptyset\).

Assume that \(A \cap B \neq \emptyset\) and choose \(i_2 \in A \cap B\). We have \((P_s(x, y))\) for all \(j_{i_2-1} < k' < j_{i_2}\). Let \(t_1 \in \text{Swap}(s, A)\) and \(t_2 \in \text{Swap}(s, B)\) be arbitrary. For \(t_1\), we have either
\[
(P_t_1(x, y)) = (P_s(x, y))\] or \((P_t_1(x, y)) = (P_s(x, y))\). On the other hand, for \(t_2\), we have either
\[
(P_t_2(x, y)) = (P_s(x, y))\] or \((P_t_2(x, y)) = (P_s(x, y))\). This implies that \(t_1 \neq t_2\) and since \(t_1, t_2\) were arbitrary, \(\text{Swap}(s, A) \cap \text{Swap}(s, B) = \emptyset\). A similar analysis holds when \(B \setminus A \neq \emptyset\).

(iii) By definition, it follows that \(t_s(s) = t_{g_1(s)}^*\) and \(t_A(s) = t_{g_2(s)}\). Therefore, \(\text{Swap}(s, A) = \text{Swap}(s, B)\). Let us continue Example 37.

Example 39: Consider the same string \(s = 110101011101010111\) as in Example 37, with the same polynomials \(P_s(x, y)\) and \(P_s'(x, y)\). Let us consider the case \(B = \{4, 5\}\). The generating polynomial of \(t_B(s)\) is
\[
P_{t_B(s)}(x, y) = 1 + x + x^2 + x^2 y + x^3 y + (x^4 y^2)
\]
\[
+ (x^5 y^2) + x^5 y^3 + x^6 y^3 + x^7 y^3 + (x^8 y^3)
\]
\[
+ x^8 y^4 + (x^9 y^4) + x^9 y^5 + (x^{10} y^5)
\]
\[
+ x^{10} y^6 + x^{11} y^6 + x^{12} y^6.
\]

The corresponding string is \(t_B(s) = 1101101101110100111\). Notice that taking \(B^* = \{5\}\) yields \(\text{Swap}(s, B^*) = \text{Swap}(s, A)\) (where \(A = \{3, 4\}\) as in Example 37) and taking \(A^* = \{4\}\) in Example 37 yields \(\text{Swap}(s, A) = \text{Swap}(s, A^*)\).

Using the previous lemma, we can characterize the set of strings that are equivalent to \(s\), i.e., the set \(E(s)\). To that end, we define the following.

Definition 40: Let \(S\) be a nonempty set and let \(A\) be a set of subsets of \(S\), i.e., \(A \subseteq 2^S\) where \(2^S\) is the power set of \(S\). We say that \(A\) is open under complements if \(X \in A\) implies \(X^c \subseteq S \setminus X \notin A\). If \(A \subseteq 2^S\) we denote by \(A^c\) an arbitrary set obtained by making \(A\) an open under complements set by arbitrarily eliminating \(X\) or \(X^c\) if both \(X, X^c \in A\), i.e., if \(X \in A\) and \(S \setminus X \in A\) we choose one of them arbitrarily and remove it from \(A\).

For a string \(s\), if \(I_s = \emptyset\), let \(F_s = \{\emptyset\}\). If \(I_s \neq \emptyset\), let \(F_s = A^c,\) where \(A = 2 I_s^c,\) meaning that \(F_s\) is open under complement, obtained from the power set of \(I_s\). Of course \(F_s\) is not unique when \(I_s \neq \emptyset\). For example, if \(I_s = \{1, 4\}\), then the power set is \(2 I_s^c = \{\emptyset, \{1\}, \{4\}, \{1, 4\}\}\). We can take \(F_s = \{\emptyset, \{1\}\}, F_s = \{\emptyset, \{4\}\}, F_s = \{\{1\}, \{4\}\},\) or \(F_s = \{\{1, 4\}\}\). However, if \(I_s \neq \emptyset\), it is easy to see that \(|F_s| = 2^{|I_s| - 1}\), regardless of the choice of \(F_s\).

Corollary 41: For any \(F_s\), we have \(E(s) = \bigcup \text{Swap}(s, A)\) and \(|E(s)| = 2^{|I_s|}\). In particular, if \(I_s = \emptyset\), we have \(E(s) = \{s\}\) and \(|E(s)| = 1\).
Proof: If \( I_\emptyset = \emptyset \), we have \( s = s^* \) and the conclusion follows trivially from Theorem 21 and Definition 35. Next, we will prove the conclusion for \( I_s \neq \emptyset \).

For each string \( t \), we have \( t \sim s \) if and only if \( t \in \text{Swap}(s, A) \) for some \( A \subseteq I_s \), as explained in Remark 36. Therefore, from the construction of \( F_s \), we have that for \( A, B \in F_s \), either \( A \cup B \neq I_s \) or \( A \neq I_s \ \setminus B \). In either case, Lemma 38 asserts that \( \text{Swap}(s, A) \cap \text{Swap}(s, B) = \emptyset \).

Therefore, \(|E(s)| = 2|F_s| = 2|I_s|\). \(\square\)

We present some examples to explain Corollary 41.

Example 42: 1) Let \( n = 6 \) and \( s = 101010 \). From Example 20, we already know that \( \ell = 1, j_1 = 2 \), and \( j_\ell + 1 = j_2 = 4 \). Therefore, \( I_s = \{1, 2\} \). From Corollary 41, we have \(|E(s)| = 2|I_s| = 2\).

2) Let \( n = 6 \) and \( s = 010111 \). From Example 20, we know that \( \ell = 0 \) so \( I_s = \{1\} \). By Corollary 41, we have \(|E(s)| = 2|I_s| = 2\).

3) Let \( n = 7 \) and \( s = 1010110 \). From Example 20, we know that \( \ell = 1 \) and \( j_1 = 2 \), so \( I_s = \{1, 2\} \). By definition, \( \text{Swap}(s, \emptyset) = \{s, s^*\} \) and \( \text{Swap}(s, \{1\}) = \{0111001, 1001111\} \). Then by Corollary 41, we have \(|E(s)| = 2|I_s| = 4\).

We see that the results in Example 42 are consistent with those in Example 12.

Our final goal for this section is to estimate the maximum size of \( E(s) \), i.e., to calculate \( E_n \). From Corollary 41, we see that in order to determine \( E_n \), we should determine the maximum value of \(|I_s|\).

Theorem 43: For \( n \geq 2 \), we have \( E_n = 2^{2n+2} \).

Proof: The proof is a case-by-case analysis.

1) Assume \( n \) is even.

If \( j_\ell = \frac{n}{2} \): We have \( j_\ell + 1 - j_\ell = n - 2j_\ell = 0 \). Then \(|I_s| \leq \left\lfloor \frac{n}{2} \right\rfloor = \frac{n}{4} \).

If \( j_\ell = \frac{n}{2} - 1 \): We have \( j_\ell + 1 - j_\ell = 2 \). Then \(|I_s| \leq \left\lfloor \frac{n-1}{2} \right\rfloor + 1 = \frac{n+2}{4} \). This bound can be achieved by taking \( j_i - j_{i-1} = 2 \) for all \( 1 \leq i < \ell \) with possibly one exception in which \( j_i - j_{i-1} = 1 \).

If \( j_\ell \leq \frac{n}{2} - 2 \): We have \( j_\ell + 1 - j_\ell > 0 \). Thus, \(|I_s| \leq \left\lfloor \frac{n-2}{2} \right\rfloor + 1 = \frac{n+2}{4} \).

Thus, if \( n \) is even, then \( E_n = \frac{n+2}{4} \).

2) Assume \( n \) is odd.

If \( j_\ell = \left\lfloor \frac{n}{2} \right\rfloor \): We have \( j_\ell + 1 - j_\ell = 1 \). Then \(|I_s| \leq \left\lfloor \frac{n-1}{2} \right\rfloor = \frac{n-1}{4} \).

If \( j_\ell \leq \frac{n}{2} - 1 \): We have \( j_\ell + 1 - j_\ell \geq 3 \). Thus, \(|I_s| \leq \left\lfloor \frac{n+2}{2} \right\rfloor + 1 = \frac{n+4}{4} \). This bound can be achieved by taking \( j_i - j_{i-1} = 1 \) for all \( 1 \leq i \leq \ell \) with possibly one exception in which \( j_i - j_{i-1} = 2 \).

Therefore, if \( n \) is odd, then \( E_n = \frac{n+2}{4} \).

Overall, we obtain that \( E_n = \frac{n+2}{4} \) which is the wanted result. \( \square \)

Remark 44: Notice that Theorem 43 asserts that \( E_n \geq 2^{n+1} \) for all \( n \geq 6 \). This means that there exists a string \( s \) of length \( n \) that cannot be uniquely reconstructed up to reversal for every \( n \geq 6 \).

Up to now we studied strings that can (or cannot) be determined up to reversal by their prefix-suffix composition multiset, which makes sense from a computational point of view. However, from the perspective of coding, one prefers constructing a large subset \( C \subseteq \Sigma_n^* \) such that each \( s \in C \) is uniquely determined by \( M(s) \). In other words, when knowing \( s \in C \), one can reconstruct \( s \) from \( M(s) \). In addition, an efficient method to reconstruct \( s \in C \) from \( M(s) \) is desired. This is the main focus of the next section.

VI. COMPOSITION CODES

In this section, we study strings that can be uniquely determined by their prefix-suffix composition multiset. To that end, we define a composition code, \( C \subseteq \Sigma_n^* \), to be a set of strings such that for \( s, t \in C \), \( M(s) \neq M(t) \) implies \( s = t \). That is to say, any \( s \in C \) is uniquely determined by \( M(s) \).

Let \( C_{\text{max}} \subseteq \Sigma_n^* \) be any composition code of length \( n \) of maximum size.

Theorem 45: \(|C_{\text{max}}| \geq \frac{1}{\deg(s)} \).

We split the proof of this theorem into two lemmas. To be specific, Lemma 46 proves the lower bound and Lemma 47 proves the upper bound.

Lemma 46: \(|C_{\text{max}}| \geq \sum_{s \in \Sigma_n^*} \frac{1}{\deg(s)} \).

Proof: We construct a graph \( G \) with \( \Sigma_n^* \) being the vertex set. For any distinct \( s, t \in \Sigma_n^* \), there is an edge connecting \( s \) and \( t \) if and only if \( M(s) \neq M(t) \). Then the degree of \( s \) in \( G \) is \( \deg(s) = |E(s)| - 1 \).

It is easy to see that a subset of \( \Sigma_n^* \) is a composition code if and only if it is an independent set of \( G \). So \( C_{\text{max}} \) is an independent of maximum size. Thus by [28, Page 100, Theorem 1], we have

\(|C_{\text{max}}| \geq \sum_{s \in \Sigma_n^*} \frac{1}{\deg(s)} = \sum_{s \in \Sigma_n^*} \frac{1}{|E(s)|} \).

\( \square \)

Let \( A = (a_{s,t}) \) be a matrix of size \( 2^n \times 2^n \), whose rows and columns are indexed by strings in \( \Sigma_n^* \), where

\[ a_{s,t} = \begin{cases} 1, & \text{if } M(s) = M(t); \\ 0, & \text{otherwise} \end{cases} \]

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Note that here we do not require $s \neq t$.

**Lemma 47:** $|C_{max}| \leq \sum_{s \in \Sigma_2^n} \frac{1}{|E(s)|}$.

**Proof:** Let $G$ be as in the proof of Lemma 46. Since a composition code is the same as an independent set of $G$, we have

$$|C_{max}| = \max \left\{ \sum_{t \in \Sigma_2^n} x_t : \sum_{s \in \Sigma_2^n} a_{s,t} x_t \leq 1, \forall s \in \Sigma_2^n \right\}.$$

From [29, Section II], we know that

$$|C_{max}| \leq \min \left\{ \sum_{s \in \Sigma_2^n} y_s : \sum_{s \in \Sigma_2^n} a_{s,t} y_s \geq 1, \forall t \in \Sigma_2^n, \right. \left. \text{where } y_s \geq 0, \forall s \in \Sigma_2^n \right\}.$$

To derive our upper bound, let $y_s = \frac{1}{|E(s)|}$. Noticing $|E(s)| = |E(t)|$ if $M(s) = M(t)$, it is easy to verify that $\sum_{s \in \Sigma_2^n} a_{s,t} y_s = 1$ for all $t \in \Sigma_2^n$. Therefore, we can conclude that $|C_{max}| \leq \sum_{s \in \Sigma_2^n} y_s = \sum_{s \in \Sigma_2^n} \frac{1}{|E(s)|}$. \hfill $\square$

Next we show how to compute $|C_{max}|$. In Theorem 43, we proved that the maximum value of $|I_s|$ is $\lfloor (n + 2)/4 \rfloor$, where $s$ ranges over $\Sigma_2^n$. For each $0 \leq b \leq \lfloor (n + 2)/4 \rfloor$, let $M_{n,b}$ denote the number of strings $s \in \Sigma_2^n$ with $|I_s| = b$. Then according to Theorem 45, we have

$$|C_{max}| = \sum_{b=0}^{\lfloor (n+2)/4 \rfloor} M_{n,b} \cdot 2^b.$$

Therefore, the problem of computing $|C_{max}|$ boils down to finding a formula for $M_{n,b}$. For each $0 \leq b \leq \lfloor (n+2)/4 \rfloor$, let

$$P_b(n) \triangleq \begin{cases} j = (j_0, j_1, \ldots, j_{\ell+1}) : & \ell \geq 0; j_0 = 0; \\
& 1 \leq j_1 < \cdots < j_{\ell} \leq \lfloor \frac{n}{2} \rfloor; \\
& j_{\ell+1} = n - j_\ell; \\
& \text{there is exactly } b \text{ }'s \text{ such that } \\
& j_i - j_{i-1} \geq 2 \end{cases}.$$

Then based on Corollary 29, we have

$$M_{n,b} = \sum_{j \in P_b(n)} \prod_{i=1}^{\ell+1} g(j_i - j_{i-1}),$$

where $\ell$ depends on $j$ and function $g(\cdot)$ is defined in Equation (18). Let

$$h(j_i - j_{i-1}) \triangleq \begin{cases} 1, & \text{if } j_i = j_{i-1} \\
2, & \text{if } j_i - j_{i-1} = 1 \\
|S_R(n - 2j_i)|, & \text{if } i = \ell + 1 \text{ and } n - 2j_\ell \geq 2 \\
|A(2j_i - 2j_{i-1})|, & \text{if } j_i - j_{i-1} \geq 2 \text{ and } i \neq \ell + 1. \end{cases}$$

Then Equation (18) and the definition of $P_b(n)$ implies

$$M_{n,b} = 2^b \left( \sum_{j \in P_b(n)} \prod_{i=1}^{\ell+1} h(j_i - j_{i-1}) \right).$$

Therefore, we have

$$|C_{max}| = \sum_{b=0}^{\lfloor (n+2)/4 \rfloor} M_{n,b} \cdot 2^b = \sum_{b=0}^{\lfloor (n+2)/4 \rfloor} \prod_{i=1}^{\ell+1} h(j_i - j_{i-1}) = \sum_{j \in P'(n)} \prod_{i=1}^{\ell+1} h(j_i - j_{i-1}),$$

where $\ell$ depends on $j$ and

$$P'(n) \triangleq \begin{cases} j = (j_0, j_1, \ldots, j_{\ell+1}) : & \ell \geq 0; j_0 = 0; \\
& 1 \leq j_1 < \cdots < j_{\ell} \leq \lfloor \frac{n}{2} \rfloor; \\
& j_{\ell+1} = n - j_\ell. \end{cases}$$

Before moving further, we note that Lemma 46 is based on [28, Page 100, Theorem 1], which is proved by a probabilistic method. So this lemma only guarantees the existence of a code with maximum size, without providing a construction. Our goal in the rest of this section is to study large composition codes with an efficient reconstruction algorithm. In the next subsections, we present several constructions for composition codes, some of which are capable of correcting errors in the read compositions. We start with constructions under the assumption that there are no errors while reading the compositions. In particular, we provide a construction of a composition code with maximum size.

### A. Error-Free Prefix-Suffix Composition Multiset

In this subsection we present a construction of composition codes under the assumption that $M(s)$ is error-free for each codeword $s$.

For a string $s \in \Sigma^n$, let $M_p(s), M_s(s)$ denote the set of all the compositions of prefixes of $s$ and the set of all suffixes of $s$, respectively. When given $M_p(s)$ or $M_s(s)$, it is straightforward to (uniquely) reconstruct $s$. So the idea is to impose additional constraints on $s$ such that when given $M(s)$, we can distinguish $M_p(s)$ from $M_s(s)$. With this idea and Corollary 29 in mind, it is now natural to give the following construction.

**Construction 48:** For all $n \geq 6$, let

$$E_1(n) = \left\{ r^{(1)} \cdots r^{(\ell)} r^{(\ell+1)} \left( t^{(\ell)} \right)^* \left( t^{(1)} \right)^* : \ell \geq 0; j_0 = 0; 1 \leq j_1 < \cdots < j_\ell \leq \lfloor n/2 \rfloor; \\
& j_{\ell+1} = n - j_\ell; \\
& r^{(\ell+1)} \in S_R(j_{\ell+1} - j_\ell)^*; \\
& r^{(i)} \in \Sigma_2^{j_i - j_{i-1} - 1} \text{ for all } 1 \leq i \leq \ell; \\
& r^{(i)} \left( t^{(i)} \right)^* \in \mathcal{A}(2j_i - 2j_{i-1})^* \cup \mathcal{D}(2) \right.$$n

for all $1 \leq i \leq \ell$, where $\mathcal{A}(m)^* \triangleq \{ u : u^* \in \mathcal{A}(m) \}$ for every even $m \geq 4$.

It follows from the construction of $E_1(n)$ that for every $s \in E_1(n)$, $\text{wt}_H(s_1^n) \geq \text{wt}_H(s_{n-j}^{n-1})$ for all $1 \leq j \leq n$. Therefore, we have

$$|C_{max}| = \sum_{b=0}^{\lfloor (n+2)/4 \rfloor} M_{n,b} \cdot 2^b = \sum_{b=0}^{\lfloor (n+2)/4 \rfloor} \prod_{i=1}^{\ell+1} h(j_i - j_{i-1}) = \sum_{j \in P'(n)} \prod_{i=1}^{\ell+1} h(j_i - j_{i-1}),$$

where $\ell$ depends on $j$ and

$$P'(n) \triangleq \begin{cases} j = (j_0, j_1, \ldots, j_{\ell+1}) : & \ell \geq 0; j_0 = 0; \\
& 1 \leq j_1 < \cdots < j_{\ell} \leq \lfloor \frac{n}{2} \rfloor; \\
& j_{\ell+1} = n - j_\ell. \end{cases}$$

Before moving further, we note that Lemma 46 is based on [28, Page 100, Theorem 1], which is proved by a probabilistic method. So this lemma only guarantees the existence of a code with maximum size, without providing a construction. Our goal in the rest of this section is to study large composition codes with an efficient reconstruction algorithm. In the next subsections, we present several constructions for composition codes, some of which are capable of correcting errors in the read compositions. We start with constructions under the assumption that there are no errors while reading the compositions. In particular, we provide a construction of a composition code with maximum size.
Recall that $t \sim s$ if and only if $t \in \text{Swap}(s, A)$ for some $A \subseteq I_s$. It is straightforward to verify that if $s \in \mathcal{E}_1(n)$, $t \sim s$, and $t \neq s$, then there exists some $j_0$ such that $\text{wt}_H(t_{j_0}^i) < \text{wt}_H(t_{j_0-j_0+1}^i)$ and hence $t \notin \mathcal{E}_1(n)$. That is to say, $\mathcal{E}_1(n)$ is a composition code. Furthermore, since $\text{wt}_H(s_{j}^i) \geq \text{wt}_H(s_{j+n-j+1}^i)$ for all $1 \leq j \leq n$, it is possible to distinguish $\mathcal{M}_p(s)$ from $\mathcal{M}_s(s)$ when given $\mathcal{M}(s)$. Thus, any $s \in \mathcal{E}_1(n)$ can be uniquely reconstructed from $\mathcal{M}(s)$. The reconstruction process is straightforward by the discussions above.

By definition, the size of $\mathcal{E}_1(n)$ is

$$|\mathcal{E}_1(n)| = \sum_{j \in P^t(n)} \prod_{i=1}^{\ell+1} h(j_i - j_{i-1}),$$  \hspace{1cm} (22)

where $P^t(n)$ and $h(\cdot)$ are given in Equation (19) and Equation (21), respectively. Compared to Equation (20), we can see that $\mathcal{E}_1(n)$ has the maximum size and hence is optimal in terms of size.

By letting $\ell = 0$, it follows from Equation (22) that

$$|\mathcal{E}_1(n)| > |S_i(n)| > \frac{n-\frac{n}{2} \log_2(n) + 2.5 + \frac{1}{2} \log_2(n)^4}{\sqrt{\pi(n+2)}}.$$  \hspace{1cm} (23)

Hence, the redundancy of $\mathcal{E}_1(n)$, which is defined to be $n - \log_2 |\mathcal{E}_1(n)|$, is less than $\frac{1}{2} \log_2(n) + 2.5 + \frac{1}{2} \log_2(n)^4$ when $n \geq 2$ since $n + 2 < 2n$. So the code $\mathcal{E}_1(n)$ has asymptotic rate 1 as $n$ goes to infinity. To the best of our knowledge, there is no work focusing on reconstructing a single string. The most related work is [26]. In that work, the authors constructed a code $C(n, h)$ (where $h \geq 2$ and $n$ is the length of the code) which can reconstruct $h$ code strings simultaneously, where $h \leq h$. In particular, the code $C(n, h)$ can also reconstruct a single code string. They showed that this code has an asymptotic rate of $1/h$. So for our setting of reconstructing a single code string, the rate of our code is higher. It must be pointed out that the code $C(n, h)$ has an efficient encoding algorithm and a stronger property, that is, multiple code strings can be reconstructed simultaneously.

It is worth noticing that Construction 48 is under the assumption that the prefix-suffix composition multiset $\mathcal{M}(s)$ is error-free. Unfortunately, this assumption is false in practical applications because the MS/MS readout technique is error-prone. The authors of [26] described the following two types of errors.

1) **Missing of prefixes or suffixes.** These types of errors occur when the masses of some prefixes and suffixes are not measured and/or reported.

2) **Mass reducing substitution errors.** These types of errors occur when a composition is mistakenly read out as a composition of the same length, but with a smaller Hamming weight, i.e., a larger mass is reported as a smaller mass due to the loss of atoms during the fragmentation process.

More details regarding the errors can be found at the beginning of [26, Section V]. The following are examples of the type of errors mentioned above.

Example 49: Let $s = 101010$. One can check that $s = 101010 \in \mathcal{E}_1(6)$ with $\ell = 1$ and $j_1 = 2$. Its prefix-suffix composition multiset is

$$\mathcal{M}(s) = \{1, 01, 012, 021^2, 0^{2}1^{3}, 0^{3}1^{3}, 0^{3}1^{2}, 0^{2}1^2, 0^{2}1, 01, 0 \}.$$

1) Consider the erroneous prefix-suffix composition multiset

$$\tilde{\mathcal{M}}_1(s) = \{1, 01, 012, 021^2, 0^{2}1^{3}, 0^{3}1^{2}, 0^{2}1^2, 0^{2}1, 01, 0 \}.$$

One can see that the prefix composition $0^{2}1^{3}$ is missing.

2) Consider the erroneous prefix-suffix composition multiset

$$\tilde{\mathcal{M}}_2(s) = \{1, 01, 012, 0^{2}1^{2}, 0^{2}1^{3}, 0^{3}1^{2}, 0^{2}1^2, 0^{2}1, 01, 0 \}.$$

One can see that the prefix composition $0^{2}1^{2}$ is mistakenly read out as $0^{3}1^{2}$.

Three codes correcting the first type of errors were proposed in [26], under the setting of reconstructing multiple strings simultaneously. Next, we will modify Construction 48 to obtain two composition codes that can correct these two types of errors under the setting of reconstructing a single string. Specifically, the first code can combat missing prefixes or suffixes and the second code can combat mass-reducing substitution errors, which were not studied in [26]. In both cases, the number of errors is assumed to be a constant.

### B. Missing of Prefixes or Suffixes

The goal of this subsection is to create composition codes that can correct errors of the first type. Suppose $s \in \mathcal{E}_1(n)$ is the string that needs to be reconstructed. For each $1 \leq i \leq n - 1$, if either $c(s_i^i)$ or $c(s_{i+1}^n)$ is missing, typically the other composition cannot be used for reconstruction because it is unclear whether it is a prefix or suffix composition. An example that illustrates this issue is provided below.

**Example 50:** We continue 1) of Example 49. Recall that the prefix composition $0^{2}1^{3}$ is missing while the suffix composition $0^{3}1^{2}$ is not. However, we do not know whether $c(s_{i}^i) = 0^{3}1^{2}$ or $c(s_{i+1}^n) = 0^{2}1^{3}$. If we take $c(s_{i}^i) = 0^{3}1^{2}$, then we can obtain (from $\mathcal{M}_s(s)$) the string 101010. If we take $c(s_{i}^i) = 0^{2}1^{3}$, then we can obtain (from $\mathcal{M}_p(s)$) a string $t = 101001$. It is easy to verify that $t \in \mathcal{E}_1(6)$ with $\ell = 2$, $j_1 = 1$ and $j_2 = 2$. This means that we cannot determine whether $s = 101010$ or $101001$.

In general, even if only one of $c(s_i^i)$ and $c(s_{i+1}^n)$ is missing, we still regard both of them as lost. In 1) of Example 49, we obtain $\mathcal{M}_p(s) = \{1, 01, 012, 0^{2}1^{2}, 0^{3}1^{3} \}$, where “-?” denotes the missing $c(s_i^i)$. Then we can reconstruct $s$ as $\tilde{s} = 1010\text{-?}$. However, “-?” means that the symbol in this position is erased. Generally, if $c(s_i^i)$ is missing, then in the reconstructed string $\tilde{s}$, at most two symbols $s_i$ and $s_{i+1}$ are erased (here we only consider reconstructing a string from $\mathcal{M}_p(s)$). In our model, we suppose that at most $t$ prefix compositions are missing, where $t$ is a constant. Then we need
to correct at most $2t$ erasures in $\bar{s}$. With this in mind, we can encode strings in $E_1(n)$ by the following three steps:

**Step 1:** Choose a systematic $[m, n, \geq 2t + 1]$ code $E$ over finite field $\mathbb{F}_q$, where $q = 2^s$ is a power of 2. Fix a basis $B$ of $\mathbb{F}_q$ over $\mathbb{F}_2$.

**Step 2:** For each $s \in E_1(n)$, we firstly encode it using $E$ into $r\prime$, where $r\prime$ is the string of redundancy symbols. Secondly, according to $B$, we convert each symbol of $r\prime$ into a binary string of length $s$. Then we concatenate these length-$s$ binary strings into $r$. At last, we get an intermediary string $sr$.

**Step 3:** The output codeword is $c = r\prime s r \in \Sigma_2^n$, where $N = n + 2sr(E)$ and $r(E)$ is the redundancy of $E$. This step is to ensure that $\text{wt}_H(c_i^j) \geq \text{wt}_H(c_{N-j+1}^N)$ for all $1 \leq j \leq N$.

The decoding of $s$ from $r\prime s r$ is straightforward. Denote the resulting code by $E_2(n)$. Then $r(E_2(n)) = r(E_1(n)) + 2s \cdot r(E)$.

Since for each cyclic code there is a systematic encoder [30, Page 128] and BCH codes are in most cases among the best cyclic codes, we can let $E$ be a narrow-sense primitive BCH code over $\mathbb{F}_2$ (thus $s = 1$) with designed distance $2t + 1$.

By [30, Theorem 5.1.7], we have $r(E) \leq t \log_2(m + 1)$ and so $m - t \log_2(m + 1) \leq n$. On the other hand, by the Singleton bound, we have $n \leq m - 2t$ and hence $r(E) \geq 2t$. Since $N = n + 2sr(E)$, it follows that $m - t \log_2(m + 1) + 4t \leq N \leq m - 2t + 2t \log_2(m + 1)$. Now we have

$$\frac{\log_2(|E_2(n)|)}{N} = \frac{\log_2(|E_1(n)|)}{N} = n - r(E_1) > \frac{m - t \log_2(m + 1) - 2.5 - \log_2(\sqrt{m - 2t}) - \log_2(\sqrt{m - 2t})}{m - 2t + 2t \log_2(m + 1)}.$$

Therefore, the code $E_2(n)$ has asymptotic rate 1.

### C. Mass Reducing Substitution Errors

In this subsection, we aim to construct composition codes capable of correcting the second type of errors. Throughout this part, we assume that errors are not in batches, i.e., if the composition of the prefix length $i$ is erroneous, then the composition of the prefixes lengths $i - 1$ and $i + 1$ are read out correctly, and similarly, if the composition of the suffix length $i$ is erroneous, then the composition of the suffixes lengths $i - 1$ and $i + 1$ are read out correctly. Although this requirement seems kind of impractical, the ideas used here might shed some light on further study of this type of errors (see the last open problem in [26, Section VI]). In this part, we only give some light on further study of this type of errors (see the above).

**Example 51:** (1) (compatible error) We continue 2) of Example 49. The prefix composition $0^i1^3$ is mistakenly read out as $0^i1^{i+1}$. From $M(s)$, we obtain $M_p(s) = \{1, 01, 012, 0^i1^2, 0^i1^3\}$. Then the string $s$ is reconstructed as $\bar{s} = 101001$, which is also a codeword in $E_1(6)$ (see Example 50). Note that $\bar{s}$ is obtained from $s$ by substituting $s_8$ with 01.

2) (incompatible error where $b_3 = b_2 + 1 = b_1 + 1$) Consider the string $s = 110010 \in E_1(6)$. Its prefix-suffix composition multiset is

$$M(s) = \{1, 1^2, 01^2, 0^31^1, 0^21^2, 0^31^3, 0^31^3, 0^31^3, 0^31^2, 0^31^4, 0^21^1, 0^21^2, 0^21^3\}.$$ 

Suppose that $M(s)$ is measured as

$$\tilde{M}(s) = \{1, 1^2, 01^2, 0^31^1, 0^21^2, 0^31^3, 0^31^3, 0^31^3, 0^31^2, 0^31^4, 0^21^1, 0^21^2, 0^21^3\}.$$ 

Clearly, the prefix composition $0^21^2$ is mistakenly read as $0^31^1$. We can obtain $M_p(s) = \{1, 1^2, 01^2, 0^31^1, 0^21^2, 0^31^3\}$ and thus string $s$ is reconstructed as $\bar{s} = 01\_0\_0\_0$. There are two erasures in $\bar{s}$.

3) (incompatible error where $b_3 = b_2 = b_1 + 1$ and $b_2 - b'_2 = 2$) Consider the string $s = 1110000 \in E_3(6)$. Its prefix-suffix composition multiset is

$$M(s) = \{1, 1^2, 1^3, 01^3, 0^21^3, 0^31^3, 0^31^3, 0^31^3, 0^31^3, 0^31^2, 0^31^4, 0^21^1, 0^21^2, 0^21^3\}.$$ 

Suppose that $M(s)$ is measured as

$$\tilde{M}(s) = \{1, 1^2, 0^21^2, 0^31^2, 0^31^3, 0^31^3, 0^31^3, 0^31^3, 0^31^2, 0^31^4, 0^21^1, 0^21^2, 0^21^3\}.$$ 

Clearly, the prefix composition $1^3$ is mistakenly read as $0^21^2$. We can obtain $M_p(s) = \{1, 1^2, 1, 01^3, 0^21^3, 0^31^3\}$ and thus string $s$ is reconstructed as $\bar{s} = 11\_0\_0\_0$. There are two erasures in $\bar{s}$.
We now define the error model. For given integers \(e_1, e_2 \geq 1\) and \(t \geq 2\), we formulate an \((e_1, e_2, t)\)-mass reducing substitution error model as follows:

- If a composition \(0^{a_1}1^{b_1}\) is mistakenly read out as \(0^{a+b-b'}1^{b'}\), then \(1 \leq b - b' \leq t\).
- There are at most \(e_1\) compatible errors.
- There are at most \(e_2\) incompatible errors.
- If the composition of the prefix length \(i\) is erroneous, then the composition of the prefixes length \(i - 1\) and \(i + 1\) are not erroneous.
- If the composition of the suffix length \(i\) is erroneous, then the composition of the suffixes length \(i - 1\) and \(i + 1\) are not erroneous.

We can encode \(E_1(n)\) as follows:

**Step 1:** Choose a systematic \([m, n, d]\) code \(E\) over finite field \(\mathbb{F}_q\) that can correct \(e_1\) substitutions and \(e_2\) erasures, where \(q = 2^e\) is a power of 2. Fix a basis \(B\) of \(\mathbb{F}_q\) over \(\mathbb{F}_2\).

**Step 2:** For each \(s \in E_1(n)\): first, encode it using \(E\) into \(sr^t\), where \(r^t\) is the string of redundancy symbols. Second, according to \(B\), convert each symbol of \(r^t\) into a binary string of length \(s\). Then concatenate these length-\(s\) binary strings into \(r\). Finally, we obtain an intermediary string \(s' = r^t sr^t \in \Sigma_{n+1}\). This step is to ensure that \(wt_H(s') = wt_H(s'_{n+1}) + 1\) for all \(1 \leq j \leq n+1\).

**Step 3:** The output codeword is \(c = 1^{r^t} sr^t \in \Sigma_{n+2}\), where \(N = n + 2sr(E) + 2t\) and \(r(E)\) is the redundancy of \(E\). This step is to ensure that \(wt_H(c') = wt_H(c_{N+1}) + t\) for all \(t + 1 \leq j \leq N - t\). Denote the resulting code by \(E_3(n)\). Then: \(r(E_3(n)) = r(E_1(n)) + 2s + r(E) + 2t\).

Since the first \(t\) bits of \(c\) are all 1’s and the last \(t\) bits of \(c\) are all 0’s, we can distinguish \(M_p(s)\) from \(M_s(s)\), due to the property that \(wt_H(c') \geq wt_H(c_{N+1}) + t\) for all \(t + 1 \leq j \leq N - t\). Now the decoding of \(s\) from \(1^{r^t} sr^t 0^t\) is straightforward. Therefore, the code \(E_3(n)\) can correct \((e_1, e_2, t)\)-mass reducing substitution errors.

**Lemma 52 [30, Theorem 1.11.6]:** Let \(C\) be an \([m, n, d]\) code. Then \(C\) is capable of correcting any pattern of \(2e_1\) substitution errors and \(2e_2\) erasures if \(4e_1 + 2e_2 \leq d - 1\).

With Lemma 52 in hand, we let \(E\) be a narrow-sense primitive BCH code over \(\mathbb{F}_2\) with designed distance \(2(2e_1 + e_2) + 1\), then \(r(E) \leq (2e_1 + e_2) \log_2(m + 1) + 1\) [30, Theorem 5.1.7]. This implies that \(m - (2e_1 + e_2) \log_2(m + 1) < n \leq m - 2(2e_1 + e_2)\) and \(N = n + 2sr(E) + 2t \leq m - 2(2e_1 + e_2) + 2(2e_1 + e_2) \log_2(m + 1) + 2t\). We now obtain Equation (23), shown at the bottom of the next page. Therefore, the code \(E_3(n)\) has asymptotic rate 1.

**VII. r-LENGTH LIMITED COMPOSITION MULTISET**

In this section we suggest a new reconstruction model. Recall that in order to store data, two molecules with different masses are synthesized to represent the 0 and 1 symbols. These molecules are combined into a string to represent the data that the user wishes to store. This string is a synthetic polymer. In order to read the data, the synthetic polymers are read by a tandem mass (MS/MS) spectrometer. As part of the reading process, the polymer molecule breaks in a position which is chosen at random, creating two fragments. The breaking of the molecule is usually made by collisions, bombardment of the molecule with electrons or photons, or a chemical reaction (a molecule can also break down due to instability). It is reasonable to assume that longer molecule fragments tend to break more often than shorter fragments. For example, using electron bombardment, it is more likely that the electron will hit a longer polymer fragment than hitting a short fragment. Thus, after obtaining two fragments, if the fragments are long enough, each fragment will be broken again into two fragments. As a result, when the mass readings of a polymer molecule are obtained, the compositions of long fragments will not appear in those readings, leaving us with a composition multiset of shorter substrings.

Since our results in this section apply to general alphabets, we assume our alphabet is a \(q\)-ary alphabet \(\Sigma_q = \{a_i : a_i = i, 0 \leq i \leq q - 1\}\), where \(q \geq 2\) is an integer. Let \(\Sigma_q\) denote the set of all strings over \(\Sigma_q\) of finite length. For a string \(s \in \Sigma_q^*,\) we define the composition \(c(s)\) of \(s\) to be \(c(s) = a_{e_0} a_{e_1} \cdots a_{e_{q-1}},\) where \(e_i\) is the number of appearances of the symbol \(a_i\) in \(s\) and we omit \(a_{e_i}^t\) if \(e_i = 0\). For example, \(s = 001112, t = 002002 \in \Sigma_3,\) then \(c(s) = 0^21^22^1\) and \(c(0) = 0^32^1.\) For a length-\(n\) string \(s\) and an integer \(1 \leq k \leq n,\) define \(C_k(s) = \{c(s^{(k+1)}) : 1 \leq i \leq n - k + 1\}\).

To model the phenomenon described as above, dubbed the \(r\)-length limited composition multiset model, we assume that the composition multiset does not contain the compositions of substrings longer than \(r\). The mathematical model is defined next.

**Definition 53:** For a string \(s \in \Sigma_q^n\) and for a positive integer \(r \leq n,\) we denote by \(C_{\leq r}(s)\) the multiset of compositions of all substrings of length at most \(r\) in \(s,\) i.e.,

\[
C_{\leq r}(s) = \bigcup_{i=1}^r C_i(s).
\]

In this section we initiate the study of \(r\)-length limited composition multiset model, and present a simple coding scheme for the case where \(r = r(n)\) is a linear function of the length of the strings \(n.\) To that end, we define the following relation on strings. For \(u, v \in \Sigma_q^n,\) we say that \(u, v\) are cycle-equivalent, denoted \(u \sim v,\) if \(v\) is a cyclic shift of \(u.\) It is immediate that \(\sim\) is an equivalence relation. A string \(u\) is called minimal if it is the minimal (with respect to the lexicographic order)\(^5\) string among all its equivalent strings. A string \(u \in \Sigma_q^n\) is called periodic if there exists a \(t \geq 1\) such that \(u_i = u_{i+t}\) for all \(1 \leq i \leq |u| - t.\) A string is called aperiodic if it is not periodic.

**Definition 54:** A string \(w \in \Sigma_q^n\) is called a Lyndon string (Lyndon word) if \(w\) is minimal and aperiodic. We denote by \(\mathcal{L}_r(q)\) the set of all Lyndon words of length \(r\) over \(\Sigma_q.\)

Lyndon words were first introduced in [31] and [32] and were used to study free abelian groups. They have been

\(^5\)There is a natural order in \(\Sigma_q,\) i.e., \(a_i \leq a_j\) if and only if \(i \leq j.\)
found useful in various topics such as differential equations, cryptography, and coding theory [33], [34], [35].

For a Lyndon word $l \in \mathcal{L}_r(q)$, denote by $m$ the maximal number of consecutive $a_{q-1}$ symbols that appear at the end of $l$, i.e., $l = l_1l_2 \ldots l_{m-1}a_{q-1}^m$ where $m \geq 0$ is maximal and $l_{r-m} < a_{q-1}$. We construct our code $C_r$ as follows

$$C_r = \{la_{q-1}^{r-m} : l \in \mathcal{L}_r(q)^*\}.$$ 

So a codeword $s \in C_r$ is a Lyndon string of length $r$ followed by a string of repeated $a_{q-1}$ such that $s$ ends with $r$ consecutive $a_{q-1}$ symbols. The strings in $C_r$ have variable length. The shortest codeword has $r + 1$ symbols (obtained by Lyndon words of the form $a_i a_{q-1}^{i-1}$ for $i < q - 1$, and then $s = a_i a_{q-1}^{i-1}$). If $q > 2$, the longest codeword has length $2r$ (obtained by the Lyndon words $a_i \ldots a_i a_j$ with $i < j < q - 1$, concatenated with $a_{q-1}^{i-1}$ to generate $s = a_i a_{q-1}^{i-1} a_j a_{q-1}^{j-1}$); if $q = 2$, the unique longest codeword is $0r^{r+1}1r$ and thus has length $2r - 1$. If length-$n$ codewords are in need, where $n \geq 2r$, such a set $C_r(n)$ can be obtained by padding every codeword in $C_r$ with $a_{q-1}$ until it reaches length $n$.

The main goal of this section is to prove the following corollary which will be readily obtained from the discussion in this section.

**Corollary 55:** Let $\Sigma_q$ be an alphabet of size $q$ and consider the $r$-length limited composition multiset model with a fixed $r$. Every codeword $s \in C_r$ is uniquely reconstructible given $C_{\leq r}(s)$ and the code rate is

$$R = \frac{\log(\mathcal{C}_r)}{2r} = \frac{1}{2} (1 + o(1)).$$

**Remark 56:** The code $C_r$ exists in the space of strings of length at least $r + 1$ and at most $2r - 2r - 1$ when $q = 2$. The size of this space is $\Theta(q^{2r})$. Therefore, we define the code rate using $2r$ in the denominator.

In order to generate the set $\mathcal{L}_r(q)$ we can use the efficient algorithm given in [36] and [37] for finding Lyndon words of length at most $r$, according to their lexicographic order. The algorithm (see Algorithm 1) is presented here for completeness (where $|s|$ denotes the length of string $s$).

**Example 57:** Let $\Sigma_q = \{0, 1, 2, 3\}$ with the usual ordering and let $w = 003103$ be a Lyndon word of length $|w| = 6$. Applying Algorithm 1 with $r = 9$ we first generate $s = 003103030$. Next, we eliminate the last symbol since it is the maximal symbol in our alphabet, to obtain $s = 0031030$. Lastly, we replace the last symbol 0 with its successor 1 to obtain $u = 00310301$. Notice that the algorithm may also provide a string shorter than $r$.

Now we explain how to reconstruct a codeword $s \in C_r$ (or $s \in C_r(n)$ where $n \geq 2r$). Instead of showing that $s$ can be uniquely reconstructed from $C_{\leq r}(s)$, we show that it can be uniquely reconstructed from the multiset $C_r(s)$ of compositions of substrings of length $r$. Notice that there are no Lyndon words of length $r$ with $r$ appearances of $a_{q-1}$.

$$\frac{\log(\mathcal{E}_x(n))}{N} = \frac{\log(\mathcal{E}_x^1(n))}{N} > \frac{m - (2e_1 + e_2) \log_2(m + 1) - 2 - 2 \log_2(\sqrt{m}) - \log_2(m - 2(2e_1 + e_2))}{m - 2(2e_1 + e_2) + 2(2e_1 + e_2) \log_2(m + 1) + 2r}. \quad (23)$$

---

**Algorithm 1** Algorithm That Generates Next Lyndon Words (Up to Length $r$)

**Input:** A Lyndon word $w \in \Sigma_q^*$. **Output:** The next Lyndon word $u$ according to the lexicographic order.

1. Let $s = (w')^T$, i.e., concatenate $w$ to itself to form a string $s \in \Sigma_q^*$ with $s_i = w_i \mod |w|$.
2. **while** last symbol of $s$ is the maximal symbol in lexicographic order **do**
   3. Remove the last symbol from $s$, i.e., $s = s_i |s_i^{-1}$.
   4. **end while**
5. Replace the last symbol in $s$ by its successor according to the Lexicographic order to generate $u$.
6. Return $u$

**Algorithm 2** Algorithm for the Reconstruction of $s$ From $C_r(s)$

**Input:** A multiset $C_r(s)$. **Output:** The unique word $s \in C_r$ that corresponds to $C_r(s)$.

1. Set $T = C_r \setminus \{a_{q-1}^r\}$, i.e., $T$ contains all the compositions in $C_r(s)$ except for $a_{q-1}^r$.
2. Set $w = \varepsilon$ (the empty word).
3. Set $j = 1$, $i = |T|$, and set $s_{i+1} = a_{q-1}^r$.
4. **while** $i > 0$ **do**
   5. Let $k$ be the number of compositions in $T$ of the form $a_0^{d_0} \ldots a_1^{d_1} a_2^{d_2} a_{q-2}^{d_{q-2}}$ with $d_0 + \ldots + d_{q-2} = j$.
   6. From $a_0^{d_0} \ldots a_{q-2}^{d_{q-2}}$, remove all the symbols that appear in $w$. Denote by $l$ the remaining symbol.
   7. Set $s_i = l$ and set $s_{i-k+1} \ldots s_{i+1} = a_{q-1}^k$.
   8. **end while**
9. Set $w = a_0^{d_0} \ldots a_{q-2}^{d_{q-2}}$, $i = i - k$, and $j = j + 1$.
10. **end while**
11. Return $s$

Thus, all the compositions $a_{q-1}^r$ refer to the $r$ (or more) $a_{q-1}$ symbols at the end of the codeword $s$. Those last $a_{q-1}$ symbols mark the end of the string. Assume that the codeword is $s = la_{q-1}^{r-m} = l_1 l_2 \ldots l_{r-m} a_{q-1}^{r-m}$. Since $l_{r-m} \neq a_{q-1}$, the composition $a_{q-1}^{r-m} l_{r-m}$ determine the symbol $l_{r-m}$. If $l_{r-m} = a_{q-1}$ then we obtain the composition $a_{q-1}^{r-m} l_{r-m}$ twice. Continuing in the same way, the entire codeword $s$ can be constructed backward.

The reconstruction process is formally described in Algorithm 2. The next lemma justifies the correctness of this algorithm.

**Lemma 58:** The process described in Algorithm 2 reconstructs $s$ from $C_r(s)$.

**Proof:** Notice that in every entry to the while loop, the number of $a_{q-1}$ symbols in the part of $s$ which is left
unreconstructed is decreased by 1. Since \( s_{r}^{i} \) is a Lyndon word, at each step, there is a unique composition in \( T \) with a maximal number of \( a_{q-1} \) symbols (although this composition may appear several times in \( T \)). Thus, the word \( s \) can be reconstructed “backward” using the above algorithm. Notice that because \( s_{r}^{i} \) is Lyndon, and because we consider compositions of length \( r \), there is at most a single composition that does not contain \( a_{q-1} \). If such a composition exists, it corresponds to the first \( r \) symbols of \( s \).

The reconstruction process is shown in the next example.

Example 59: Assume \( \Sigma_{q} = \{0, 1, 2, 3\} \) and \( s \in C_{r} \) with \( r = 6 \) is such that its \( r \)-length limited composition multiset is given by

\[
\{\{3^{6}, 2^{5}, 3^{5}, 2^{2}3^{4}, 0^{2}2^{2}3^{4}\}\}.
\]

We follow Algorithm 2 to recover string \( s \). We set by starting \( j = 1 \), \( |C_{r}| = 5 \) and \( i = |T| = 4 \). Thus, we set \( s_{r+1}^{i} = s_{1}^{0} = 3^{6} \).

We enter the first while loop and we search for compositions in which \( q \) appears 5 times. We have two such (identical) compositions, \( 2^{3^{5}} \) so \( k = 2 \). Since \( w = \varepsilon \), we remove nothing from the symbol 2 and we obtain \( l = 2 \). We set \( s_{4} = 2 \) and \( s_{4-2+1} = s_{3} = 3 \). Next, we remove from \( T \) the 2 compositions \( 2^{3^{5}} \) and obtain \( T = \{\{2^{2}3^{4}, 0^{2}2^{2}3^{4}\}\} \), set \( w = 2, i = 4 - 2 = 2, \) and \( j = 2 \).

We repeat the while loop with the new parameters. There is a single \( (k = 1) \) composition with \( 3^{r-i} = 3^{4} \), namely \( 2^{2}3^{4} \). Removing from \( 2^{2}3^{4} \) the symbols in \( w = 2 \) we obtain \( l = 2 \). Therefore, \( s_{i} = s_{2} = 2 \). We now set \( T = \{\{0^{2}2^{3}3^{4}\}\} \), \( w = 2^{2}, i = 2 - 1 = 1, \) and \( j = 3 \).

Repeating the loop, we have \( k = 1 \) with a single composition that contains the symbol 3 \( j = 3 \) times, the composition \( 0^{2}2^{3}3^{4} \). Removing from \( 0^{2}2^{3}3^{4} \) the symbols that appear in \( w = 2^{2} \) we obtain \( l = 0 \). We set \( s_{1} = 0 \) and set \( i = 0 \) which means we are done.

We return \( s = s_{1}s_{2} \ldots s_{r+|C_{r}|} = 0232233333 \).

The number of length-\( r \) Lyndon words over an alphabet \( \Sigma_{q} \) is given by Witt’s formula [38]

\[
|\mathcal{L}_{r}(q)| = \frac{1}{r} \sum_{d|r} \mu(d) q^{r/d}
\]

where \( \mu \) is the Möbius function.

Since \( \mu(\cdot) \in \{0, 1, -1\} \), we can bound the size \( |\mathcal{L}_{r}(q)| \). Notice that \( \mu(1) = 1 \) which implies that

\[
|\mathcal{L}_{r}(q)| \leq \frac{1}{r} q^{r} + \frac{1}{r} \sum_{d|r, d > 1} \mu(d) q^{r/d}
\]

and similarly,

\[
|\mathcal{L}_{r}(q)| \leq \frac{1}{r} q^{r} - q^{r/2}.
\]

Thus, the number of Lyndon words over an alphabet of size \( q \) is \( |\mathcal{L}_{r}(q)| = \frac{q^{r}}{r} (1 + o(1)) \) and we obtain that

\[
R = \frac{\log_{q} |\mathcal{L}_{r}(q)|}{2r}
\]

\[
= r - \frac{\log_{q}(r) + \log_{q}(1 + o(r))}{2r}
\]

\[
= \frac{1}{2} - \frac{\log_{q}(r)}{2r} + o(1/\alpha)
\]

\[
= \frac{1}{2}(1 + o(1/r)).
\]

Remark 60: A similar encoding process can be used to reconstruct a string \( s \in C_{r}(n) \) when \( n \geq 2r \). Thus, it is possible to use \( C_{r}(n) \) as a code of length \( n \). Applying similar calculations with \( n \) instead of \( 2r \), we obtain code rate \( R = r/n \). Thus, when \( r = n/\alpha \) is a linear function of \( n \), such as \( r(n) = n/\alpha \) for some \( \alpha \in \mathbb{N} \), we obtain an asymptotic rate of \( 1/\alpha \). If \( r(n) = o(n) \) the asymptotic code rate is 0.

VIII. Conclusion

In this paper, we first solved the problem of whether a string can be uniquely reconstructed up to reversal or not, when only given its prefix-suffix composition multiset:

- we characterized their structures;
- we give an explicit formula of their numbers;
- for any given string \( s \), we gave a method to find all the strings that share a common prefix-suffix composition multiset with \( s \), and we gave a formula of the number of such strings.

As applications of the third result, we presented a way to find a composition code \( \mathcal{E}_{1}(n) \) of maximum size. It is shown that the redundancy of this subcode is less than \( 0.5 \log(n) + 0.5 \log(\pi) + 2 + 0.5 \log(1 + \alpha) \) for any \( 0 < \alpha < 1 \) and \( n \geq 2/\alpha \).

In addition, we studied two error models of composition errors. For each error model, we modified \( \mathcal{E}_{1}(n) \) to get a composition code that can correct the studied type of error; each code has an asymptotic rate 1.

The authors of [26] focused on the problem of uniquely reconstructing a set of strings from the union of their prefix-suffix composition multisets. We believe that our results in this paper might be a helpful step toward fully understanding this problem. It is also interesting to extend our results to this situation. Moreover, in either this paper or [26], the constructed codes use the idea that we can distinguish the prefix composition multiset from the suffix composition multiset. An interesting question is whether it is possible to construct codes without this property, such that a set of strings can be uniquely reconstructed from the union of their prefix-suffix composition multiset. We leave this problem for a future endeavor.

Lastly, we presented the problem of \( r \)-length limited composition multiset reconstruction, and provided a simple coding scheme that obtains a rate of \( 1/2 \). This problem gives rise to many interesting problems. For example, is it possible to improve the code presented in this problem for the case \( r = r(n) \) is a linear function of \( n \)? What about codes for a constant \( r \) or for different functions \( r(n) \)? Is
it possible to find bounds on the capacity of such codes? Also, some generalizations of the model itself are of interest. For example, consider the case in which compositions of substrings of lengths between $r_1$ and $r_2$ are obtained. Consider the appearances of errors, or even consider a scattered noise model in which the compositions are obtained according to a probability which is a function of their length. As an example, consider the case that all the compositions up to length $r_1$ are obtained, and the compositions of strings of length $r_1 + j$ are obtained with a probability of $1/2^{j-1}$.

ACKNOWLEDGMENT

The authors express their gratitude to the anonymous reviewers for their detailed and constructive comments which are very helpful to the improvement of the presentation of this article. They would also like to thank Prof. Antonia Wachter-Zeh, the associate editor, for her excellent editorial job.

REFERENCES

[1] J. G. David Reinsel and J. Rydning. (2018). The Digitization of the World From Edge to Core. IDC. [Online]. Available: https://www.seagate.com/files/whitepaper.pdf
[2] G. M. Church, Y. Gao, and S. Kosuri, “Next-generation digital information storage in DNA,” Science, vol. 337, no. 6102, p. 1628, Sep. 2012.
[3] N. Goldman et al., “Towards practical, high-capacity, low-maintenance information storage in synthesized DNA,” Nature, vol. 494, no. 7435, pp. 77–80, Feb. 2013.
[4] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, “Robust chemical preservation of digital information on DNA in silica with error-correcting codes,” Angew. Chem. Int. Ed., vol. 54, no. 8, pp. 2552–2555, Feb. 2015.
[5] S. M. H. Tabatabaei Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, “A rewritable, random-access DNA-based storage system,” Sci. Rep., vol. 5, no. 1, p. 14318, Sep. 2015.
[6] Z. Victor, M. Z. Reza, S. S. Gurtej, M. C. George, and L. H. William, “Nucleic acid memory,” Nat. Mater., vol. 15, no. 4, pp. 366–370, Mar. 2016.
[7] A. Al Ouahabi, J.-A. Amalian, L. Charles, and J.-F. Lutz, “Mass spectrometry sequencing of long digital polymers facilitated by programmed inter-byte fragmentation,” Nature Commun., vol. 8, no. 1, p. 967, Oct. 2017.
[8] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, “Portable and error-free DNA-based data storage,” Sci. Rep., vol. 7, no. 1, p. 5011, Jul. 2017.
[9] M. G. Rutten, F. W. Vaandrager, J. A. Elemans, and R. J. Nolte, “Encoding information into polymers,” Nature Rev. Chem., vol. 2, no. 11, pp. 365–381, Nov. 2018.
[10] R. Heckel, G. Mikutis, and R. N. Grass, “A characterization of the DNA data storage channel,” Sci. Rep., vol. 9, no. 1, p. 9663, Jul. 2019.
[11] N. Takahashi, M. H. Nguyen, K. Strauss, and L. Ceze, “Demonstration of end-to-end automation of DNA data storage,” Sci. Rep., vol. 9, no. 1, p. 4998, Mar. 2019.
[12] S. K. Tabatabaei et al., “DNA punch cards: Encoding data on native DNA sequences via topological modifications,” BioRxiv, Jun. 2019, Art. no. 672394.
[13] S. K. Tabatabaei et al., “DNA punch cards for storing data on native DNA sequences via enzymatic nicking,” Nature Commun., vol. 11, no. 1, p. 1742, Apr. 2020.
[14] K. Matange, J. M. Tuck, and A. J. Keung, “DNA stability: A central design consideration for DNA data storage systems,” Nature Commun., vol. 12, p. 1358, Mar. 2021.
[15] K. Launay et al., “Precise alkoxylamine design to enable automated tandem mass spectrometry sequencing of digital poly(phosphodiesters),” Angew. Chem. Int. Ed., vol. 60, no. 2, pp. 917–926, Jan. 2021.
[16] C. Pan, S. K. Tabatabaei, S. M. H. T. Yazdi, A. G. Hernandez, C. M. Schroeder, and O. Milenkovic, “DNA stability: A central design consideration for DNA data storage systems,” Nature Commun., vol. 13, p. 2984, May 2022.