SIAM J. DISCRETE MATH.
Vol. 29, No. 3, pp. 1340–1371

STRING RECONSTRUCTION FROM SUBSTRING COMPOSITIONS*

JAYADEV ACHARYA†, HIRAKENDU DAS‡, OLGICA MILENKOVIC§, ALON ORLITSKY¶, AND SHENGJUN PAN‖

Abstract. Motivated by mass-spectrometry protein sequencing, we consider the problem of reconstructing a string from the multisets of its substring composition. We show that all strings of length 7, one less than a prime and one less than twice a prime, can be reconstructed uniquely up to reversal. For all other lengths, we show that unique reconstruction is not always possible and provide sometimes-tight bounds on the largest number of strings with given substring compositions. The lower bounds are derived by combinatorial arguments, while the upper bounds follow from algebraic approaches that lead to precise characterizations of the sets of strings with the same substring compositions in terms of the factorization properties of bivariate polynomials. Using results on the transience of multidimensional random walks, we also provide a reconstruction algorithm that recovers random strings over alphabets of size ≥ 4 from their substring compositions in optimal near-quadratic time. The problem considered is related to the well-known turnpike problem, and its solution may hence shed light on this longstanding open problem as well.

Key words. string reconstruction, mass spectrometry, turnpike problem, polynomial factorization, backtracking algorithm, protein sequencing

AMS subject classification. 05

DOI. 10.1137/140962486

1. Motivation.

A protein is composed of long peptide chains, i.e., sequences of amino acids whose composition and order determine the protein's shape and functional properties. A common family of technological methods for "reading" amino-acid sequences is *mass spectrometry* [7, 20]. Mass spectrometers takes a large number of identical proteins, randomly break the proteins into substrings, and analyze the resulting mixture to determine the substring weights. The substring weights are consequently used to infer the amino-acid sequence.

We make two simplifying assumptions that reduce protein mass-spectrometry sequence reconstruction to a simply-stated combinatorial problem that we then proceed to analyze.

Assumption 1. The *composition* of every peptide substring may be deduced uniquely from its weight. For example, let A, B, and C represent the letters assigned to three amino acids with respective weights 13, 7, and 4. Consider strings consisting of these three letters. A string of weight 11 clearly consists of one B and one C. Similarly, an observed weight 18 implies that the string consists of two B's and one C. However, a weight 20 string could potentially arise from one A and one B,

*Received by the editors March 27, 2014; accepted for publication (in revised form) June 2, 2015; published electronically August 4, 2015. Parts of this work appeared in *Proceedings of the IEEE International Symposium on Information Theory*, 2010 and 2014.
http://www.siam.org/journals/sidma/29-3/96248.html

†Department of Electrical Engineering and Computer Science, Massachusetts Institite of Technology, Cambridge, MA 02139 (jayadev@csail.mit.edu).

‡Yahoo Labs!, Sunnyvale, CA 94089 (hdas@yahoo-inc.com).

§Department of Electrical and Computer Engineering, University of Illinois, Urbana Champaign, IL 61801 (milenkov@illinois.edu).

¶Department of Electrical and Computer Engineering, University of California, San Diego, CA 92093 (alon@ucsd.edu).

‖Google, Mountain View, CA 94043 (s1pan@ucsd.edu).

or from 5 C's; hence, we cannot deduce the composition of the string from its weight alone. The underlying assumption states that such confusions never arise.

Assumption 2. Each peptide bond gets cut independently with the same probability p. For example, if the sequence equals ABC, then the partition $A|B|C$ is obtained with probability p^2, the partitions $A|BC$ and $AB|C$ are obtained with probability $p(1-p)$, while the partition ABC is obtained with probability $(1-p)^2$.

While both assumptions are clearly idealized, there exist settings where they roughly hold. For example, although there exist amino acids with the same weight, one can distinguish the amino acids via reference genomes or via additional biochemical testing. In addition, under the "uniform coverage" model, one may assume that each peptide bond gets cut with the same probability 1/2. More flexible assumptions, such as some amino-acid weight similarities, or unequal cut probabilities, may be considered as well (see e.g., section 16), but the two current assumptions provide a clean formulation amenable for combinatorial and algorithmic analysis. Additionally, independently of their precise validity, these assumptions convert protein sequence reconstruction to a simple string reconstruction problem that is interesting in its own right. The problem is also a combinatorial simplification of the longstanding open *turnpike problem* [8, 27] and may provide new insight into how to improve its current solutions.

The *composition* of a string is the multiset of all symbols appearing in it. More precisely, the information provided by the composition of a string is its symbol alphabet accompanied by the number of times each symbol occurs in the string. Compositions of strings, often referred to as string *types*, may be suitably represented by *Parikh vectors*. For example, the composition of the string $BABCAA$ is the multiset $\{A, A, A, B, B, C\}$, denoted by $w(BABCAA) = A^3B^2C$ to indicate that the string consists of three A's, two B's, and one C. Given that the generative alphabet of the string is (A, B, C), its Parikh vector equals $(3, 2, 1)$.

To formulate the restricted reconstruction problem, observe that Assumption 1 implies that the composition, and hence also the length, of each substring can be determined from its mass. The second assumption implies that, ignoring small effects at the terminal ends of the sequence, all substrings of a given length should appear roughly the same number of times. For example, for cut probability p, the number of strings of length k would be proportional to $p^2(1-p)^{k-1}$. Since the substring lengths can be determined from the mass spectrometer measurements, their number of appearances can be normalized so that the composition of each substring appears exactly once. The problem then reduces to reconstructing a length-n sequence from the multiset of all of its $\binom{n+1}{2}$ substring compositions.

As an example, sequencing the string $ACAB$ would result in $\binom{4+1}{2} = 10$ substring compositions: A, A, B, C, AB, AC, AC, A^2C, ABC, and A^2BC. Note that for each substring, only the composition, and not the order of the symbols, is given, and that the compositions are provided along with their multiplicity, but not the location at which they start in the string. To reconstruct the original string $ACAB$ from its substring compositions, note that the compositions imply that it consists of two A's, a B, and a C, and that the two A's do not appear in consecutive order. Hence, the string equals $ABAC$, $ACAB$, $ABCA$, or their reversals. The substring composition A^2C implies that the original string is $ACAB$ or its reversal.

Clearly, a string and its reversal have the same composition multiset and hence cannot be distinguished. Therefore, we attempt to recover a string from its multiset composition only up to reversal.

As a final simplification, note that peptide and protein sequences are specified over an alphabet consisting of 20 letters representing the amino acids. However, for the theoretical results in the paper, we only consider reconstruction of binary strings from their composition multisets. In section 4, we show that whenever binary strings can be uniquely reconstructed, strings over all other alphabets can be uniquely reconstructed as well. Moreover, binary strings simplify the notation and make the theoretical exposition easier to follow.

2. Definition and results.

The *composition multiset* of a string $s = s_1 s_2 \dots s_n \in \{0{,}1\}^n$ is the multiset of multisets

$$\mathcal{S}_s \overset{\text{def}}{=} \{\{s_i, s_{i+1}, \dots, s_j\} : 1 \le i \le j \le n, \},$$

i.e., the multiset of compositions of all $\binom{n+1}{2}$ contiguous substrings of s. For example,

$$\mathcal{S}_{001} = \{0, 0, 1, 0^2, 01, 0^2 1\} \quad \text{and} \quad \mathcal{S}_{010} = \{0, 0, 1, 01, 01, 0^2 1\},$$

where we used the previously described shorthand notation $0^2 = 00$, and $0^2 1 =$ 001, or 010, or 100. Note that the number of substrings with a given composition is reflected in the multiset, although the locations of the substrings within the string are not known.

Two strings s and t are *equicomposable*, denoted $s \sim t$, if they have the same composition multisets. Equicomposability is clearly an equivalence relation. The *reversal* of a string $s = s_1 s_2 \dots s_n$ is the string $s^* \overset{\text{def}}{=} s_n s_{n-1} \dots s_1$. A string and its reversal are clearly equicomposable. We say that a string is *reconstructable* if it is equicomposable only with itself and its reversal, and hence may be determined up to reversal from its composition multiset. A string s is *nonreconstructable* or *confusable* if it is equicomposable with another string $t \ne s^*$. In this case, we also call s and t *confusable* strings.

Using computer simulations, we find that all binary strings of length at most 7 are reconstructable, and interestingly, for length 8, there are two confusable strings, neither of which is reconstructable. In particular, in section 5, we give a simple combinatorial argument to show that

$$01001101 \sim 01101001.$$

An extension of the example yields nonreconstructable binary strings of length n, whenever $n+1$ is a product of two integers, each at least 3. For example, we produce nonreconstructable strings of length 8, as $8+1 = 3 \cdot 3$, length 11, as $11+1 = 4 \cdot 3$, etc. This leads to the question as to whether all binary strings of the remaining lengths are reconstructable. Observe that these remaining lengths n are precisely those for which $n+1$ is either 8, a prime, or twice a prime. One of the results we prove is that all strings of such lengths are reconstructable.

To do so, in sections 6 and 7 we represent a binary string s by a *generating* bivariate polynomial $P_s \in \mathbb{Z}[x, y]$, and formally define the *reciprocal* P^* of the polynomial P. We show that two strings s and t are equicomposable if and only if

$$P_s P_s^* = P_t P_t^*.$$

We also consider the sets

$$E_s \overset{\text{def}}{=} \{t : t \sim s\} \tag{1}$$

consisting of all strings equicomposable with s, and the sets

$$\mathcal{P}_{E_s} \overset{\text{def}}{=} \{P_t : t \sim s\} \tag{2}$$

of the generating polynomials of all strings equicomposable with s. We show that these sets have a simple algebraic (polynomial) characterization, outlined in what follows. Let

$$P_s = P_1 P_2 \cdots P_k$$

be the prime factorization of P_s over $\mathbb{Z}[x, y]$. The equicomposable set of s can be expressed exactly and simply as

$$\mathcal{P}_{E_s} = \{\tilde{P}_1 \tilde{P}_2 \cdots \tilde{P}_k : \text{each } \tilde{P}_i \text{ is } P_i \text{ or } P_i^*\}.$$

In section 10, we use these results to show that, indeed, if $n + 1$ is 8, a prime, or twice a prime, all length-n binary strings are reconstructable. Along with the constructions of section 5, this establishes the exact lengths for which all strings are reconstructable.

In sections 11 and 12, we consider lengths for which reconstruction is not always unique. Let

$$E_n \overset{\text{def}}{=} \max\{|E_s| : s \in \{0,1\}^n\}$$

denote the largest number of mutually equicomposable n-bit strings. Since every string is equicomposable with its reversal, $E_n \geq 2$ for every $n \geq 2$, and from the previous discussion, $E_n \geq 4$ whenever $n + 1$ is a product of two integers ≥ 3.

Introducing a new interleaving sequence construction, we lower-bound E_n, and using the polynomial representation results and results on cyclotomic-polynomials, we also upper bound the same quantity. In particular, based on the unique-prime factorization of $n + 1 = 2^{e_0} p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$, we show that

$$2^{\lfloor \frac{e_0}{2} \rfloor + e_1 + e_2 + \cdots + e_k} \leq E_n \leq \min\{2^{(e_0+1)(e_1+1)\cdots(e_k+1)-1}, (n+1)^{1.23}\},$$

where $d(n+1) \overset{\text{def}}{=} (e_0 + 1)(e_1 + 1) \cdots (e_k + 1)$ is the number of distinct divisors of $n + 1$. We also show that when $n + 1$ is a prime power or twice a prime power, the lower bound is tight. For all $k \geq 1$,

$$E_{2^k - 1} = 2^{\lfloor \frac{k}{2} \rfloor},$$

and for primes $p \geq 3$,

$$E_{p^k - 1} = E_{2p^k - 1} = 2^k.$$

Plugging $p = 3$ above gives

$$E_n = (n+1)^{\log_3 2}.$$

We then proceed to describe a backtracking algorithm that reconstructs a string from its substring compositions. Using some well-known results on the behavior of random walks in high dimensions, we show that with high probability, the algorithm reconstructs a string in time $O(n^2 \log n)$ with high probability for any constant alphabet size ≥ 4. The complexity of our algorithm is significantly smaller than that of polynomial factorization algorithms, which comes at the cost of only having probabilistic performance guarantees. The algorithm can also reconstruct strings over alphabet sizes 2 and 3, although with weaker computational performance guarantees.

3. Relation to other work.

String reconstruction from multiset decompositions is a new problem, related to two types of known problems. Through its formulation, it is similar to other string-reconstruction problems, while through its mathematical analysis, it is closely related to the well-known turnpike problem.

Several string-reconstruction variations have been previously considered. Reconstruction of a string from a few substrings was considered in [19], and in the context of patterns matching, in the seminal paper [29]. Reconstruction of a string from its subsequences (not necessarily contiguous) was considered in [18, 11, 3, 30]. However, all the aforementioned contributions assume that the substrings or subsequences themselves, which include the *order of their symbols*, are given. By contrast, in our problem, for each substring we are given just the composition, and neither the order of the symbols within it nor the substring locations within the original string.

Note that four possible problems of reconstructing a string from its substrings may be formulated within the described settings. They differ by whether or not one is given (a) the order of the substrings in the string and (b) the order of the bits in each substring. All four problems are of interest when only some of the $\binom{n+1}{2}$ substrings are available. But when all substrings are given, knowing the order of either the substrings or their symbols clearly determines the original string. It is only when, as in the proposed mass-spectrometry application, neither the substring order nor the symbol order are provided that our reconstruction question arises.[1]

As already pointed out, in terms of the solutions and proof techniques, our problem is closely related to the turnpike problem, where the locations of n highway exits need to be recovered from the multiset of their $\binom{n}{2}$ interexit distances [27]. For example, the interexit distances 1, 2, 3, 3, 5, 6 correspond to exit locations 0, 1, 3, 6.

The turnpike problem originally arose in X-ray crystallography [23, 24] and has found many applications, including to DNA analysis [10]. It is also of independent theoretical interest as it has an algorithm whose run time is polynomial in the largest interexit distance, but it is not known whether it has an algorithm whose run time is polynomial in n, independently on the values of the distances. Several variations of the problem have been recently considered, e.g., [9, 5] and references therein. A related problem of characterizing strings that have the same *set*, instead of a multiset of compositions, is considered in [12].

String reconstruction from substring multisets is a combinatorial specialization of the turnpike problem [14, 5]. To see that this is the case, convert every string-reconstruction instance to a turnpike problem whose solution implies the original string as follows. Given an n-bit string reconstruction problem, replace each substring composition by an interexit distance obtained by replacing each 0 by 1 and each 1 by $n+1$ and summing the values. Then find the exit locations and replace back the bit values.

For example, to recover the string 1011 from the compositions 0, 1, 1, 1, 01, 01, 11, 01^2, 01^2, 01^3, replace every 0 by 1 and every 1 by $4+1=5$ to obtain the interexit distances 1, 5, 5, 5, 6, 6, 10, 11, 11, and 16. These interexit distances in turn correspond to the locations 0, 5, 6, 11, and 16. Writing the adjacent location differences 5, 1, 5, and 5, and converting a 1 back to 0, and a 5 back to 1 produces the original string 1011. Note that the mapping from 1 to $n+1$ is chosen to prevent spurious solutions.

This reduction shows that the string reconstruction problem may be solved in polynomial time [17], which is a useful observation but not the main focus of the

[1] Preliminary results on this problem were presented in [1, 2].

work. Instead, our scope is broader: we show that several questions unsolved for the turnpike problem can be answered for string reconstruction. Hence, in addition to the problem's intrinsic theoretical and practical value, its solution may provide useful new insights needed to analyze the general turnpike problem. For example, some of the more important unsolved questions about the turnpike problem concern the number of solutions a given instance may have [27]. A turnpike problem with $n \geq 6$ exits may have multiple solutions, but their largest possible number is not known for any such n. By contrast, for string reconstruction, we show that when $n+1$ is 8, a prime, or twice a prime, reconstruction is always unique. We also determine the exact number of maximal reconstructions whenever $n+1$ is a prime power or twice a prime power and conjecture that the same formula holds for all remaining string lengths.

We also note that some of the polynomial techniques we use are related to those applied to the turnpike problems [26, 27]. Yet others, such as the bivariate polynomial formulation and relation to cyclotomic polynomials, seem new.

4. Binary and higher order alphabets.

We start our analysis by showing that if binary strings are reconstructable, then strings of over higher alphabet sizes are also reconstructable. In fact, one can use the algorithmic methods for binary reconstruction to achieve this task.

LEMMA 1. *If binary strings of a certain length are reconstructable, then strings of the same length over any finite alphabet are reconstructable.*

Proof. We first consider an example. Suppose that we want to reconstruct a string from the multiset $\{A, A, B, C, AB, AC, AC, A^2C, ABC, A^2BC\}$. We replace one of the symbols, say, A, by 1 and all other symbols, i.e., B and C, by 0. This yields the compositions 0, 0, 1, 1, 01, 01, 01, 0^21, 01^2, 0^21^2. Since the binary string is reconstructable, we recover that the original string equals 1010 and that consequently, A appears in the first and third position (or second and fourth, when ordered from the opposite end). We then replace the symbols A and B with 1, and the symbol C with 0. This gives the position of the symbols B relative to the positions of the A's and also yields the locations of C.

For general strings, we follow the same approach. We decide a (lexical) ordering of the symbols appearing in the string, say, $A_1, \ldots, A_m$. At stage $i < m$, we replace the symbols $A_1, \ldots, A_i$ with 1 and others symbols with 0. We are able to obtain the relative positions of one symbol at a time and can therefore determine the complete string at the end of this sequential procedure. $\square$

Hence, for the theoretical sections of the paper, we only consider the reconstruction of binary strings from their composition multisets. When we design randomized reconstruction algorithms, we observe that larger alphabets make the problem actually much easier to handle.

5. Simple confusions.

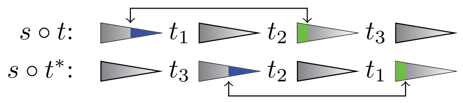
An exhaustive computer search shows that all binary strings of length at most 7 are reconstructable. For length 8, we exhibit two strings, 01001101 and 01101001, which are confusable and are not reversals of each other. To see that the strings are confusable, note that they can be parsed as **01** 0 **01** 1 **01** and **01** 1 **01** 0 **01**. Both have a common substring **01**, which is interleaved with 0 1 in the first string and with its reversal 1 0 in the second string. This prompts us to define the notion of string interleaving.

The *interleaving* of a string s with the bits of $t = t_1 \ldots t_m$ is the string $s \circ t \overset{\text{def}}{=} s\, t_1\, s\, t_2 \cdots t_m\, s$. The first string in the above example may hence be written as $01 \circ 01$, while the second string may be written as $01 \circ 10$. A similar notation is used when interleaving multiple strings.

For our subsequent discussion, recall that t^* represents the reversal of t and that $\sim$ indicates that two strings are equicomposable.

LEMMA 2. *For any s and t, $s \circ t \sim s \circ t^*$*

Proof. We first prove the result when t has length 3. The interleaving $s \circ t$ generated in this case is schematically depicted in the figure below, where each triangle represents the string s.



There is a bijection from the substrings of $s \circ t$ to the substrings of $s \circ t^*$ that preserves compositions. For example, as illustrated in the figure, the substring in $s \circ t$ consisting of a tail of s (blue), t_1, s, t_2, and a head of s (green) has the same composition as the substring in $s \circ t^*$ consisting of the same tail of s (blue), t_2, s, t_1, and the same head of s (green). Thus, $s \circ t$ and $s \circ t^*$ have the same multiset of compositions and hence are equicomposable.

In general, suppose that s and t are strings of lengths m and m', respectively. We present a general bijection as done for $m' = 3$. It is helpful to have the above figure in mind when dealing with strings t of length longer than 3. Let r and r' denote $s \circ t$ and $s \circ t^*$, respectively. Consider a substring of r. If r does not contain elements from the string t, then we map the substring to the same location in r'. In this case, we have an identity map which preserves compositions. For substrings r that contain some symbols $t_i, t_{i+1}, \ldots, t_j$ from t, we consider the substring of r' that contains the same symbols in reversed order $t_j, t_{j-1}, \ldots, t_i$. We then identify the number of symbols preceding t_j in r' and the same number of symbols preceding t_i in r. We do the same for symbols trailing t_i in r' and t_j in r. This substring map preserves compositions and hence proves the lemma. $\square$

If the lengths of s and t are m and m', respectively, then the length of $s \circ t$ equals $n = (m+1)(m'+1) - 1$. If $m \geq 2$, $m' \geq 2$, we can always choose a nonpalindromic t, namely, a t such that $t \neq t^*$, ensuring that $s \circ t \neq s \circ t^*$, and a nonpalindromic s, such that $s \neq s^*$, ensuring that $s \circ t \neq (s \circ t^*)^*$. In particular, we can choose both s and t to be strings of the form $100 \ldots 0$, where the number of 0's determines the lengths of the sequences. As a corollary, we then obtain the result below.

COROLLARY 3. *Whenever $n+1$ is a product of two integers, each ≥ 3, there exist confusable n-bit strings.*

6. Polynomial representation. The previous section described some confusable strings obtained via interleaving. To further characterize confusability, we represent strings as polynomials. A similar representation has been used for the turnpike problem [26, 27], but the polynomials in the former work are univariate, whereas bivariate polynomials are better suited for string reconstruction.

Polynomial representations are used to show that the strings equicomposable with a string s can be determined by the prime factorization of the polynomial representing s and that s can be reconstructed from its composition multiset via polynomial factorization.

All polynomials and factorizations in the paper are over $\mathbb{Z}[x, y]$. A polynomial whose nonzero coefficients are all equal to 1 is termed a *0-1* polynomial. We say that $x^a y^b$ is a *monomial* with *x-degree* a, *y-degree* b, and *total degree* $a + b$. With a slight abuse of terminology, we refer to terms of the form $x^a y^b$ as monomials even when the exponents a and b are negative. The x-, y-, and *total-degrees* of a polynomial are the highest corresponding degrees of the monomials. For example, for $x^2y + xy^3$, the x-degree, y-degree, and total degree of the polynomial are 2, 3, and 4, respectively.

When representing strings by polynomials, 0 is denoted by x and 1 by y. Also, a_i denotes the number of 0's in the first i bits of a string and a denotes their total number, while b_i and b represent the same counts for 1's. By definition, $a_0 = 0$ and $b_0 = 0$ for an empty string. For example, 01011 has $a_3 = a = 2$, $b_3 = 1$, and $b = 3$.

The *generating polynomial* of a binary string $s = s_1s_2\cdots s_n$ is defined as

$$P_s(x,y) \overset{\text{def}}{=} \sum_{i=0}^{n} x^{a_i}y^{b_i} = \sum_{i=0}^{n} x^{a_i}y^{i-a_i}. \tag{3}$$

For example,

$$P_{0100}(x,y) = 1 + x + xy + x^2y + x^3y.$$

Generating polynomials of n-bit strings are characterized by the following sufficient properties:

G1. The polynomials are 0-1.

G2. The polynomials have $n+1$ monomials, exactly one of each total degree $0, 1, \ldots, n$.

G3. For all $1 \le i \le n$, the ratio of the monomials of total degrees i and $i-1$ is either x or y.

By the definition of a_i and b_i, for $i \ge 1$,

- $a_{i+1} = a_i + 1$ and $b_{i+1} = b_i$ when the $(i+1)th$ bit is a 0, and
- $a_{i+1} = a_i$ and $b_{i+1} = b_i + 1$ when the $(i+1)th$ bit is a 1.

Property G1 holds by virtue of the definition of P_s. Properties G2 and G3 are true since each monomial is obtained by multiplying the monomial of total-degree by one smaller either with x or y.

For example, the polynomial $1 + x + xy + x^2y + x^3y$ is a string generating polynomial, while $1 - x$, $1 + x^2$, and $1 + x + y^2$ are not.

In a similar vein, we may represent a composition 0^a1^b by the monomial x^ay^b and define $\mathcal{S}_s(x,y)$ to be the sum of all monomials corresponding to the substring compositions of s. For example,

$$\mathcal{S}_{0100}(x,y) = 3x + y + x^2 + 2xy + 2x^2y + x^3y,$$

which is the sum of the monomial representations of the compositions in the multiset $\mathcal{S}_{0100} = \{0, 0, 0, 1, 0^2, 01, 01, 0^21, 0^21, 0^31\}$.

We next provide a simple functional connection between the generating polynomial of a string and the polynomial corresponding to its composition multiset.

LEMMA 4.

$$P_s(x,y)P_s\left(\frac{1}{x},\frac{1}{y}\right) = n + 1 + \mathcal{S}_s(x,y) + \mathcal{S}_s\left(\frac{1}{x},\frac{1}{y}\right).$$

Proof. By (3) and the properties G1–G3, the product on the left-hand side is precisely the sum of monomials of the form $x^{a_j}y^{b_j} \cdot (x^{a_i}y^{b_i})^{-1}$ for $1 \le i,j \le n$. Furthermore,

$$x^{a_j}y^{b_j} \cdot (x^{a_i}y^{b_i})^{-1} = x^{a_j-a_i}y^{b_j-b_i}$$

represents the composition of the substring $s_{i+1}\ldots s_j$ when $i < j$, and the reciprocal of the composition of $s_{i+1}\ldots s_j$ when $i > j$. Therefore, the monomials with positive degree correspond to $\mathcal{S}_s(x,y)$, while the monomials with negative degree correspond to its corresponding reciprocal polynomial. ▯

For our running example, the lemma proves the following expansion:

$$\begin{aligned}&P_{0100}(x,y)P_{0100}\left(\frac{1}{x},\frac{1}{y}\right)\\&= 5+3x+y+x^2+2xy+2x^2y+x^3y+\frac{3}{x}+\frac{1}{y}+\frac{1}{x^2}+\frac{2}{xy}+\frac{2}{x^2y}+\frac{1}{x^3y}\\&= 5+\mathcal{S}_{0100}(x,y)+\mathcal{S}_{0100}\left(\frac{1}{x},\frac{1}{y}\right).\end{aligned}$$

To reconstruct a string from a composition multiset $\mathcal{S}$, we need to find all strings s whose generating polynomials P_s satisfy Lemma 4. To this end, we also have the following simple lemma.

LEMMA 5. *Strings s and t are equicomposable if and only if*

$$P_s(x,y)P_s\left(\frac{1}{x},\frac{1}{y}\right) = P_t(x,y)P_t\left(\frac{1}{x},\frac{1}{y}\right). \tag{4}$$

Proof. By the 1-1 correspondence between $\mathcal{S}_s$ and $\mathcal{S}_s(x,y)$, it follows that $s \sim t$ if and only if $\mathcal{S}_s(x,y)=\mathcal{S}_t(x,y)$. The result then follows from Lemma 4. ▯

We next show that for a string s, an equation like (4) holds only for generating polynomials. The approach we pursue is akin to that of [26, 27], with additional verification that the polynomial obtained satisfies all the properties of generating polynomials.

LEMMA 6. *If P is a generating polynomial, and $Q \in \mathbb{Z}[x,y]$ with $Q(0,0) > 0$ is such that*

$$P(x,y)P\left(\frac{1}{x},\frac{1}{y}\right) = Q(x,y)Q\left(\frac{1}{x},\frac{1}{y}\right), \tag{5}$$

then $Q(x,y)$ is a generating polynomial as well.

Proof. We first show that Q satisfies G1.

G1. Evaluating (5) at $x=y=1$, we obtain $P(1,1)^2 = Q(1,1)^2$, which implies that $P(1,1) = \pm Q(1,1)$. Suppose that the correct sign is "+." Let $P(x,y) = \sum_{i,j} p_{i,j}x^iy^j$ and $Q(x,y) = \sum_{i,j} q_{i,j}x^iy^j$, where $p_{i,j}, q_{i,j} \neq 0$. Then,

$$\sum_{i,j} p_{i,j} = \sum_{i,j} q_{i,j}.$$

Comparing the constant terms on both sides of (5) gives

$$\sum_{i,j} p_{i,j}^2 = \sum_{i,j} q_{i,j}^2.$$

Subtracting the last from the next to last expression shows that

$$\sum_{i,j} p_{i,j}(1-p_{i,j}) = \sum_{i,j} q_{i,j}(1-q_{i,j}).$$

Since P is a generating polynomial, by G1 all $p_{i,j}$ are 1, and the left and hence the right side is identically equal to zero. For all integers, i, $i(1-i) \le 0$ with equality if and only if i is either 0 or 1. Since all $q_{i,j}$ are nonzero integers, they must equal 1. The case of a negative sign "$-$" is handled similarly.

G2. Each monomial in $P(x,y)P(1/x,1/y)$ is of the form $x^{a_j-a_i}y^{b_j-b_i}$. Thus, the exponents of x and y cannot have different signs in the polynomial product. The polynomial Q cannot have two monomials of the same total degree because their corresponding product in $Q(x,y)Q(1/x,1/y)$ would have a monomial with x- and y-degrees of opposite signs, which by G1 do not get cancelled by another product term. Let $0 = d_0 < d_1 < \ldots < d_n$ be the degrees of the different monomials in Q. The largest degree monomial in (5) on the left-hand side is $x^{a_n}y^{n-a_n}$ of degree n. The largest degree on the right-hand side is $d_n - d_0 = d_n$. So $d_i = d_0 + i = i$, which proves property G2.

G3. Consider the linear terms (of the form constant times x or y) in (5). For any 0-1 polynomial $F(x,y)$, the sum of the coefficients of the linear terms in $F(x,y)F(1/x,1/y)$ is the number of pairs of monomials in F whose ratio is x or y. The polynomial P has n such pairs, and hence so does Q. By G2, Q has $n+1$ monomials, one of each total degree $0, 1, \ldots, n$. Therefore, the ratio between any two "consecutive" monomials must equal x or y. $\square$

Recall E_s and $\mathcal{P}_{E_s}$ defined (1) and (2).

THEOREM 7.

$$\mathcal{P}_{E_s} = \left\{ P(x,y) \in \mathbb{Z}[x,y] : P(0,0) > 0, P(x,y)P\left(\frac{1}{x},\frac{1}{y}\right) = P_s(x,y)P_s\left(\frac{1}{x},\frac{1}{y}\right) \right\}.$$

7. Reciprocal polynomials. To apply existing results on polynomial factorization, we relate $P(1/x,1/y)$ to a standard polynomial form known as the *the reciprocal* polynomial of P. Let $\deg_x P$ and $\deg_y P$ be the highest x- and y-degrees of a bivariate polynomial P. The *reciprocal* of P is the polynomial

$$P^*(x,y) \overset{\text{def}}{=} x^{\deg_x P}y^{\deg_y P}P\left(\frac{1}{x},\frac{1}{y}\right).$$

For example,

$$(y + 3xy - 2y^2)^* = xy^2\left(\frac{1}{y} + \frac{3}{xy} - \frac{2}{y^2}\right) = xy + 3y - 2x.$$

We make use of the following properties of reciprocal polynomials:

R1. $\deg_x P^* \le \deg_x P$.

R2. The reciprocal of the product is the product of the reciprocals: $(P_1P_2)^* = P_1^*P_2^*$.

R3. The reciprocal of the generating polynomial of s generates the reversal of the string s: $P_s^* = P_{s^*}$.

The monomial in P^* with $x-$degree d in P is $\deg_x P - d \le \deg_x P$, which proves claim R1. Property R2 follows from a simple expansion and matching of monomials in the two polynomials. Recall that a_i and b_i denote the number of 0's and 1's within the first i positions of s . Since s and s^* are reversals of each other, they have the same number of 0's and 1's (say, a and b). The number of 0's and 1's in the first $n-i$ symbols of s^* is $a - a_i$ and $b - b_i$, respectively, which leads to the monomial $x^ay^b \cdot \frac{1}{x^{a_i}}\frac{1}{y^{b_i}}$ of $x^ay^bP_s(1/x,1/y)$. This proves property R3.

For example,

$$P_{0100}^*(x,y) = (1 + x + xy + x^2y + x^3y)^* = 1 + x + x^2 + x^2y + x^3y = P_{0010}(x,y).$$

These properties imply polynomial formulations of Lemmas 5 and 6 and Theorem 7. The proofs of the results are omitted, as they closely follow those of previously established counterparts.

LEMMA 8. *Strings s and t are equicomposable if and only if*

$$P_sP_s^* = P_tP_t^*.$$

LEMMA 9. *If P is a generating polynomial and $Q \in \mathbb{Z}[x,y]$ with $Q(0,0) > 0$ is such that*

$$PP^* = QQ^*,$$

then Q is also a generating polynomial.

THEOREM 10.

$$\mathcal{P}_{E_s} = \{P(x,y) \in \mathbb{Z}[x,y] : P(x,y) > 0, P(x,y)P^*(x,y) = P_s(x,y)P_s^*(x,y)\}.$$

So far, we have studied products of generating polynomials and their reciprocals. The next lemma addresses the question of factoring generating polynomials.

A polynomial $P(x,y)$ over is *reducible* over $\mathbb{Z}[x,y]$ if it can be represented as a product $P_1(x,y) \cdot P_2(x,y)$ for nonconstant polynomials P_1 and P_2, both in $\mathbb{Z}[x,y]$, and is *irreducible* otherwise.

LEMMA 11. *Two strings satisfy $s \sim t$ if and only if for some $A, B \in \mathbb{Z}[x,y]$,*

$$P_s = AB \qquad \text{and} \qquad P_t = AB^*.$$

Proof. Let $A = \gcd(P_s, P_t)$ be the greatest common divisor polynomial of P_s and P_t over $\mathbb{Z}[x,y]$. Then, we may write $P_s = AB$ and $P_t = AC$ for relatively prime polynomials B and C. We show next that $C = B^*$.

By Lemma 8, $P_sP_s^* = P_tP_t^*$. By R2, $P_s^* = A^*B^*$ and $P_t^* = A^*C^*$, and hence $BB^* = CC^*$. Since B and C are relatively prime, C has to divide B^*. But since P_s and P_t have the same total degree, B, B^*, C, and C^*, all have the same total degree. And since P_s and P_t are generating polynomials, the four polynomials have constant term 1. Hence, $C = B^*$.

Conversely, if $P_s = AB$ and $P_t = AB^*$, then it is easy to see that $P_sP_s^* = (AB)(AB)^* = AA^*BB^* = AB^*(AB^*)^* = P_tP_t^*$. □

For example, we saw that the 8-bit strings $s = 01001101$ and $t = 01101001$ are confusable. Indeed $P_s(x,y) = (1+x+xy)(1+x^2y+x^3y^3)$ and $P_t(x,y) = (1+x+xy)(1+xy^2+x^3y^3) = (1+x+xy)(1+x^2y+x^3y^3)^*$. Here, A is a generating polynomial, but this is not always the case. The 23-bit strings 01000101010000100011001 and 01010100010000110010001 are confusable, but the divisors $A = 1+y+xy+xy^2+x^4y^8+x^4y^9+x^4y^{10}+x^5y^{10}$ and $B = 1+xy^3+x^3y^5$ are not generating polynomials. In the former examples, A and B are 0-1 polynomials, but even this claim does not always hold true. The string 01001001001 can be factored as $(1+x+x^3y-x^4y^2+x^3y^3+x^5y^3+x^5y^4)(1+xy)$, where the first factor has a negative coefficient. We will return to the structure of these factors in section 11.

The previous results imply a very simple expression for E_s, the set of strings equicomposable with s. Consider the prime factorization of P_s into (irreducible) polynomials

$$P_s = P_1P_2\cdots P_k.$$

The next theorem asserts that every product of the P_i's or their reciprocals is a generating polynomial and that the resulting generating polynomials exactly correspond to all strings equicomposable with s.

THEOREM 12. *For every string s,*

$$\mathcal{P}_{E_s} = \{\tilde{P}_1\tilde{P}_2\cdots\tilde{P}_k : \text{each } \tilde{P}_i \text{ is } P_i \text{ or } P_i^*\}.$$

Proof. If $P = \tilde{P}_1\tilde{P}_2\cdots\tilde{P}_k$, where each $\tilde{P}_i$ is either P_i or P_i^*, let A be the product of the nonreciprocated polynomials P_i and let B be the product of the reciprocated polynomials P_i (i.e., P_i^*). Then, $P_s = AB$ and $P = AB^*$. Hence, $P_sP_s^* = PP^*$, and by Lemma 9, $P = P_t$ for some string t. But then $P_sP_s^* = P_tP_t^*$, and by Lemma 8, $t \in E_s$.

Conversely, if $t \in E_s$, then by Lemma 11, $P_s = AB$ while $P_t = AB^*$. The prime factorizations $A = P_1P_2\cdots P_j$ and $B = P_{j+1}\cdots P_k$ yield the prime factorizations $P_s = P_1P_2\cdots P_k$ and $P_t = P_1P_2\cdots P_jP_{j+1}^*\cdots P_k^*$. Therefore, P_t can be written as $\tilde{P}_1\tilde{P}_2\cdots\tilde{P}_k$. ☐

The theorem provides a simple formula for the number of strings equicomposable with any given string s. To derive the formula, we observe that a polynomial is *palindromic* if it equals its reciprocal. For example, $1 + x + xy + xy^2 + x^2y^2$ is palindromic, while $1 + x + xy$ ($(1+x+xy)^* = 1+y+xy$) is not. Let ν_s be the number of nonpalindromic terms in the prime factorization of $P_s(x, y)$.

COROLLARY 13. *For every string s,*

$$|E_s| \le 2^{\nu_s}.$$

In particular, if $\nu_s = 1$, then s is reconstructable.

Proof. The first part follows trivially from the theorem. The second part follows as $\nu_s = 0$ implies $E_s = \{s\}$, and $\nu_s = 1$ implies $E_s = \{s, s^*\}$. ☐

In section 9, we return to this result and show that the inequality can be replaced with equality.

8. Cyclotomic polynomials.

Theorem 12 characterizes equicomposability in terms of prime factorization of generating polynomials. Factoring bivariate polynomials is hard in general, but the current analysis is simplified by the fact that any factorization of a generating polynomial $P(x, y)$ implies a factorization of the univariate polynomial $P(x, x)$ obtained by evaluating $P(x, y)$ at $y = x$. From properties G1 and G2 of generating polynomials, if $P(x, y)$ generates an n-bit string, then

$$P(x,x) = 1 + x + x^2 + \cdots + x^n = \frac{x^{n+1}-1}{x-1}.$$

Factorizations of $x^n - 1$, and hence of $x^{n+1} - 1$, have been studied extensively, and for completeness we provide a small sampling of results from the literature (see e.g., [15, 21]) needed for our analysis of confusability.

The n complex numbers $\omega_1 = e^{\frac{2\pi i}{n}}, \omega_2 = e^{2\frac{2\pi i}{n}}, \ldots, \omega_n = e^{n\frac{2\pi i}{n}} = 1$, whose nth powers are 1, are referred to as the nth *roots of unity.* They are exactly the roots of $x^n - 1$, and therefore

$$x^n - 1 = \prod_{i=1}^{n}(x - \omega_i).$$

The prime factorization of $x^n - 1$ over $\mathbb{Z}[x]$ partitions the n degree-one polynomials into groups, each group multiplying to an irreducible polynomial over $\mathbb{Z}[x]$.

For a positive integer d, a dth root of unity is *primitive* if none of its positive powers $\le d-1$ equals 1. Examples of primitive roots are listed in the table below.

d	1	2	3	4	5	6
primitive dth roots of unity	1	-1	$e^{\pm 2\pi i/3}$	$\pm i$	$e^{2\pi i j/5}, j = 1, \ldots, 4$	$e^{\pm 2\pi i/6}$

The dth *cyclotomic polynomial* is defined as

$$\Phi_d(x) \overset{\text{def}}{=} \prod_{\omega} (x - \omega),$$

where the product ranges over all primitive dth roots of unity ω. For example,

$$\begin{aligned}
\Phi_1(x) &= x - 1,\\
\Phi_2(x) &= x + 1,\\
\Phi_3(x) &= (x - e^{2\pi i/3})(x - e^{-2\pi i/3}) = x^2 + x + 1,\\
\Phi_4(x) &= (x - i)(x + i) = x^2 + 1,\\
\Phi_5(x) &= \prod_{j=1}^{4} (x - e^{\frac{2\pi i j}{5}}) = x^4 + x^3 + x^2 + x + 1,\\
\Phi_6(x) &= (x - e^{2\pi i/6})(x - e^{-2\pi i/6}) = x^2 - x + 1.
\end{aligned}$$

Every nth root of unity is a primitive dth root of unity for some $1 \le d \le n$ such that $d \mid n$, where $d \mid n$ means d is a divisor of n. Hence, for every n,

$$x^n - 1 = \prod_{d \mid n} \Phi_d(x). \tag{6}$$

For example,

$$\begin{aligned}
x^2 - 1 &= (x - 1)(x + 1),\\
x^3 - 1 &= (x - 1)(x^2 + x + 1),\\
x^4 - 1 &= (x - 1)(x + 1)(x^2 + 1),\\
x^5 - 1 &= (x - 1)(x^4 + x^3 + x^2 + 1),\\
x^6 - 1 &= (x - 1)(x + 1)(x^2 + x + 1)(x^2 - x + 1).
\end{aligned}$$

It is easy to verify that cyclotomic polynomials have integer coefficients. Comparing coefficients shows that if $a(x), b(x) \in \mathbb{Q}[x]$ and $a(x)b(x) \in \mathbb{Z}[x]$, then $a(x), b(x) \in \mathbb{Z}[x]$. By induction on n, assume that for all $d < n$ with $d \mid n$, $\Phi_d(x) \in \mathbb{Z}[x]$. Then (6) implies that

$$\Phi_n(x) = \frac{x^n - 1}{\prod\limits_{d<n,\, d \mid n} \Phi_d(x)}.$$

Since different cyclotomic polynomials are relatively prime (as no two cyclotomic polynomials share a root), $\Phi_n(x) \in \mathbb{Q}[x]$, and by the fact mentioned above, $\Phi_n(x) \in \mathbb{Z}[x]$. Gauss showed, e.g., [15], that the cyclotomic polynomials are irreducible. The irreducibility proof of general cyclotomic polynomials is somewhat involved. However, we will mostly need to factor $x^p - 1$ and $x^{2p} - 1$ for a prime p. The factorizations of these two polynomials require establishing the irreducibility of $\Phi_p(x)$ and $\Phi_{2p}(x)$, which follow easily from *Eisenstein's criterion.*

LEMMA 14 (Eisenstein's criterion [13]). *Let $P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0$. If some prime $p \mid a_0, a_1, \ldots, a_{n-1}$ but $p \nmid a_n$ and $p^2 \nmid a_0$, then $P(x)$ is irreducible over $\mathbb{Z}$.*

Proof. Suppose toward a contradiction that $P(x) = (b_k x^k + \cdots + b_0)(c_\ell x^\ell + \cdots + c_0)$, where $k, \ell \geq 1$ and $b_k, c_\ell \neq 0$. Since p divides $a_0 = b_0 c_0$ but p^2 does not, p divides exactly one of b_0 and c_0. Without loss of generality, assume that $p | b_0$. Since p does not divide $a_n = b_k c_\ell$, there is a smallest index $i < k < n$ such that p does not divide b_i. It follows that $p \mid a_i, b_0, b_1, \ldots, b_{i-1}$ and $p \nmid c_0, b_i$, contradicting $a_i = b_i c_0 + b_{i-1}c_1 + \cdots + b_0 c_i$. ∎

COROLLARY 15. *For every prime p,*

$$\Phi_p(x) = 1 + x + x^2 + \cdots + x^{p-1}$$

is irreducible over $\mathbb{Z}$.

Proof. Substituting x by $x+1$, we may write

$$\Phi_p(x+1) = \frac{(x+1)^p - 1}{x} = \binom{p}{1} + \binom{p}{2}x + \cdots + \binom{p}{p}x^{p-1}.$$

Since p is prime, it divides $\binom{p}{1}, \binom{p}{2}, \ldots, \binom{p}{p-1}$ but not $\binom{p}{p}$. Therefore, p divides all coefficients except for the leading coefficient. Furthermore, p^2 does not divide the constant coefficient $\binom{p}{1}$. By Eisenstein's criterion, $\Phi_p(x+1)$ and therefore $\Phi_p(x)$ are irreducible over $\mathbb{Z}$. ∎

Note that $\Phi_{2p}(x) = \Phi_p(-x)$ for odd primes p and is therefore irreducible.

LEMMA 16. *$\Phi_d(x)$ is palindromic for all $d \geq 2$.*

Proof. For any polynomial $P(x)$, the roots of $P^*(x)$ over $\mathbb{C}$ are exactly the reciprocals of the roots of $P(x)$, and the constant coefficient of $P^*(x)$ is the leading coefficient of $P(x)$. Since the reciprocal polynomial always exists, and the roots and constant coefficient specify a polynomial, these two conditions are also sufficient for a polynomial to be the reciprocal of $P(x)$.

For every d, the roots of $\Phi_d(x)$ are on the unit circle, and hence the reciprocal of a root is also its complex conjugate. Furthermore, since $\Phi_d(x)$ has real coefficients, every root appears with its complex conjugate, and hence the reciprocals of the roots are roots too. Finally, for $d \geq 2$, $\Phi_d(x)$ has the same leading and constant coefficient, namely, 1, and hence it is its own reciprocal. ∎

9. The size of confusable sets.

The next lemma shows that in any factorization of a generating polynomial, no two terms are reciprocal of each other. Our proof is significantly simpler than the one used in [27] for the general turnpike problem.

LEMMA 17. *Generating polynomials cannot have two mutually reciprocal factors.*

Proof. Let $P(x,y)$ generate an n-bit string. If $A(x,y)A^*(x,y)$ divides $P(x,y)$, then $A(x,x)A^*(x,x) = A^2(x,x)$ divides $P(x,x) = 1 + x + x^2 + \cdots + x^n$. Clearly, $A^2(x,x)$ has roots of multiplicity two over $\mathbb{C}$. The roots of $1 + x + \cdots + x^n$ are all distinct, which implies that it has no roots of multiplicity larger than one. This leads to a contradiction that establishes the claimed result. ∎

If follows that Corollary 13 holds with equality.

Corollary 18. *For every string s,*

$$|E_s| = 2^{\nu_s},$$

and s is reconstructable if and only if $\nu_s \le 1$.

For example, if $s = 01001101$, its generating polynomial has irreducible factorization $P_s(x,y) = (1+x+xy)(1+x^2y+x^3y^3)$. Hence, $\nu_s = 2$, and there are exactly four strings confusable with s, $E_{01001101} = \{01001101, 01101001, 10110010, 10010110\}$.

10. Unique reconstruction for prime-related lengths. From Corollary 3, we see that whenever $n+1$ is a product of two integers ≥ 3, there exist confusable n-bit strings. The lengths n remaining undecided regarding reconstructability are those for which $n+1$ is a prime, twice a prime, or equal to 8. We use the polynomial representation to show that for all these lengths, reconstruction may always be performed uniquely.

While factorization and irreducibility questions of $P(x,y)$ are related to those of $P(x,x)$, the two are not equivalent. Irreducibility of $P(x,y)$ *does not imply* irreducibility of $P(x,x)$. For example, $P(x,y) = 2x^2 - y^2 = (\sqrt{2}x+y)(\sqrt{2}x-y)$ is irreducible over $\mathbb{Z}$, yet $P(x,x) = x^2$ factors as $x \cdot x$ and is not irreducible.

Similarly, in the implication direction we need, irreducibility of $P(x,x)$ does not generally imply irreducibility of $P(x,y)$ either. For example, $P(x,y) = (x-y+1)(x+1)$ is not irreducible even though $P(x,x) = x+1$ is irreducible. Yet, as the next two lemmas show, for generating polynomials, and more generally whenever $\deg P(x,y) = \deg P(x,x)$, either both $P(x,x)$ and $P(x,y)$ are irreducible or both polynomials are reducible.

Lemma 19. *Any factor of a generating polynomial has a single monomial of highest total degree.*

Proof. Let $\widehat{P}(x,y)$ be the sum of the highest-total-degree monomials in a polynomial $P(x,y)$, e.g., if $P(x,y) = x^3 + x^2y^2 + y^4$, then $\widehat{P}(x,y) = x^2y^2 + y^4$. If $A(x,y)$ is a factor of $P(x,y)$, then $\widehat{A}(x,y)$ divides $\widehat{P}(x,y)$, and if $P(x,y)$ is also generating, then $\widehat{P}(x,y)$ consists of a single monomial, and hence so does its factor $\widehat{A}(x,y)$. $\square$

Lemma 20. *The number of terms in the prime factorization of a generating polynomial $P(x,y)$ is at most the corresponding number for $P(x,x)$. Hence, if $P(x,x)$ is irreducible, then so is $P(x,y)$.*

Proof. Let $P_1(x,y)P_2(x,y)\cdots P_k(x,y)$ be the prime factorization of $P(x,y)$. Then $P_1(x,x)P_2(x,x)\cdots P_k(x,x)$ is a factorization of $P(x,x)$ into polynomials that by Lemma 19 are nonconstant. In particular, this uses that $\deg P(x,y) = \deg P(x,x)$ for a generating polynomial P. $\square$

We are now ready to state the main results of the paper.

Theorem 21. *All strings of length 7 are reconstructable.*

Proof. The proof is given in Appendix A. The result was also independently verified via computer search. $\square$

Theorem 22. *All strings of length one less than a prime are reconstructable.*

Proof. If s is $(p-1)$-bits long, then by the properties of generating polynomials,

$$P_s(x,x) = 1 + x + x^2 + \ldots + x^{p-1}.$$

By Corollary 15, $P_s(x,x)$ is irreducible. By Lemma 20, $P_s(x,y)$ is irreducible. By Corollary 13, s is reconstructable. $\square$

For example, for $n = 2$, both $P_{00}(x,y) = 1+x+x^2$ and $P_{01}(x,y) = 1+x+xy$, and their symmetric versions, $P_{10}(x,y)$ and $P_{11}(x,y)$, are irreducible.

To prove the result for $n+1$ equal twice a prime, we find the following lemma useful.

LEMMA 23. *If s is confusable, then $P_s = AB$ for some nonpalindromic polynomials A and B.*

Proof. Let s be confusable with t. By Lemma 11, $P_s = AB$ and $P_t = AB^*$ for some A and B. If B is palindromic, then $P_s = AB = AB^* = P_t$, namely, $s = t$, while if A is palindromic, then $P_s^* = A^*B^* = AB^* = P_t$, namely, $s = t^*$. ◻

THEOREM 24. *All strings of length one less than twice a prime are reconstructable.*

Proof. Let s have length n one less than twice a prime. We show that in any factorization

$$P_s(x,y) = f(x,y)g(x,y),$$

at least one of the two polynomials $f(x,y)$ and $g(x,y)$ is palindromic. By Lemma 23, s is reconstructable.

Eisenstein's criterion of Lemma 14 (also [21, 13]) implies the prime factorization

$$\begin{aligned} P_s(x,x) &= 1 + x + x^2 + \cdots + x^{2p-1} \\ &= (1+x)(1 - x + x^2 - \cdots + x^{p-1})(1 + x + \cdots + x^{p-1}). \end{aligned}$$

Hence, there are only three possible factorizations of $P_s(x,y)$ into two factors:

Case 1. $f(x,x) = 1+x$ and $g(x,x) = (1-x+x^2-\cdots+x^{p-1})(1+x+\cdots+x^{p-1})$.

Case 2. $f(x,x) = 1+x+\cdots+x^{p-1}$ and $g(x,x) = (1+x)(1-x+x^2-\cdots+x^{p-1}) = 1+x^p$.

Case 3. $f(x,x) = 1-x+\cdots+x^{p-1}$ and $g(x,x) = (1+x)(1+x+\cdots+x^{p-1})$.

We show that in all three cases, at least one of the polynomials $f(x,y)$ and $g(x,y)$ is palindromic.

Case 1. By Lemma 19, $f(x,y)$ must be either $1+x$ or $1+y$, and both are palindromic.

Case 2. Let $s = s_1 s_2 \dots s_n$. Then $P(x,y) \overset{\text{def}}{=} P_s(x,y) = \sum_{i=0}^{2p-1} x^{a_i} y^{i-a_i}$. Denote $\dfrac{\partial P}{\partial x}(x,y)|_{y=x}$ by P'_x, in which case

$$P'_x = \sum_{i=1}^{2p-1} a_i x^{i-1}.$$

Therefore,

$$\frac{\partial P}{\partial x}(x,y) = f(x,y)\frac{\partial g}{\partial x}(x,y) + g(x,y)\frac{\partial f}{\partial x}(x,y)$$

implies

$$P'_x = (1 + x + \cdots + x^{p-1})g'_x + (1+x^p)f'_x,$$

and multiplying both sides by $1-x$ results in

$$(1-x)P'_x = (1-x^p)g'_x + [x^p(1-x) + (1-x)]f'_x.$$

The left-hand side may be written as

$$\sum_{i=1}^{2p-1} a_i x^{i-1} - \sum_{i=1}^{2p-1} a_i x^i = a_1 + \sum_{i=1}^{2p-2} (a_{i+1} - a_i) x^i - a_{2p-1} x^{2p-1} = \sum_{i=0}^{2p-2} s_{i+1} x^i - a_{2p-1} x^{2p-1},$$

while the right-hand side may be written as

$$\underbrace{[(1-x)f'_x + g'_x]}_{(I)} + \underbrace{x^p \cdot [(1-x)f'_x - g'_x]}_{(II)}.$$

Note that the degree of the term denoted by I is $\leq \max\{\deg f, \deg g - 1\} = p - 1$, while all terms in (II) have degree $\geq p$. Hence,

$$(1-x)f'_x + g'_x = \sum_{i=0}^{p-1} s_{i+1} x^i,$$
$$(1-x)f'_x - g'_x = \sum_{i=p}^{2p-2} s_{i+1} x^{i-p} - a_{2p-1} x^{p-1} = \sum_{i=0}^{p-2} s_{i+1+p} x^i - a_{2p-1} x^{p-1}.$$

Subtracting the two equalities leads to

$$2g'_x = \sum_{i=0}^{p-2} (s_{i+1} - s_{i+1+p}) x^i + (s_p + a_{2p-1}) x^{p-1}.$$

Since g has integer coefficients and since the s_i's are binary valued, for $i = 1, 2 \ldots, p-1$,

$$s_i = s_{i+p},$$

and hence $s = t \circ u$, where t consists of $p-1$ bits and u is a single bit. Letting a and b denote the number of zeros and ones in the concatenation tu, $P_s(x,y) = (1 + x^a y^b) P_t(x,y)$. By Theorem 22, $P_t(x,y)$ has at most one nonpalindromic factor. It suffices to show that $1 + x^a y^b$, which itself is palindromic, has no nonpalindromic factors. If $a = 0$ or $b = 0$, $1 + x^a y^b$ has no nonpalindromic factor. If both a and b are nonzero, since $a + b = p$, the underlying polynomial has at most two factors, one of which by Lemma 19 is either $1 + x$ or $1 + y$. This is clearly not possible.

Case 3. Similar to Case 2, under modulo 2 operations,

$$g(x,x) = (1+x)(1 + x + \cdots + x^{p-1}) = 1 + x^p \pmod 2.$$

Therefore, strings of length one less than twice a prime are reconstructable. □

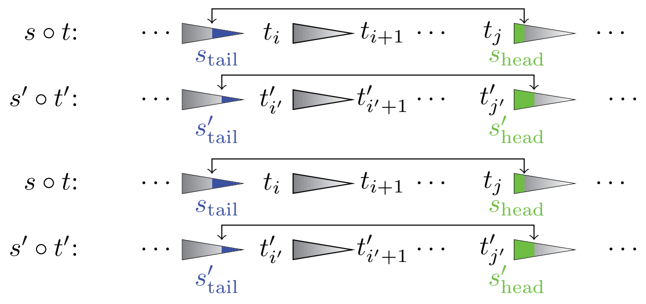
11. Lower bound on E_n. Recall that E_n denotes the largest number of mutually equicomposable n-bit strings. In section 5, we constructed confusable strings for all lengths n such that $n+1$ is a product of two integers ≥ 3. The interleaved strings are nonpalindromic, and hence $E_n \geq 4$. We next generalize the interleaving construction in two ways in order to construct equicomposable sets that for some n are as large as $(n+1)^{\log_3 2}$.

We first prove the following lemma.

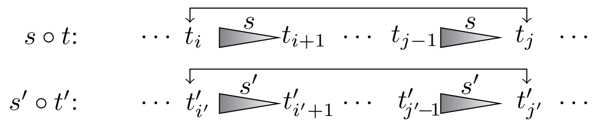
LEMMA 25. *$s \sim s'$ and $t \sim t'$, and then*

$$s \circ t \sim s' \circ t'.$$

Proof. Any substring of $s \circ t$ is of the form $s_{\text{tail}}\, t_i\, s\, t_{i+1}\, s \cdots s\, t_j\, s_{\text{head}}$, where the *tail* and *head* substrings are of the form $s_{\text{tail}} = s_\ell s_{\ell+1} \cdots s_n$ for some integer ℓ, and $s_{\text{head}} = s_1 s_2 \cdots s_h$ for some integer h. Note that the head or tail substrings may also be empty or equal to the string s. We hence show that the above described substring of $s \circ t$ may be bijectively mapped to a substring $s'_{\text{tail}}\, t'_{i'}\, s'\, t'_{i'+1}\, s' \cdots s'\, t'_{j'}\, s'_{\text{head}}$ of $s' \circ t'$, where $s'_{\text{tail}} = s'_{\ell'} s'_{\ell'+1} \cdots s'_n$ and $s'_{\text{head}} = s'_1 s'_2 \cdots s'_{h'}$.

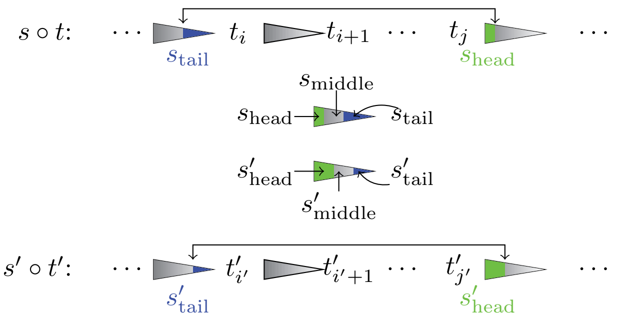


Since $s \sim s'$, there exists a bijection $f_{s,s'}$ that maps every substring of s to a substring of s' with the same composition, and a similar bijection $f_{t,t'}$ for the strings t and t'. Let $f_{t,t'}$ map the substring $t_i \cdots t_j$ to a substring $t'_{i'} \cdots t'_{j'}$ of the same composition, and therefore of the same length. Since $s \sim s'$, the strings $t_i\, s\, t_{i+1}\, s \cdots s\, t_j$ and $t'_{i'}\, s'\, t'_{i'+1}\, s' \cdots s'\, t'_{j'}$ have the same composition.



Using the tail and head notation above, it remains to show that ℓ and h can be bijectively mapped to ℓ' and h' such that $s_{\text{tail}} s_{\text{head}}$ and $s'_{\text{tail}} s'_{\text{head}}$ have the same composition. There are two cases to consider.

If the length of $s_{\text{tail}} s_{\text{head}}$ is at most n, removing s_{tail} and s_{head} from s yields $s_{\text{middle}} = s_{h+1} \cdots s_{\ell-1}$, which $f_{s,s'}$ maps to a substring $s'_{\text{middle}} = s'_{h'+1} \cdots s'_{\ell'-1}$ of s'. Removing s'_{middle} from s' yields a tail $s'_{\text{tail}} = s'_{\ell'} \cdots s'_n$ and a head $s'_{\text{head}} = s'_1 \cdots s'_{h'}$ of s'. Their "combined composition" equals that of the original head and tail substrings.



Similarly, if the length of $s_{\text{tail}} s_{\text{head}}$ is greater than n, s_{tail} and s_{head} have a common substring $s_{\text{middle}} = s_\ell \cdots s_h$, which $f_{s,s'}$ maps to a substring $s'_{\text{middle}} = s'_{\ell'} \cdots s_{h'}$ in s'. The head $s'_{\text{head}} = s'_1 \cdots s'_{h'}$ and tail $s'_{\ell'} \cdots s'_n$ of s' have the same combined composition as the original head and tail. ∎

The second generalization of the interleaving construction is to interleave more than two strings. For this multistring interleaving construction, we find the following properties of interleaved strings useful:

Associativity. $(s_1 \circ s_2) \circ s_3 = s_1 \circ (s_2 \circ s_3)$.

Unique factorization. Every string has a unique maximal factorization into interleaved strings.

Proof. We first show that if $s \circ s' = t \circ t'$, where s and t are irreducible under the interleaving operation, then both equal $u \circ u'$ for some string u of length $(|s|+1, |t|+1) - 1$, where (m, n) is the greatest common divisor of m and n. If $|s| = |t|$, then we easily get $s = t$, as otherwise we would have $(|s|+1) \cdot (|s'|+1) = (|t|+1) \cdot (|t'|+1)$ and hence $(|s|+1, |t|+1) = |u|+1 > 1$. We can expand $s = u_1 a_1 u_2 a_2 \cdots u_M$, where $M = (|s|+1)/(|u|+1)$, $t = v_1 b_1 v_2 b_2 \cdots v_N$ with $N = (|t|+1)/(|u|+1)$ and $|u_i| = |v_i| = |u|$. Note that each of the a_i and b_i elements are bits. Since $(M, N) = 1$ for any $(i, j) \in ([M], [N])$, there are substrings s^* and t^* of $s \circ s'$ and $t \circ t'$, respectively, starting and ending at the same index, such that $s^* = u_i$ and $t^* = v_j$. This implies that $u_i = v_j$ for all i and j, and thus both s and t are reducible. ▯

THEOREM 26.

$$E_{s_1 \circ s_2 \circ \cdots \circ s_k} = \{t_1 \circ t_2 \circ \cdots \circ t_k : t_i \sim s_i \textit{ for all } i\}.$$

Proof. Proving $\supseteq$ is straightforward. By associativity and induction, if $s_i \sim t_i$ for $i = 1, \ldots, m$, then

$$s_1 \circ s_2 \circ \cdots \circ s_m \sim t_1 \circ t_2 \circ \cdots \circ t_m.$$

The proof of $\subseteq$ is more involved and relegated to Appendix B. ▯

The ($\supseteq$) part of the proof suggests a simple construction of large confusable sets. Every string is equicomposable with its reversal, and hence

$$E_{s_1 \circ s_2 \circ \cdots \circ s_k} \supseteq \{\tilde{s}_1 \circ \tilde{s}_2 \circ \cdots \circ \tilde{s}_k : \tilde{s}_i \text{ is either } s_i \text{ or } s_i^*\}.$$

If all the aforementioned strings s_i are nonpalindromic, then by unique factorization, each of the resulting interleaved products is different. Hence,

$$|E_{s_1 \circ s_2 \circ \cdots \circ s_k}| \geq 2^k.$$

Let $|s_i| = m_i$. We have $|s_1 \circ s_2| = (m_1+1)(m_2+1) - 1$, and by induction

$$|s_1 \circ s_2 \circ \cdots \circ s_k| = \prod_{i=1}^{k} (m_i + 1) - 1.$$

For example, taking $s_1 = \ldots = s_k = 01$ and $n = 3^k - 1$ leads to

$$E_n = E_{3^k - 1} \geq |E_{\underbrace{01 \circ 01 \circ \cdots \circ 01}_{k}}| \geq 2^k = (n+1)^{\log_3 2}.$$

For general lengths n, consider the unique prime factorization

$$n + 1 = 2^{e_0} p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}.$$

For every prime $p \geq 3$, take a nonpalindromic string of length $p - 1$, and for every pair of 2's, take a nonpalindromic string of length $2 \cdot 2 - 1 = 3$. (It is straightforward to see that nonpalindromic binary strings of length greater than one always exist.) Interleaving all these strings and their reversals lower bounds the largest number of mutually equicomposable strings, as stated in the next theorem.

THEOREM 27.

$$E_n \geq 2^{\lfloor \frac{e_0}{2} \rfloor + e_1 + e_2 + \cdots + e_k}.$$

Throughout the remainder of the section, we discuss possible extensions of the interleaving construction.

The confusable strings we have exhibited so far were interleaved as in Theorem 26. Exhaustive search shows that so are all confusable strings of length ≤ 22. Yet, Theorem 26 shows that equicomposable sets larger than implied by Theorem 27 must be based on noninterleaved strings. We next provide a number of examples of noninterleaved confusable strings.

We begin with a polynomial interpretation of interleaving.

LEMMA 28. *For any string s with a 0's and b 1's and any string t,*

$$P_{s \circ t}(x,y) = P_s(x,y) P_t(x^{a+1}y^b, x^a y^{b+1}).$$

Proof. Let $t \overset{\text{def}}{=} t_1 t_2 \cdots t_m$, so that $s \circ t = s t_1 s t_2 \cdots t_m s$, and define $a_i = \sum_{j=1}^{i} t_j$ and $b_i = \sum_{j=1}^{i}(1-t_j)$ to be the numbers of 0's and 1's in $t_1 t_2 \cdots t_i$. Then,

$$\begin{aligned} & P_{s \circ t}(x,y) \\ &= P_s(x,y) + x^{a+t_1} y^{b+(1-t_1)} P_s(x,y) + \cdots + x^{ma+t_1+\cdots+t_m} y^{mb+(1-t_1)+\cdots+(1-t_m)} P_s(x,y) \\ &= P_s(x,y) \sum_{i=1}^{m} x^{ia+a_i} y^{ib+b_i}. \end{aligned}$$

Therefore,

$$\begin{aligned} \sum_{i=1}^{m} x^{ia+a_i} y^{ib+b_i} &= \sum_{i=1}^{m} x^{(a_i+b_i)a+a_i} y^{(a_i+b_i)b+b_i} \\ &= \sum_{i=1}^{m} (x^{a+1}y^b)^{a_i} (x^a y^{b+1})^{b_i} = P_t(x^{a+1}y^b, x^a y^{b+1}), \end{aligned}$$

proving the lemma. $\square$

The next result is a straightforward consequence of Theorem 12.

THEOREM 29. *Let $s_1, s_2, \ldots, s_k$ be strings whose generating polynomials have a common factor $D(x,y)$. Derive $t_1, t_2, \ldots, t_k$ from $s_1, s_2, \ldots, s_k$ by replacing $D(x,y)$ by its reciprocal. Then for any $x_1 x_2 \ldots x_{k-1}$,*

$$s_1\, x_1\, s_2\, x_2 \,\cdots\, x_{k-1}\, s_k \sim t_1\, x_1\, t_2\, x_2 \,\cdots\, x_{k-1}\, t_k. \tag{7}$$

It is straightforward to see that equicomposable strings obtained via our interleaving procedure represent a special case of the construction of Theorem 29. We also have the following consequences of the aforementioned results.

COROLLARY 30. *Let $u \sim v$ and for $i = 1, 2, \ldots, k$, let $(s_i, t_i) \in \{(u, v'), (v, u')\}$, where $u \sim u'$ and $v \sim v'$. Then, for any string $\bar{x} = x_1 x_2 \cdots x_{k-1}$, the two strings listed in* (7) *are equicomposable.*

Corollary 31. *Let $s_1, \ldots, s_k$ be strings with the same composition. Then, for any string s_0 and bits $x_1 x_2 \ldots x_{k-1}$,*

$$(s_1 \circ s_0)x_1(s_2 \circ s_0)x_2 \ldots x_{k-1}(s_k \circ s_0) \sim (s_1 \circ s_0^*)x_1(s_2 \circ s_0^*)x_2 \ldots x_{k-1}(s_k \circ s_0^*).$$

For example, taking $s_1 = 010$, $s_2 = 001$, $s_0 = 01$, and $x_1 = 0$, we obtain the 23-bit confusable strings

$$\begin{aligned}\mathbf{010}0\mathbf{010}1\mathbf{010}\mathit{0}\mathbf{001}0\mathbf{001}1\mathbf{001} &= (\mathbf{010} \circ 01)\mathit{0}(\mathbf{001} \circ 01)\\ &\sim (\mathbf{010} \circ 10)\mathit{0}(\mathbf{001} \circ 10)\\ &= \mathbf{010}1\mathbf{010}0\mathbf{010}\mathit{0}\mathbf{001}1\mathbf{001}0\mathbf{001}.\end{aligned}$$

These strings are clearly noninterleaved and represent the shortest example of noninterleaved confusable strings. Note that this example also shows that string equicomposability differs from partition coarsening [4] using ribbon Schur functions.

12. Upper bounds on E_n.

To upper bound the largest number of mutually equicomposable strings, we use well-known results on the factorization of $x^n - 1$. Recall that $d(n)$ denotes the number of divisors of n.

Theorem 32. *For all n,*

$$E_n \le \min\left\{2^{d(n+1)-1}, (n+1)^{1.23}\right\}.$$

Proof. The $(n+1)^{1.23}$ bound follows from results in [16] by replacing y in $P(x,y)$ by x^{n+1} to obtain a univariate polynomial, and reproving theorems therein for this polynomial, showing that $E_n \le (n+1)^{1.23}$.

To prove the first upper bound, let s have length n. Then,

$$P_s(x,x) = 1 + x + x^2 + x^3 + \ldots + x^n = \frac{x^{n+1}-1}{x-1}.$$

Hence, the prime factorization of $P_s(x,x)$ contains $d(n+1)-1$ terms. From Lemma 20, the prime factorization of $P_s(x,y)$ contains at most that many terms, and the bound follows from Corollary 13. ▯

To relate the form of this upper bound to that of the lower bound described in Theorem 27, let once more $n+1 = 2^{e_0} p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$ be the unique factorization of $n+1$ into distinct primes. Then, $d(n+1) = (e_0+1)(e_1+1)\cdots(e_k+1)$, and hence

$$E_n \le \min\left\{2^{(e_0+1)(e_1+1)\cdots(e_k+1)-1}, (n+1)^{1.23}\right\}.$$

The lower bound in Theorem 27 is tight when $n+1$ is a prime power or twice a prime power.

Theorem 33. *For any k,*

$$E_{2^k-1} = 2^{\lfloor \frac{k}{2} \rfloor},$$

and for prime $p \ge 3$,

$$E_{p^k-1} = 2^k.$$

Proof. Eisenstein's criterion implies the prime factorization

$$P_s(x,x) = 1 + x + x^2 + \cdots + x^{2^k-1} = \prod_{j=1}^{k-1}(1 + x^{2^j}).$$

To prove the bound, it suffices to show that any factorization of $P_s(x, y)$ of the form $f(x, y)g(x, y)$ with $g(x, x) = 1 + x^{2^i}$ implies that $g(x, y)$ is palindromic.

Let $s = s_1 s_2 \dots s_{2^k-1}$ and as before, let a_i be the number of zeros among $s_1, \dots, s_i$. Then $P(x, y) \overset{\text{def}}{=} P_s(x, y) = \sum_{i=0}^{2^k-1} x^{a_i} y^{i-a_i}$. Denote $\frac{\partial P}{\partial x}(x, y)|_{y=x}$ by P'_x. Then

$$(1 - x)P'_x = (1 - x)f(x, x)g'_x + (1 - x)g(x, x)f'_x$$

implies that

$$\sum_{i=0}^{2^k-2} s_{i+1}x^i - a_{2^k-1}x^{2^k-1} = \left(1 - x^{2^i} + x^{2\cdot 2^i} - x^{3\cdot 2^i} + \cdots - x^{(2^{k-i}-1)\cdot 2^i}\right) g'_x + (1 + x^{2^i})(1 - x)f'_x.$$

The degree of g'_x is $2^i - 1$ and that of $(1 - x)f'_x$ is $2^k - 1 - 2^i$, so we may write

$$(1 - x)f'_x = \sum_{j=0}^{2^{k-i}-2} x^{j\cdot 2^i} f_j,$$

where each f_j has degree $\leq 2^i - 1$. Defining $f_{-1} = f_{2^k} = 0$ and $s_{2^k} = -a_{2^k-1}$ leads to

$$\sum_{i=0}^{2^k-2} s_{i+1}x^i - a_{2^k-1}x^{2^k-1} = \sum_{j=0}^{2^{k-i}-1} (-1)^j x^{j\cdot 2^i} g'_x + \sum_{j=0}^{2^{k-i}-2} x^{j\cdot 2^i} f_j + \sum_{j=1}^{2^{k-i}-1} x^{j\cdot 2^i} f_j$$

$$\Rightarrow \sum_{j=0}^{2^{k-i}-1} x^{j\cdot 2^i} \left(\sum_{l=0}^{2^i-1} s_{j\cdot 2^i+l+1} x^l \right) = \sum_{j=0}^{2^{k-i}-1} x^{j\cdot 2^i} \left((-1)^j g'_x + f_{j-1} + f_j \right).$$

If we sum up all the terms with even values of j in the above equation and subtract the result from the sum of the terms corresponding to odd values, we get

$$\sum_{j=0}^{2^{k-i}-1} (-1)^{j\cdot 2^i} \left(\sum_{l=0}^{2^i-1} s_{j\cdot 2^i+l+1} x^l \right) = 2^{k-i} g'_x.$$

Since all elements s_l except for possibly s_{2^k} are either 0 or 1, we must have

$$g'_x = a \cdot x^{2^i-1},$$

and thus

$$g(x, y) = 1 + g_1(x, y) + x^a y^{2^i-a}.$$

Claim 34. *If $f(x, x) = K$, a constant, and $f'_x = 0$, then $(x - y)^2 | f(x, y) - K$.*

Proof. Let $f(x, y) = (x - y)f_1(x, y) + f_2(x)$. Then, clearly $f_2(x) = K$. Also, $f'_x = 0$ implies $f_1(x, x) = 0$, and thus $f_1(x, y)$ has $x - y$ as a factor. □

Using the claim, we conclude that $g(x, y) = 1 + (x - y)^2 g_1(x, y) + x^a y^{2^i-a}$.

Now, consider

$$f(x, y) (g - g^*).$$

The polynomial is the difference of two generating polynomials, but substituting $y = -x$, we see that each coefficient is divisible by 4, which cannot be true unless $g - g^* = 0$.

It is easy to see that the bound is achievable by taking strings of the form $t_0 \circ t_1 \circ t_2 \circ \cdots \circ t_{\lfloor \frac{k}{2} \rfloor}$, where $t_i = 001$ for $i \geq 1$, and t_0 is the empty string when k is even, and otherwise it is the single bit 0.

When $n = p^k - 1$ for a prime $p \geq 3$, Theorems 27 and 32 provide the same result. □

13. Reconstruction algorithm.

We next present an algorithm for reconstructing strings from their composition multiset. The algorithm takes as input the composition multiset $\mathcal{S}_s$ of a string s over an alphabet Σ and outputs elements of E_s, the set of all strings confusable with s. We show that for alphabet size ≥ 4, the string s that generated $\mathcal{S}_s$ is added to the reconstructed sequence list in quadratic time. The algorithm successively reconstructs s from both ends and backtracks when it errs. It can be viewed as a modification of a similar algorithm for the turnpike problem [8].

We first establish two results regarding $\mathcal{S}_s$ that help reduce the search space of the algorithm. The first result shows that the composition multiset determines the set $\{s_i, s_{n+1-i}\}$ of symbols at the symmetric positions i and $(n+1-i)$ for $i = 1, 2, \ldots, \lceil \frac{n}{2} \rceil$.

LEMMA 35. *$\mathcal{S}_s$ determines the multiset $\{s_i, s_{n+1-i}\}$ for $i = 1, 2, \ldots, \lceil \frac{n}{2} \rceil$.*

Proof. Let the union of compositions be their multiset union: for example, $A^2B \cup ABC^2 \cup AC = A^4B^2C^3$. For a string s, let M_i denote the union of the compositions of all substrings of length i. For example, for ABAC, $M_1 = A^2BC$, $M_2 = A^3B^2C$. Note that all unions M_i can be easily determined from the composition multiset $\mathcal{S}_s$ and that for $1 \leq i \leq \lfloor n/2 \rfloor$, $M_{n+1-i} = M_i$. For a multiset S, let $j \times_U S$ denote the j-fold union $S \cup \ldots \cup S$. It is easy to see that

$$M_2 \cup \{s_1, s_n\} = 2 \times_U M_1,$$

and hence $\{s_1, s_n\}$ can be determined from $\mathcal{S}_s$. More generally, for $i = 2, \ldots, \lceil \frac{n}{2} \rceil$,

$$M_i \cup \{s_{i-1}, s_{n-i+2}\} \cup 2 \times_U \{s_{i-2}, s_{n-i+3}\} \cup \ldots \cup (i-1) \times_U \{s_1, s_n\} = i \times_U M_1.$$

Using this equation inductively over $i = 2, \ldots, \lceil \frac{n}{2} \rceil$ yields all multisets $\{s_i, s_{n-i+1}\}$. □

For $1 \leq i < \lceil n/2 \rceil$, let $\mathcal{T}_i$ be the collection of compositions of strings s_j^k where $j, k \leq i$, $j, k \geq n+1-i$, or $j \leq i+1 \leq n-i \leq k$, namely, the collection of compositions of all strings that are either on "one side" of s_{i+1}^{n-i} or "straddle" s_{i+1}^{n-i}. (For ease of notation, we henceforth use s_i^j, $j \geq i$, to denote the substring $s_i, s_{i+1}, \ldots, s_j$.)

The next lemma shows that the composition of the whole string, along with the strings s^i and s_{n+1-i}^n, determines $\mathcal{T}_i$.

LEMMA 36. *$\mathcal{S}_s$, s^i, and s_{n+1-i}^n determine $\mathcal{T}_i$.*

Proof. $\mathcal{T}_i$ consists of compositions of three types of strings: those that are substrings of s_1^i, that are substrings of s_{n+1-i}^n, and that cover all symbols in between, i.e., s_{i+1}^{n-i}. The first and last i symbols determine the compositions of the first two type of strings. The third type of strings are those that contain the string s_{i+1}^{n-i}. This is a *symmetric substring*, namely, it has the same number of symbols on its left and right, and by Lemma 35 we can determine its composition. Knowing this composition and the first and last i symbols yields the multiset of strings of the third type. □

We use the two lemmas to devise a method for reconstructing the underlying string.

LEMMA 37. *Let $w(s)$ and as before denote the composition of a string. If $w(s_1^i) \neq w(s_{n-i+1}^n)$, then $\mathcal{S}_s$, s_1^i, and s_{n+1-i}^n determine the pair (s_{i+1}, s_{n-i}).*

Proof. By Lemma 36, we can determine $\mathcal{T}_i$. Consider the two longest compositions in $\mathcal{S}_s \setminus \mathcal{T}_i$. They correspond to the length-$(n-i-1)$ strings s_1^{n-i-1} and s_{i+2}^n. The complements of these two compositions are therefore the compositions of s^{i+1} and s_{n-i}^n.

By Lemma 35, we can also determine the multiset $\{s_{i+1}, s_{n-i}\}$. If $s_{i+1} = s_{n-i}$, then we can determine s^{i+1} and s_{n-i}^n. Otherwise, $s_{i+1} \neq s_{n-i}$, and since $w(s_1^i) \neq w(s_{n+1-i}^n)$, it is easy to see that $\{w(s_1^i) \cup w(s_{i+1}), w(s_{n-i}) \cup w(s_{n-i+1}^n)\} \neq \{w(s_1^i) \cup w(s_{n-i}), w(s_{i+1}) \cup w(s_{n-i+1}^n)\}$, and hence we can determine the pair (s_{i+1}, s_{n-i}). ☐

The algorithm reconstructs the string by sequentially deciding on the values of the pair of symbols s_i and s_{n-i+1} that occupy symmetric positions. It starts with the symbols s_1 and s_n at the two ends of the string and progressively moves toward the center. The algorithm then determines $s_1 s_2$ and $s_{n-1}s_n$, up to reversal. By Lemma 35, we know the multiset $\{s_1, s_n\}$ and may arbitrarily choose which symbol is s_1 and which is s_n. The next step of the procedure is to determine s_2 and s_{n-1}. Again, by the lemma, we know $\{s_2, s_{n-1}\}$, and if $s_1 = s_n$, we can decide on s_2 and s_{n-1} arbitrarily, while if $s_1 \neq s_n$, by Lemma 37, we can determine s_2 and s_{n-1}. For $\{s_3, s_{n-2}\}$, the ends s_1^2 and s_{n-1}^n may differ, while their weights could be the same. In such cases, Lemma 37 cannot be applied. However, if $s_3 = s_{n-2}$, which can be deduced from Lemma 35, we can easily determine (s_3, s_{n-2}). In other words, if $s_i = s_{n+1-i}$, the algorithm easily determines (s_i, s_{n+1-i}). Therefore, from this point on, when the algorithm has reconstructed the first and last i symbols, and $w(s_1^i) = w(s_{n+1-i}^n)$ but $s_{i+1} \neq s_{n+1-i}$, it guesses one of the two possibilities for (s_{i+1}, s_{n+1-i}) and proceeds, while keeping track of the number t and locations $i_1 < i_2 < \ldots < i_t$ where guesses were made. After determining/guessing s_{i+1} and s_{n-i}, the algorithm updates $\mathcal{T}_i$ to T_{i+1}. It then checks whether $\mathcal{T}_{i+1} \subseteq \mathcal{S}_s$ as multisets. If at some point $\mathcal{T}_{i+1} \subsetneq \mathcal{S}_s$, the algorithm *backtracks*. It changes its guess at location i_t by swapping s_{i_t} and s_{n+1-i_t}, changes t to $t-1$, and restarts reconstruction from location $i_t + 1$. The process continues until the whole string is reconstructed, namely, until $i = \lceil \frac{n}{2} \rceil$ and $\mathcal{T}_{\lceil n/2 \rceil} = \mathcal{S}_s$. If at that point there are $t \geq 1$ guesses, then as before the algorithm backtracks to guess t and identify additional strings in E_s.

Since our algorithm relies on the compositions of strings computed from both ends, it is helpful to introduce some terminology that will ease the exposition.

Let

$$\ell_s \overset{\text{def}}{=} \left|\{i < n/2 : w(s^i) = w(s_{n+1-i}^n) \text{ and } s_{i+1} \neq s_{n-i}\}\right|$$

be the number of substrings read from both ends having the same composition, such that their immediately following two symbols are distinct. In other words, ℓ_s denotes the number of *maximal equicomposition substrings* appearing as prefixes and suffixes.

Furthermore, let

$$L_s \overset{\text{def}}{=} \max_{t \in E_s} \ell_t$$

be the largest value of ℓ over all strings in E_s.

The backtracking algorithm induces a binary tree where the nodes represent locations at which there are two possible reconstructions. The procedure described above does a depth-first search. Instead, we could also perform a breadth first search, where we consider all branches at once, namely, at any time, all potential reconstructions correspond to level t or $t+1$. This implies that given $\mathcal{S}_s$, the algorithm is able to find s before depth $\ell_s + 1$.

We bound the time required to reconstruct s from $\mathcal{S}_s$ in the following theorem.

THEOREM 38. *The backtracking algorithm, run using proper data structures, and given inputs $\mathcal{S}_s$ and ℓ_s, outputs a set of strings that contains s in time $O(2^{\ell_s} n^2 \log n)$. Furthermore, E_s can be recovered in time $O(2^{L_s} n^2 \log n)$.*

Proof. We assume an arbitrary order over the set of symbols in Σ. This introduces a lexicographical ordering over compositions of strings on Σ. We use a red-black tree [6] to store multisets of compositions. The advantage of this data structure is that insertion, deletion, and search all require time $O(\log n)$, where n is the length of the underlying string.

Notice that $\mathcal{T}_{i+1}$ is obtained from $\mathcal{T}_i$ by adding the compositions of substrings that have as endpoints s_{i+1} or s_{n-i}. In particular, at most $2n$ compositions are added, requiring $O(n \log n)$ time. For each branch, we keep a copy of $\mathcal{S}_s$ and prune it to populate $\mathcal{T}$, i.e., while constructing $\mathcal{T}_{i+1}$ we simultaneously remove the new entries/compositions from the copy of $\mathcal{S}$ corresponding to this branch. This procedure requires additional $O(n \log n)$ operations. When there are two possible reconstructions at any stage, we make copies of $\mathcal{T}_i$ and $\mathcal{S}$ corresponding to that node and proceed along each. This step takes time $O(n^2 \log n)$.

While reconstructing s, the algorithm does not "fork out" more then $\ell_s + 1$ times, and therefore the number of branches created before reconstructing s is at most 2^{ℓ_s}. Along each path, we make $n/2$ iterations requiring a total time of $O(n^2 \log n)$. Combining these, the total complexity of reconstructing s is at most $O(2^{\ell_s} n^2 \log n)$.

To reconstruct E_s, we note that the number of "forks" is $\le L_s$. Therefore, a similar computation shows that the algorithm runs in time $O(2^{L_s} n^2 \log n)$. ◻

We have bound the run time of reconstructing a string as a function of ℓ_s. For random strings, we bound the run time by studying the distribution of ℓ_s.

We do this by using some well-known results in random walk theory [28] and Stirling's approximation formula for factorials.

LEMMA 39 (Stirling's approximation). *For any $n \ge 1$, there is a $\theta_n \in (\frac{1}{12n+1}, \frac{1}{12n})$ such that*

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\theta_n}.$$

We now apply this result to bound the probability that two random strings have the same composition. A stronger version of the result that finds asymptotic equality is proved in [25] but is not required for our purposes.

LEMMA 40. *For $|\Sigma| = k$, let s and t be two random strings over Σ of length n. Then, for $k \ge n$,*

$$P\left(w(s) = w(t)\right) < \frac{n!}{k^n},$$

and for $k < n$

$$P\left(w(s) = w(t)\right) < \frac{k^{k/2} e^{1/12n}}{(2\pi n)^{(k-1)/2}}.$$

Proof. The probability that the symbols in Σ appear, respectively, $i_1, \dots, i_k$ times in a random string of length n, under some lexical ordering of the symbols, is

$$\frac{1}{k^n} \binom{n}{i_1, \dots, i_k},$$

where $i_1 + \ldots + i_k = n$. Therefore, the probability that two random strings have the same composition is

$$\sum_{i_1+\ldots+i_k=n} \frac{1}{k^{2n}} \binom{n}{i_1,\ldots,i_k}^2 \leq \frac{1}{k^n} \max_{i_1+\ldots+i_k=n} \binom{n}{i_1 \ldots i_k} \cdot \sum_{i_1+\ldots+i_k=n} \frac{1}{k^n} \binom{n}{i_1,\ldots,i_k}$$
$$= \frac{1}{k^n} \max_{i_1,\ldots,i_k} \binom{n}{i_1 \ldots i_k},$$

where the last step follows from

$$\sum_{i_1+\ldots+i_k=n} \frac{1}{k^n} \binom{n}{i_1,\ldots,i_k} = 1.$$

For $k \geq n$,

$$\max_{i_1,\ldots,i_k} \binom{n}{i_1 \ldots i_k} = n!,$$

implying the first part of the claim.

For $n \geq k$, note that

$$f(x) \overset{\text{def}}{=} \sqrt{2\pi x} \left(\frac{x}{e}\right)^x$$

is convex in $(1, \infty)$. Therefore,

$$\prod_{j=1}^{k} i_j! \overset{(a)}{>} \prod_{j=1}^{k} f(i_j) \overset{(b)}{>} \left(f\left(\frac{n}{k}\right)\right)^k = \left(\sqrt{2\pi\frac{n}{k}}\right)^k \left(\frac{n}{ke}\right)^n,$$

where (a) follows from Stirling's approximation, and (b) follows from convexity of f. Hence,

$$\max_{i_1,\ldots,i_k} \binom{n}{i_1 \ldots i_k} < \frac{n!}{\left(f\left(\frac{n}{k}\right)\right)^k} < \frac{k^{k/2} e^{1/12n}}{(2\pi n)^{(k-1)/2}},$$

where in the last step we approximated $n!$ using Stirling's formula. ☐

We now study the distribution of ℓ_s. We first consider alphabet sizes ≥ 4 in detail and then provide separate performance guarantees for alphabet sizes 3 and 2.

14. Alphabet size $\geq$ 4.

Consider two uniformly at random generated independent infinite strings s^∞ and t^∞ over alphabet Σ of size k. A set of integers $i_1 < i_2 < \ldots < i_m$ is termed *nonconsecutive* if $i_{j+1} > i_j + 1$.

Let F_m be the event that there are at least $m+1$ *nonconsecutive* integers $i_0 \overset{\text{def}}{=} 0 < i_1 < i_2 < \ldots < i_m$ such that for each $j \leq m$, $w(s_1^{i_j}) = w(t_1^{i_j})$. After a location i_j at which $w(s_1^{i_j}) = w(t_1^{i_j})$ holds is reached, by independence, the process becomes equivalent to a process starting at index 1. It hence follows that $P(F_{j+1}|F_j) = P(F_1)$. Therefore,

$$P(F_m) = P(F_1)^m. \tag{8}$$

Let $M(s^\infty, t^\infty)$ denote the total number of nonconsecutive integers for which $w(s_1^i) = w(t_1^i)$. Then by (8),

$$\mathbb{E}[M] = \sum_{m\geq 1} P(M \geq m) = \sum_{m \geq 1} P(F_m) = \frac{P(F_1)}{1 - P(F_1)}.$$

However, if instead of nonconsecutiveness, we only restrict $i_1 \geq 2$, Lemma 40 shows that for an alphabet of size k, one has

$$\mathbb{E}[M] \leq \sum_{n=2}^{k} \frac{n!}{k^n} + \sum_{n=k+1}^{\infty} \frac{k^{k/2} e^{1/12n}}{(2\pi n)^{(k-1)/2}}.$$

The right-hand side of this equation is finite for $k \geq 4$. In fact, it decays as $O(1/k^2)$ with k. This implies that $p_k \overset{\text{def}}{=} P(F_1) < 1$, and therefore for a random string s, (8) gives

$$P(\ell_s > m) \leq P(F_m) = p_k^m,$$

proving the following lemma.

LEMMA 41. *A random string over alphabet size $k \geq 4$ with probability $\geq 1 - p_k^m$ satisfies $\ell_s \leq m$.*

This implies that with probability $\geq 1 - \delta$, $\ell_s < \log_{1/p_k} \frac{1}{\delta}$. Therefore, applying Theorem 38 leads to the next theorem.

THEOREM 42. *For a random string s over an alphabet of size ≥ 4, the backtracking algorithm with probability $> 1 - \delta$ outputs a subset of E_s containing s in time $O_{\delta,k}(n^2 \log n)$.*

Recall from Theorem 32 that $|E_s| < n^{1.23}$. Therefore, by the union bound, we obtain the following result.

LEMMA 43. *With probability $\geq 1 - n^{1.23} p_k^m$, a random string over alphabet size $k \geq 4$ satisfies $L_s < m$.*

Applying these two lemmas to Theorem 38, we prove the next theorem.

THEOREM 44. *For a random string s over an alphabet of size ≥ 4, the backtracking algorithm with probability $> 1 - \delta$ outputs E_s in time $O_\delta\left(n^{1.23 \log_{1/p_k} 2} n^2 \log n\right)$.*

Recall that p_k decays as $1/k^2$, and for sufficiently large k, the algorithm reconstructs the entire set E_s in near quadratic time.

We now consider strings over alphabet sizes 3 and 2.

15. Alphabet size 3 and 2.

By Lemma 40 for $k = 3$, two random strings of length $n \geq 2$ have the same composition with probability less than

$$\frac{3\sqrt{3} e^{1/12n}}{2\pi n} < \frac{0.87}{n}.$$

Recall that the harmonic sum $H_n = 1 + 1/2 + \ldots + 1/n$ converges to $\ln n + \gamma$, where γ is the Euler–Mascheroni constant, and therefore, for a random string s,

$$\mathbb{E}[\ell_s] < \sum_{i=2}^{n/2} \frac{0.87}{n} < 0.87 \ln n + 0.87\gamma.$$

By Markov's inequality, with probability $\geq 1 - \delta$, a random string over alphabet of size 3 satisfies $\ell_s < \frac{0.84 \ln n + 0.84\gamma}{\delta}$. Again, applying Theorem 38 establishes the next theorem.

Theorem 45. *The backtracking algorithm for random strings over alphabet size 3 outputs with probability $\geq 1-\delta$ a subset of E_s containing s in time $O\left(n^{\frac{0.6}{\delta}}\right)$.*

For alphabet size 2, Lemma 40 shows that

$$\mathbb{E}[\ell_s] < 1.9\sqrt{n} + 0.84\gamma.$$

A similar use of Markov's inequality and Theorem 38 yield to the last theorem of this section.

Theorem 46. *The backtracking algorithm for random strings over alphabet size 2 outputs a subset of E_s containing s with probability $\geq 1-\delta$ in time $O\left(2^{\frac{2\sqrt{n}}{\delta}}\right)$.*

16. Conclusions and extensions.

Starting with the problem of protein mass-spectrometry reconstruction, we made two simplifying assumptions: that all peptide bond cuts are equally likely and that substring weights imply their compositions. These two assumptions reduced protein-reconstruction to the simple problem of reconstructing a string from its substring compositions. We noted that this is the only unstudied variation of four related substring reconstruction problems, that solving the problem for binary strings suffices to provide a solution for all alphabet sizes, and that the reconstruction problem represents a combinatorial simplification of the long-open turnpike problem.

We called strings with the same composition multiset *equicomposable*, strings equicomposable only with themselves and their reversal *reconstructable*, and those with more than these two trivial equicomposable strings *confusable*. We noted that all strings of length at most seven are reconstructable. For all lengths one short of a product of two integers, each at least three, we obtained confusable strings of this length.

Extending polynomial techniques used for the turnpike problem, we represented strings as bivariate *generating* polynomials. We used this formulation to characterize equicomposability in terms of both polynomial multiplication and polynomial factorization, showing in particular that equicomposable strings are determined exactly by the prime factorization of their generating polynomials. We then showed that all strings of lengths not included in the earlier construction, namely, seven, and one less than either a prime or twice a prime, are reconstructable.

Interleaving multiple strings, we constructed sets of $(n+1)^{\log_3 2}$ length-n strings that are mutually equicomposable and exhibited a pair of noninterleaved confusable strings. Using cyclotomic polynomials, we upper bounded the largest number of confusable strings, showing in particular that the lower bound is tight when the sequence length is one short of a prime power and twice a prime power.

Many questions remain. All confusable strings that we are aware of are described by Theorem 29. Finding other confusable strings, or proving that this describes all confusable strings, would be of interest. We made two assumptions. The first implied that we are given the composition of all substrings. What happens when we are given a fraction of all substrings or of all substrings of a given length? The proofs provided here use algebraic arguments. Direct combinatorial proofs would be of interest. The second assumption was that all compositions are given correctly. It would be interesting to know whether some errors can be tolerated.

While prime-related reconstructability may be interesting, reconstructability for arbitrary lengths may be of more practical relevance. It would be interesting to determine whether the lower bound of the size of equicomposable sets in Theorem 27 is always tight. It would mean that every string is confusable with at most a sublinear number of strings. A related question is whether most strings of a given length are reconstructable. This question is related to the open problem of whether most 0-1 polynomials are irreducible over the integers [22].

A related question addresses the number of composition multisets. If this number is close to 2^n, then most strings can be reconstructed. Another variation is when instead of a string, the bits are arranged on a ring. The constructions presented here extend to ring. Proving the upper bounds is still open.

Other problems relate to algorithms for reconstructing a string from its substring composition. As noted earlier, n-bit reconstruction can be reduced to solving a turnpike problem with $n+1$ exits and total length $\leq n^2$. This implies a polynomial-time algorithm for the reconstruction. However, such a generic algorithm may have high complexity, and an algorithm that uses the structure of the reconstruction problem is of interest.

Appendix A. Proof of Theorem 21. The proof follows along the same line as the derivations behind Theorem 24.

We first show that in any factorization

$$P_s(x,y) = f(x,y)g(x,y),$$

at least one of the polynomials $f(x,y)$ and $g(x,y)$ is palindromic. By Lemma 23, s is reconstructable.

Write

$$P_s(x,x) = 1 + x + x^2 + \cdots + x^7 = (1+x)(1+x^2)(1+x^4).$$

Hence, there are only three factorizations of $P_s(x,y)$ into two factors, considered separately in what follows:

Case 1. $f(x,x) = 1+x$ and $g(x,x) = (1+x^2)(1+x^4)$. This scenario is identical to Case 1 of Theorem 24.

Case 2. $f(x,x) = 1+x^2$ and $g(x,x) = (1+x)(1+x^4)$. In this case, we note that $f(x,y)$ has no linear terms; otherwise it must have both the terms x and y, which would imply the existence of terms of the form x^a and y^b for positive integers a and b, which violates G3.

Case 3. $f(x,x) = 1+x^4$ and $g(x,x) = (1+x)(1+x^2)$.

Let us focus on $g(x,x) = (1+x)(1+x^2) = 1+x+x^2+x^3$. Just as in the Case 2 of the proof of the Theorem 24, we can show that the string is a concatenation of the form tut, where t is a length 3 string and u is a binary symbol. This implies that $P_s = P_t(1+x^a y^b)$, where a and b are the number of 0's and 1's in tu, so that $a+b=4$. Since $1+x^4$ is irreducible, this factor is palindromic, which proves the third case as well.

Appendix B. Proof of $\subseteq$ in Theorem 26. We start by proving the following lemma.

LEMMA 47. *Let $P(x,y)$ be a generating polynomial. Any $Q(x,y) \in \mathbb{Z}[x,y]$ with constant term equal to* 1*, satisfying*

$$P(x^{a+1}y^b, x^a y^{b+1})P^*(x^{a+1}y^b, x^a y^{b+1}) = Q(x,y)Q^*(x,y),$$

has the form $R(x^{a+1}y^b, x^a y^{b+1})$, where $R(x,y)$ is a generating polynomial.

Proof: Similar to the proof of Lemma 9, we can show that

$$P(x^{a+1}y^b, x^a y^{b+1})P\left(\frac{1}{x^{a+1}y^b}, \frac{1}{x^a y^{b+1}}\right) = Q(x,y)Q\left(\frac{1}{x}, \frac{1}{y}\right), \tag{9}$$

where $Q(x,y)$ is a 0-1 polynomial.

Note that all monomials in the expansion of the left-hand side of (9) have the form

$$\left(x^{a+1}y^b\right)^s \left(x^a y^{b+1}\right)^t \cdot \frac{1}{\left(x^{a+1}y^b\right)^{s'} \left(x^a y^{b+1}\right)^{t'}} = x^{(a+1)(s-s')+a(t-t')}\, y^{b(s-s')+(b+1)(t-t')}.$$

For any monomial $x^i y^j$ present in $Q(x,y)$, consider another monomial of the form $x^i y^j \cdot 1$ in the expansion of the right-hand side of (9). Then, for some h, l and h', l',

$$\begin{aligned} i &= (a+1)(h-h') + a(l-l'), \\ j &= b(h-h') + (b+1)(l-l'). \end{aligned}$$

For simplicity, let $u_{ij} = h - h' \in \mathbb{Z}$ and $v_{i,j} = l - l' \in \mathbb{Z}$. Then

$$Q(x,y) = \sum_{i,j} x^i y^j = \sum_{i,j} \left(x^{a+1}y^b\right)^{u_{ij}} \left(x^a y^{b+1}\right)^{v_{ij}}.$$

Let $R(x,y) = \sum_{i,j} x^{u_{ij}} y^{v_{ij}}$. Then $Q(x,y) = R(x^{a+1}y^b, x^a y^{b+1})$.

It remains to show that $R(x,y)$ is a generating polynomial. Let $u \overset{\text{def}}{=} \min u_{ij}$, and $v \overset{\text{def}}{=} \min v_{ij}$. Then $T(x,y) \overset{\text{def}}{=} x^{-u}y^{-v}R(x,y) \in \mathbb{Z}[x,y]$. It is straightforward to see that

$$P(x,y)P\left(\frac{1}{x}, \frac{1}{y}\right) = T(x,y)T\left(\frac{1}{x}, \frac{1}{y}\right).$$

From Lemma 9, it follows that $T(x,y)$ is a generating polynomial. Also, note that

$$Q(x,y) = (x^{a+1}y^b)^u (x^a y^{b+1})^v\, T\left(x^{a+1}y^b, x^a y^{b+1}\right).$$

Since both $Q(x,y)$ and $T\left(x^{a+1}y^b, x^a y^{b+1}\right)$ are polynomials with constant term 1, we must have $(a+1)u + av = 0$ and $bu + (b+1)v = 0$, which imply that $u = v = 0$. Hence, $R(x,y) = T(x,y)$ is a generating function, and

$$Q(x,y) = R(x^{a+1}y^b, x^a y^{b+1}).$$

Proof of Theorem 26. It suffices to consider $k = 2$, i.e., $s = s_1 \circ s_2$. For $i = 1, 2$, let $P_i(x,y)$ be the generating polynomial of S_i. By Lemma 28, the generating polynomial of s is $P_s(x,y) = P_{s_1}(x,y)P_{s_2}\left(x^{a+1}y^b, x^a y^{b+1}\right)$, where a and b are the numbers of ones and zeroes in s_1. Let

$$P_{s_1}(x,y) = \prod_{i=1}^{k_1} A_i(x,y), \text{ and } P_{s_2}\left(x^{a+1}y^b, x^a y^{b+1}\right) = \prod_{i=1}^{k_2} B_i(x,y),$$

where $A_1, A_2, \ldots A_{k_1}$ and $B_1, B_2, \ldots, B_{k_2}$ are irreducible factors. Then

$$P_s(x,y) = \prod_{i=1}^{k_1} A_i(x,y) \prod_{i=1}^{k_2} B_i(x,y).$$

Since $s \sim t$, by Theorem 12, there exist integers $K_1 \subseteq [k_1]$ and $K_2 \subseteq [k_2]$ such that

$$P_t(x,y) = \prod_{i \in K_1} A_i(x,y) \prod_{i \in [k_1]\setminus K_1} A_i^*(x,y) \prod_{i \in K_2} B_i(x,y) \prod_{i \in [k_2]\setminus K_2} B_i^*(x,y).$$

Let

$$Q_1(x,y) \overset{\text{def}}{=} \prod_{i \in K_1} A_i(x,y) \prod_{i \in [k_1]\setminus K_1} A_i^*(x,y),$$
$$Q_2(x,y) \overset{\text{def}}{=} \prod_{i \in K_2} B_i(x,y) \prod_{i \in [k_2]\setminus K_2} B_i^*(x,y).$$

Note that $Q_1(x,y)$ and $Q_2(x,y)$ satisfy

$$Q_1(x,y)Q_1^*(x,y) = P_{s_1}(x,y)P_{s_1}^*(x,y),$$
$$Q_2(x,y)Q_2^*(x,y) = P_{s_2}(x^{a+1}y^b, x^a y^{b+1})P_{s_2}^*(x^{a+1}y^b, x^a y^{b+1}).$$

The first equation and Lemma 9 imply that $Q_1(x,y)$ is a generating polynomial. Since both P_t and Q_1 have constant term 1, $Q_2(x,y)$ also has constant term 1. Then the second equation and Lemma 47 imply that

$$Q_2(x,y) = R(x^{a+1}y^b, x^a y^{b+1}),$$

where $R(x,y)$ is a generating polynomial. By Lemma 28, $t = t_1 \circ t_2$, where t_1 has generating polynomial $Q_1(x,y)$, and t_2 has generating polynomial $R(x,y)$. $\square$

Acknowledgments. We thank Sampath Kannan, Ananda Theertha Suresh, and Alex Vardy for helpful discussions and suggestions.

REFERENCES

[1] J. Acharya, H. Das, O. Milenkovic, A. Orlitsky, and S. Pan, *On reconstructing a string from its substring compositions*, in Proceedings of the IEEE Symposium on Information Theory, 2010, pp. 1238–1242.

[2] J. Acharya, H. Das, O. Milenkovic, A. Orlitsky, and S. Pan, *Quadratic-backtracking algorithm for string reconstruction from substring compositions*, in Proceedings of the 2014 IEEE International Symposium on Information Theory (ISIT), IEEE, 2014, pp. 1296–1300.

[3] T. Batu, S. Kannan, S. Khanna, and A. McGregor, *Reconstructing strings from random traces*, in Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms, 2004, pp. 910–918.

[4] L. J. Billera, H. Thomas, and S. van Willigenburg, *Decomposable compositions, symmetric quasisymmetric functions and equality of ribbon Schur functions*, Adv. Math., 204 (2006), pp. 204–240.

[5] S. Chen, Z. Huang, and S. Kannan, *Reconstructing numbers from pairwise function values*, in Proceedings of the International Symposium on Algorithms and Computation, 2009, pp. 142–152.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge, MA, 2001.

[7] T. E. Creighton, *Proteins: Structures and Molecular Properties*, 2nd ed., W. H. Freeman, San Francisco, 1992.

[8] T. Dakic, *On the Turnpike Problem*, Ph.D. thesis, Simon Fraser University, Burnaby, Canada, 2000.

[9] A. Daurat, Y. Gérard, and M. Nivat, *Some necessary clarifications about the chords' problem and the partial digest problem*, Theor. Comput. Sci., 347 (2005), pp. 432–436.

[10] T. I. Dix and D. H. Kieronska, *Errors between sites in restriction site mapping*, Comput. Appl. Biosci., 4 (1988), pp. 117–123.

[11] M. Dudik and L. J. Schulman, *Reconstruction from subsequences*, J. Combin. Theory Ser. A, 103 (2003), pp. 337–348.

[12] G. Fici and Z. Lipták, *On prefix normal words*, in Developments in Language Theory, G. Mauri and A. Leporati, eds., Lecture Notes in Comput. Sci. 6795, Springer, New York, 2011, pp. 228–238.

[13] D. J. H. Garling, *A Course in Galois Theory*, Cambridge University Press, Cambridge, 1986.

[14] S. Kannan, *Private communication*, 2009.

[15] S. Lang, *Algebra*, 3rd ed., Grad. Texts in Math., Springer, New York, 2002.

[16] P. Lemke and M. Werman, *Inverting the Autocorrelation and the Problem of Locating Points on a Line, Given Unlabelled Distances Between Them*, Technical report 453, IMA preprint, Institute for Mathematics and Its Applications, Minneapolis, MN, 1988.

[17] A. K. Lenstra, H. W. Lenstra, and Lászlo Lovász, *Factoring polynomials with rational coefficients*, Ann. of Math., 261 (1982), pp. 515–534.

[18] V. I. Levenshtein, *Efficient reconstruction of sequences from their subsequences or supersequences*, J. of Combin. Theory Ser. A, 93 (2001), pp. 310–332.

[19] D. Margaritis and S. S. Skiena, *Reconstructing strings from substrings in rounds*, in Proceedings of the 36th Annual Symposium on Foundations of Computer Science, 1995, pp. 613–620.

[20] D. W. Mount, *Bioinformatics: Sequence and Genome Analysis*, 2nd ed., Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY, 2001.

[21] I. Niven, H. S. Zuckerman, and H. L. Montgomery, *An Introduction to the Theory of Numbers*, 5th ed., Wiley Interscience, New York, 1991.

[22] A. M. Odlyzko and B. Poonen, *Zeros of polynomials with* $0, 1$ *coefficients*, Enseign. Math., 39 (1993), pp. 317–348.

[23] A. L. Patterson, *A direct method for the determination of the components of interatomie distanees in crystals*, Zeitsehr. Krist., 90 (1935), pp. 517–554.

[24] A. L. Patterson, *Ambiguities in the X-ray analysis of crystal stryeture*, Phys. Rev., 65 (1944), pp. 195–201.

[25] L. B. Richmond and J. O. Shallit, *Counting Abelian squares*, Electron. J. Combin., 16 (2009), pp. 317–348.

[26] J. Rosenblatt and P. D. Seymour, *The structure of homometric sets*, SIAM J. Algebr. Discrete Methods, 3 (1982), pp. 343–350.

[27] S. S. Skiena, W. D. Smith, and Paul Lemke, *Reconstructing sets from interpoint distances (extended abstract)*, in Proceedings of the Sixth Annual Symposium on Computational Geometry, ACM, 1990, pp. 332–339.

[28] F. Spitzer, *Principles of Random Walk*, Grad. Texts in Math. 34, Springer, New York, 2001.

[29] E. Ukkonen, *Approximate string-matching with q-grams and maximal matches*, Theoret. Comput. Sci., 92 (1992), pp. 191–211.

[30] K. Viswanathan and R. Swaminathan, *Improved string reconstruction over insertion-deletion channels*, in Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2008, pp. 399–408.