Insufficiency of Four Known Necessary Conditions on String Unavoidability

Christine E. Heitsch

Computation and Informatics in Biology and Medicine Training Program
Departments of Chemistry and Computer Science
Mathematics Department
University of Wisconsin – Madison
Madison, WI 53706
E-mail: heitsch@math.wisc.edu

Address for correspondence:
University of Wisconsin – Madison
Mathematics Department
480 Lincoln Dr.
Madison, WI 53706-1388, U.S.A.
Email: heitsch@math.wisc.edu
Phone: +1.608.213.1734
Fax: +1.608.263.8891

Proposed Running Head:
Necessary but Insufficient Unavoidability Conditions
The reductive decision procedure for unavoidable strings was recently shown to have an exponential lower bound. Hence, as a special case of generalized pattern matching, the existence of an efficient algorithm deciding string unavoidability remains an interesting open question. It has been hypothesized that some combination of the four necessary conditions implied by the known decidability results would be sufficient. Three of these criteria are determined in polynomial time, and the fourth provides the needed recursion. In this paper, however, we demonstrate the existence of arbitrarily many unavoidable strings meeting any extended conjunction of the four necessary conditions. These insufficiency results are achieved by analyzing the appropriate graphical interpretations of the given algorithms. We provide a new combinatorial operation on the corresponding strings and generate arbitrary counterexamples from an empirically located minimal set. Thus, string unavoidability cannot be efficiently decided by the known reductive method or its immediate implications.

Key Words: unavoidable strings, pattern avoidance, blocking, Zimin words, generalized pattern matching.
1. INTRODUCTION

The majority of recent research in the area of unavoidable strings has focused on the complementary problem of avoidability \cite{5, 10, 12}. However, determining the computational complexity of deciding string unavoidableability remains an interesting open question. As was recently demonstrated \cite{7, 8}, the known reduction decision procedure of Zimin \cite{16} and Bean \textit{et al.} \cite{4} is computationally intractable. However, these results did not rule out the possibility of other, better decision procedures as hypothesized in \cite{9} which states, “there is a polynomial time algorithm verifying whether a word is avoidable (unavoidable).” This conjecture that unavoidableability can be decided efficiently was based on two necessary criteria for unavoidable strings \cite{14}: the unavoidableness of all substrings and the existence of an isolated symbol. In this paper, we consider the possibility of basing an efficient decision procedure on these conditions, along with two other. However, we show that, although necessary, the conditions are not sufficient to decide string unavoidableability.

The fundamental decision results on unavoidable strings are given in Section 2. Since deciding unavoidableability based on the existence of a reductive deletion sequence is known to be sufficient but not efficient, this material provides the necessary background but will not be the primary focus of the work presented here. Section 3 states the four necessary conditions which may be extracted from the decidability results. The fact that all substrings of an unavoidable strings must be likewise unavoidable strongly suggests a recursive decision procedure. The three other criteria impose constraints on the frequency with which symbols occur, the length of an unavoidable string, and the relationship among substrings of length two.

The obvious first attempt at an efficient decision procedure and its associated graphical interpretation is given in Section 4. This method is easily seen to produce no false negatives and even to characterize unavoidable strings having at most three symbols. It is, however, known to be insufficient for strings with four symbols. To generate arbitrarily many strings forcing false positives from the proposed decision procedure, we prove Theorem 5.1 in Section 5. This crucial result provides a new method of combining two strings where the (un)avoidability of the resulting string is completely determined by the two original ones. Thus, this theorem, coupled with empirically located minimal counter-examples, demonstrates the insufficiency of the first algorithm.

Section 6 describes the best generalization of the method from Section 4 and its corresponding interpretation as an augmentation of the original rooted binary trees. In this case, the association with unavoidable strings is not obvious, and we prove Theorem 6.1 to show that this proposed decision procedure also produces no false negatives. Although this algorithm, in
conjunction with the other necessary conditions, now determines unavoidable strings with four symbols, it fails for strings over five symbols. Hence, even this optimal method is not adequate to decide string unavoidability since the argument for Theorem 5.2 essentially applies to Theorem 6.2 as well. Finally, Sections 7 and 8 discuss the remaining issues pertaining to efforts to decide unavoidability on the basis of these necessary conditions.

2. BACKGROUND

2.1. Generalized pattern matching

Let $S$ be a finite and non-empty set of symbols. The collection of all possible non-empty strings over $S$ will be denoted $S^+$. We let $\varepsilon$ denote the empty string, and $S^+ \cup \{\varepsilon\} = S^*$. Note that $S^*$ is the set of reduced strings, since $s\varepsilon = \varepsilon s = s$ for all $s \in S$ and by extension for all $t \in S^*$. We let $|S|$ denote the size of a set $S$ and $|t|$ denote the length of a string $t$.

**Definition 2.1.** [Substring] For $t, t' \in S^+$, we say that $t$ is a **substring** of $t'$ and write $t \leq t'$ if there exist strings $u, v \in S^*$ so that $utv = t'$.

Determining whether a specific substring occurs in another given string is the standard pattern matching problem. In this context, we typically distinguish the two strings by speaking about a pattern $p$ and a text word $w$ over a set of letter symbols. In this case, we will call the set of letters our alphabet $A$ as in [11].

**Definition 2.2.** [Pattern occurrence][1] For $p, w \in A^+$, we say that the pattern $p$ **occurs** in the word $w$ if and only if $p \leq w$.

We think of the standard pattern matching question, “Does $p$ occur in $w$?,” as asking whether there exists an identity mapping between a pattern $p$ and a substring of the text word $w$, when both strings are over the same alphabet $A$. The idea of pattern occurrence as establishing a matching correspondence was previously extended with the introduction of parameterized pattern matching [2]. In this setting, there are two sets of symbols, the alphabet $A$ and the parameters $P$.

**Definition 2.3.** [Parameterized match][2] Suppose $A \cap P = \emptyset$. Two parameterized strings $p, p' \in (A \cup P)^+$ are a **parameterized match** if and only if one string can be transformed into another by applying a one-to-one function that renames the parameter symbols.

Under parameterized pattern matching the correspondence is broadened from the identity mapping to bijections of $A \cup P$ whose restriction to $A$ is the
identity. Hence, the symbols of the alphabet $A$ remain “constant.” Here we further broaden the notion of matching correspondence by introducing a new set $V$ of “variable” pattern symbols, which may map to non-empty words in $A^+$. 

**Definition 2.4.** [Generalized pattern occurrence][7] For $p \in (V \cup A)^+$, $w \in A^+$, and $V \cap A = \emptyset$, we say that $p$ occurs as a generalized pattern in $w$ and write $p \mid w$ if and only if there exists a map $\phi : (V \cup A) \rightarrow A^+$ such that $\phi(a) = a$ for all $a \in A$ and $\phi(p) \leq w$ under the induced homomorphism.

**Example 2.1.** Let $A = \{a, b, c\}$, $V = \{x\}$, $p = axaxa$, $w_1 = boabababc$, and $w_2 = abacaaa$. Then $p \mid w_1$ under the mapping $\phi(a) = a$ and $\phi(x) = bc$, but $p \nmid w_2$.

### 2.2. Unavoidable strings

The conception of unavoidable strings as a type of pattern consisting purely of variable symbols was first observed in [12]. In this context, a string is unavoidable if it occurs, as a generalized pattern with only variable symbols, in every long enough word over a finite alphabet $A$.

**Definition 2.5.** [Unavoidable] [4, 16] Consider $p \in V^+$ and finite alphabets $A_k = \{a_1, a_2, \ldots, a_k\}$ where, without loss of generality, $V \cap A_k = \emptyset$. A string $p$ is said to be unavoidable if for every integer $k \geq 1$ there exists an integer $l$ such that $p \mid w$ for all $w \in A_k^+$ with $|w| \geq l$.

We note that $l$ depends on both the string $p$ and $k$, the size of the alphabet $A_k$. We say that a string $p$ is avoidable if for some $k$ there exists arbitrarily long $w \in A_k$ such that $p \nmid w$.

**Example 2.2.** [13, 15] Let $V = \{x\}$ and $A_2 = \{a, b\}$. Consider $p = xxx$ and $w_k = \psi^k(a)$ where $\psi(a) = ab$ and $\psi(b) = ba$. Since the sequence of words, $w_1 = ab$, $w_2 = abba$, $w_3 = abbaabab$, $w_4 = abababaababababa$, \ldots, is known to have no substring which is repeated three times, $p = xxx$ is avoidable.

**Example 2.3.** Let $p = xyz$ for $V = \{x, y\}$. For a word $w$ over the finite alphabet $A_k$, if $|w| > 2k$, then $p \mid w$. Hence, $p$ is unavoidable. This is because if $p \nmid w$, then for every $a \in A_k$ only one of three possibilities holds:

1. the letter $a$ is not found in $w$, or
2. there is exactly one instance of a in w, or
3. the substring aa occurs exactly once in w.

Clearly, the longest $w \in A_k^*$ where $p \not\mid w$ is a word which contains a substring aa for every $a \in A_k$.

According to the definition, unavoidability is a characteristic inherited by substrings; if $p$ is unavaoidable, then so is every substring $q \leq p$. We will consider $\varepsilon$ to be an unavoidable string of length zero. Conversely, if $q$ is avoidable and $q \leq p$, then $p$ must also be avoidable.

Unavoidability is a decidable string characteristic, as was proved independently in [4] and [16]. As described later in this section, the common result states that the unavoidability of a string can be determined by reducing it to the empty string under an appropriate sequence of deletions. Additionally, [16] provides an equivalent unavoidability characterization in terms understood as generalized pattern occurrence.

**Definition 2.6.** [Zimin words] Let $Z_1 = a_1$ and recursively define $Z_k$ on $A_k = \{a_1, a_2, \ldots, a_k\}$ as $Z_k = Z_{k-1}a_kZ_{k-1}$. Equivalently, define a mapping $\theta : A_{k-1} \rightarrow A_k$ where $\theta(a_i) = a_{i+1}a_1$. Then $Z_k = a_1\theta(Z_{k-1})$.

**Example 2.4.**

$Z_1 = a_1$
$Z_2 = a_1 a_2 a_1$
$Z_3 = a_1 a_2 a_1 a_3 a_1 a_2 a_1$
$Z_4 = a_1 a_2 a_1 a_3 a_1 a_2 a_1 a_4 a_1 a_2 a_1 a_3 a_1 a_2 a_1$

We see that the unavoidable string $p = xyz$ is just a relabeling of the symbols of the second Zimin word. In fact, such relabeled $Z_k$ are the canonical unavoidable strings, as shown by the next theorem. (We note that the relabeling is necessitated by the requirement $(V \cap A) = \emptyset$ which distinguishes variable symbols $V$ from the constant alphabet symbols $A$ in the definition of generalized pattern matching.)

Let $\alpha(s)$ be the number of distinct symbols occurring in a string $s$.

**Theorem 2.1.** [16] Let $p \in V^+$ with $\alpha(p) = k$ and suppose, without loss of generality, that $V \cap A_k = \emptyset$. Then $p$ is unavoidable if and only if $p \mid Z_k$.

Hence, every unavoidable string on $k$ symbols has length at most $2^k - 1$, the length of $Z_k$. Furthermore, if an unavoidable string $p$ has $k$ symbols and length $2^k - 1$, then $p$ is isomorphic to $Z_k$. 
[a,x] \[x,b\] \[a,x\] \[x,b\] \[a,y\] \[y,b\]

\[p = xzx \text{ with } A = \{x\}, B = \{x\}. \quad p = xzy \text{ with } A = \{x\}, B = \{y\} \]
and \(A' = \{y\}, B' = \{x\} \].

**FIG. 1.** Two examples of \(G(p)\) and the associated two-window sets.

In [16], Theorem 2.1 is proved by showing that the matching correspondence \(\phi\) for \(p \mid Z_k\) can always be constructed according to a particular sequence of deletions. Intuitively, the following criteria permits the symbols of \(p\) to be iteratively “squeezed” into the mapping \(\phi\) as needed. The fact that the unavoidability of a string can be determined by reducing it to the empty string under an appropriate sequence of deletions was previously and independently demonstrated in [4]. Since the two sets of results use different terminology and notation, we have chosen whichever seemed the most appropriate for the purposes of this paper.

### 2.3. Deletion sequences

**Definition 2.7.** [Two-window sets] [4, 16] Given \(p \in V^+\), we say that \(A, B \subseteq V\) are two-window sets for \(p\) if and only if, for all \(xy \leq p\) with \(|xy| = 2\), \(x \in A\) if and only if \(y \in B\).

Note that we will disregard the combinations \(A = \emptyset = B\) and \(A = V = B\) which are two-window sets for any \(p\). As illustrated in Figure 1, the smallest two-window sets for a given \(p\) can be simultaneously constructed by considering its adjacency graph.

**Definition 2.8.** [Minimal \(\sigma\)-graph] [3, 7, 8] For \(p \in V^+\) and \(a, b \notin V\), let \(G(p)\) be the bipartite graph with vertices \([a, x]\) and \([x, b]\) for every symbol \(x\) in \(p\). \(([a, x], [y, b])\) is an edge in \(G(p)\) if and only if \(xy \leq p\), \(|xy| = 2\). We call \(G(p)\) the minimal \(\sigma\)-graph for \(p\).

The rationale for calling \(G(p)\) the minimal \(\sigma\)-graph of \(p\) follows from the fact that each connected component of \(G(p)\) yields a pair of sets \(A\) and \(B\) which minimally satisfy the two-window criteria for \(p\). As stated in [7] and proved in [8], these minimal sets are sufficient to generate the necessary reducing free set for a \(\sigma\)-deletion as described in the following definitions.

**Definition 2.9.** [Free set] [4, 16] The nonempty \(F \subseteq V\) is a free set for \(p \in V^+\) if and only if there exist sets two-window sets \(A, B \subseteq V\) such that \(F \subseteq B \setminus A\).
We note that $B \setminus A$ was arbitrarily chosen over $A \setminus B$; there is a dual free set definition which requires $F \subseteq A \setminus B$. According to Definition 2.9, given two-window sets $A$ and $B$, then any nonempty subset of $B \setminus A$ is a free set. Recalling that the connected components of $G(p)$ are related to the two-window sets for $p$, we see that if $G(p)$ has only one connected component, then $p$ has no free sets. In this case, as in [3], we will say that $G(p)$ is **locked**.

**Example 2.5.** The string $p = xxx$ has no free sets since the one pair of two-window sets is $A = \{x\}$, $B = \{x\}$. The string $p = xyx$ has two possible free sets, $F = \{y\} = \{y\} \setminus \{x\}$ and $F' = \{x\} = \{x\} \setminus \{y\}$.

**Definition 2.10.** $[\sigma$-deletion] [4, 16] If, and only if, $F \subseteq V$ is a free set for $p \in V^+$, we define a $\sigma$-deletion of $p$, written $\sigma_F(p)$, induced by the mapping $\sigma_F : V \rightarrow V \cup \{\varepsilon\}$ where

$$\sigma_F(x) = \begin{cases} x & \text{if } x \notin F \\ \varepsilon & \text{if } x \in F \end{cases}$$

**Example 2.6.** As before, we consider the string $p = xyx$ and its two possible free sets, $F = \{y\} = \{y\} \setminus \{x\}$ and $F' = \{x\} = \{x\} \setminus \{y\}$. Thus, $\sigma_F(p) = xx$ and $\sigma_{F'}(p) = y$.

Note that $\sigma_F(p)$ always refers to the reduced string in $V^*$. If $F$ can be $\sigma$-deleted from $p$, then we say that $\sigma_F$ is a **reduction** of $p$ to $\sigma_F(p)$. The major theorem of both [4] and [16] shows that unavoidability can be decided by the repeated application of $\sigma$-deletions.

**Theorem 2.2.** [4, 16] $p$ is unavoidable if and only if $p$ can be reduced to $\varepsilon$ by a sequence of $\sigma$-deletions.

**Example 2.7.** Let $p = xx$. Then, like the avoidable string $xxx$, $p$ has one pair of two-window sets and no free sets. Thus, $p$ cannot be reduced by any $\sigma$-deletions and so must be avoidable. Hence, rather than producing an square-free sequence, as is known to exist over a three letter alphabet [13, 15], we can show that squares are avoidable by demonstrating that $p = xx$ has no $\sigma$-deletions.
Call a \( \sigma \)-deletion and its associated free set \( F \) **worthwhile** if \( \sigma_F(p) \) is unavoidable, that is if a \( \sigma \)-deletion sequence beginning with \( \sigma_F \) reduces \( p \) to the empty string. For a string \( p \), a sequence of \( \sigma \)-deletions \( \sigma_F \) with \( 1 \leq i \leq k \leq \alpha(p) \) is said to be **complete** if \( \sigma_{F_k}(\sigma_{F_{k-1}}(\ldots \sigma_{F_1}(p) \ldots)) = \varepsilon \).

Finally, we note that other results [6, 7, 8] have shown the range of sizes which a worthwhile free set may assume; unavoidable strings \( p \) and their unique worthwhile \( \sigma \)-deletions are known to exists whose free sets \( F \) have sizes \( |F| = 1, 2, 3, \ldots, |B \setminus A| \) for two-windows sets \( A, B \) and \( F \subseteq B \setminus A \).

**Example 2.8.** We know that \( p = xyz \) has two possible \( \sigma \)-deletions, \( \sigma_F(p) = xx \) for \( F = \{y\} \) and \( \sigma_F'(p) = y \) for \( F' = \{x\} \). We see that \( \sigma \)-deleting \( F \) gives a string \( xx \) which is avoidable, and so cannot be reduced to \( \varepsilon \) by further \( \sigma \)-deletions. Thus, \( F = \{x\} \) is not a worthwhile \( \sigma \)-deletion of \( p \). However, there does exist a complete \( \sigma \)-deletion sequence beginning with \( F' \) since \( \sigma_F'(p) = y \) has the free set \( F'' = \{y\} \) for the two-window sets \( A'' = \emptyset \) and \( B'' = \{y\} \). Since \( \sigma_{F''}(y) = \sigma_{F''}(\sigma_F'(p)) = \varepsilon \), we have again demonstrated that \( p = xyz \) is unavoidable.

**Example 2.9.** A complete \( \sigma \)-deletion sequence for \( p = xyzwxyr \).

\[
\begin{array}{cccc}
F_1 &=& \{y\} & \downarrow \quad \sigma_{F_1}(p) = xyzwx \\
B_1 &=& \{w,y\} & A_1 = \{w, x, z\} \\
F_2 &=& \{x\} & \downarrow \quad \sigma_{F_2}(\sigma_{F_1}(p)) = zw \\
B_2 &=& \{x\} & A_2 = \{w, z\} \\
F_3 &=& \{w\} & \downarrow \quad \sigma_{F_3}(\sigma_{F_2}(\sigma_{F_1}(p))) = z \\
B_3 &=& \{w\} & A_3 = \{z\} \\
F_4 &=& \{z\} & \downarrow \quad \sigma_{F_4}(\sigma_{F_3}(\sigma_{F_2}(\sigma_{F_1}(p)))) = \varepsilon \\
B_4 &=& \{z\} & A_4 = \emptyset \\
\end{array}
\]

We say that if \( y \in V \) occurs exactly once in \( p \), then \( y \) is an **isolated** symbol in \( p \). We observe from the complete \( \sigma \)-deletion sequence that every unavoidable string \( p \) has at least one isolated symbol \( y \). Let \( q = \sigma_{F_{k-1}}(\ldots \sigma_{F_1}(p) \ldots) \) and suppose that \( \sigma_F(q) = \varepsilon \). Then \( q = y \) for some symbol \( y \in V \). In general, if \( xyz \leq p \) for \( x, z \in V \cup \{\varepsilon\} \), then \( y \in F \) implies \( x, z \notin F \) and \( xz \in \sigma_F(p) \). Hence, if \( \sigma_F(q) = \varepsilon \) we would have that \( x = z = \varepsilon \). We note also that, because free sets are defined as nonempty subsets of the difference of two-window sets, **at most** every other symbol can be \( \sigma \)-deleted from \( p \).
3. FOUR KNOWN NECESSARY CONDITIONS

As discussed in Section 1, our consideration of possible efficient algorithms for deciding string unavoidability is motivated by the intractability of the reductive decision procedure [7, 8]. It had been conjectured [14] that an efficient algorithm for deciding the (un)avoidability of a string $p$ over $V$ with $|V| = \alpha(p) = k$ could be obtained by combining Conditions 3.1 and 3.2.

**Condition 3.1.** If $p$ is unavoidable and $q \leq p$, then $q$ is unavoidable.

As remarked before, this condition follows immediately from the definition of unavoidability; if there exist only finitely many words $w$ over a finite alphabet $A$ such that $p \not\mid w$, then for $q \leq p$ there can exist no more (and usually fewer) words in $A^*$ with $q \not\mid w$.

**Condition 3.2.** $p$ is unavoidable only if $p$ has an isolated symbol, that is only if there exists $x \in V$ such that $p = uv$ for some $u, v \in (V \setminus \{x\})^*$.

At the end of Section 2 we discussed the fact that the last $\sigma$-deletion in a complete $\sigma$-deletion sequence of $p$ must be applied to a string of length one which is an isolated symbol occurring in $p$. In addition to the existence of an isolated symbol, there are two other conditions on unavoidable strings which are easy to check and which help in deciding the unavoidability of at least some strings as explained in Section 5. There may well be other necessary conditions on unavoidable strings besides these four which are also efficient to determine, although at this point they are “unknown” since there are no other obvious candidates. Hence, these four conditions form the current natural candidates for the basis of an efficient algorithm deciding string (un)avoidability.

**Condition 3.3.** $p$ is unavoidable only if $|p| \leq 2^k - 1$. If $|p| = 2^k - 1$, then $p$ is isomorphic to $Z_k$ under a one-to-one mapping $\phi : V \rightarrow A_k$.

This follows from Theorem 2.1 which says that that if $p$ is unavoidable, then $p \mid Z_k$. Finally, there can be no initial $\sigma$-deletion of $p$ if $G(p)$ has only one connected component since, for any sets $A$ and $B$ satisfying the two-window criteria for $p$, $B \setminus A = \emptyset$.

**Condition 3.4.** $p$ is unavoidable only if $G(p)$ is not locked, that is only if $G(p)$ has at least two connected components.

The last three conditions are easily checkable in time polynomial in the length of $p$, while the first strongly suggests a recursive algorithm for the
(un)avoidability question. However, as we will show, an efficient decision procedure is not so easily extracted from these necessary conditions.

Although an unavoidable string must necessarily satisfy the previous four conditions, they are far from sufficient to establish avoidability on their own. There are many, in fact most, strings which satisfy the length criteria but which are avoidable. The existence of an isolated variable does not guarantee avoidability, nor does the avoidability of all substrings, nor even the existence of possible ω-deletions from the different connected components of $G(p)$. Furthermore, we will show that in combination these properties are provably insufficient to decide avoidability, and thus that the decision results [4, 16] of Theorems 2.2 and 2.1 remain the only known necessary and sufficient conditions for avoidability.

4. AN IMMEDIATE IMPLEMENTATION

As has been observed before [14], Conditions 3.1 and 3.2 naturally point to the possibility of a simple recursive algorithm deciding the avoidability of a string $p \in V^+$. 

**Algorithm 4.1.** If $p = \varepsilon$, accept. Else if there exists $x \in V$ such that $p = p_1xp_2$ with $p_1, p_2 \in (V \setminus \{x\})^*$, then repeat for $p_1$ and for $p_2$.

We say that Algorithm 4.1 accepts a string $p$ if and only if every leaf of the computation tree is an acceptance. A more explicit pseudocode implementation is given in Figures 2 and 3. Clearly, if $p$ is accepted by Algorithms 4.1 and $q \leq p$, then $q$ is also accepted. Note the independent choice of isolated symbols across substrings in the recursion of Algorithm 4.1. Otherwise, as can be seen in Figure 4, unavoidable strings such as $p = xyxzx$ would be falsely rejected. In such cases, it is not possible to break $p = p_1xp_2$ into substrings $p_1, p_2 \in (V \setminus \{x\})^+$ which both contain the same isolated symbol. (For further discussion of the possibilities in the dependent case, refer to Section 8.) As it is, we easily see by induction and Conditions 3.1, 3.2 that Algorithm 4.1 produces no false negatives in deciding string avoidability.

**Lemma 4.1.** If $p$ is unavoidable, then $p$ is accepted by Algorithm 4.1.

Despite the appealing simplicity of Algorithm 4.1, we show that it returns an arbitrary number of false positives, even in conjunction with Conditions 3.3 and 3.4. Hence, this combination of the four necessary conditions is provably insufficient for a decision procedure. The results rely on exploiting a connection between unavoidable strings and labeled rooted binary trees.
Algorithm 1 (Simple(p)).

begin
    if \(|p| \leq 1\) then
        return TRUE;
    isosym := Isolated(p);
    if isosym = 0 then
        return FALSE;
    i := 0;
    while \(p[i] \neq isosym\) do
        \(p_1[i] := p[i]\);
        i := i + 1;
    end while
    i := i + 1;
    while i < |p| do
        \(p_2[i] := p[i]\);
        i := i + 1;
    end while
    return Simple(p_1) and Simple(p_2);
end

FIG. 2. A pseudocode implementation of Algorithm 4.1, where the input has been normalized to strings over \(\{1, 2, 3, \ldots, k\}\). A subroutine for calculating the length of a string \(|p|\) is assumed. The complexity of this procedure including the subroutine from Figure 3 is \(O(n^3)\).
Algorithm 2 (Isolated($p$)).

begin
  i := 0;
  while i < $|p|$ do
    cnt[i] := 0;
    i := i + 1;
  end while
  i := 0;
  while i < $|p|$ do
    i := i + 1;
  end while
  i := 0;
  while i < $|p|$ do
    if cnt[i] = 1 then
      return i + 1;
    end if
    i := i + 1;
  end while
  return 0;
end

FIG. 3. A subroutine for the pseudocode implementation of Algorithm 4.1 in Figure 2, where the input has been normalized to strings over $\{1, 2, 3, \ldots, k\}$. This subroutine returns the first isolated symbol in $p$ or 0 if there is none. As discussed following Lemma 4.2, the algorithm's output is not impacted by choosing among different possible isolated symbols.

FIG. 4. The two possible computation trees for $p = xyxx$ under Algorithm 4.1, depending on first isolated symbol chosen. The edges are labeled with the substrings $p_1$ and $p_2$ at each stage of the recursion.
In Figure 4, we labeled the edges of a rooted binary tree representing the processing of a string $p$ by Algorithm 4.1. Now, rather than labeling the edges of the trees, we label the nodes with the isolated symbols as in Figure 5. In this way, we capture the processing of a string by Algorithm 4.1 as a rooted binary tree with the nodes appropriately labeled by symbols from $V$. More generally, we recursively define the following classes of tree structures over a set of symbols $V$ by specifying a left subtree, root node, and right subtree.

**Definition 4.1.** [V-trees] We say that $\mathcal{B}(V)$ is the set of **V-trees** and define $\mathcal{B}(V) = \{(B_L, x, B_R) : x \in V, B_L, B_R \in \mathcal{B}(V \setminus \{x\}) \cup \mathcal{B}(\emptyset)\}$ for $|V| \geq 1$ and $\mathcal{B}(\emptyset) = \{\emptyset\}$.

Clearly, the maximum number of nodes in a V-tree is $2^{|V|} - 1$. The only restriction placed on node labeling is that $x$ may not appear again in any of its subtrees. It may, however, appear in any other branches of the binary tree not following from $x$ as a root node. As long as the consistency of the node labeling is preserved, V-trees may be created from, as well as disassembled into, left and right subtrees. Using an in-order walk along the nodes of a V-tree, we return to a string over the set $V$ by defining a mapping $\pi$ which associates to each V-tree a unique (reduced) string in $V^*$.

**Definition 4.2.** Let $B = (B_L, x, B_R) \in \mathcal{B}(V)$. Define $\pi : \mathcal{B}(V) \rightarrow V^*$ by $\pi(B) = \pi(B_L)x \pi(B_R)$ where $\pi(\emptyset) = \varepsilon$.

As illustrated by Figure 5, this correspondence between strings and V-trees is certainly not unique. In an extreme case, let $f(n)$ be the number of distinct V-trees representing $p = x_1 x_2 \ldots x_n$, when each of the $x_i$ symbols occurs exactly once. Then $f(0) = 1$, $f(1) = 1$, $f(2) = 2$, $f(3) = 5$, . . . so that $f(n) = C_n$ where $C_n = \sum_{i=1}^{n} f(i-1)f(n-i) = \frac{1}{n+1}(2^n)$ is the $n$th Catalan number. In the V-tree case, summing over $i$ represents the
i = 1, . . . , n different choices for the first isolated symbol. Not withstanding
the different possible V-trees representing the same string p, we show that
any string recognized by Algorithm 4.1 has at least one corresponding V-
tree, and vice versa.

**Lemma 4.2.** \( p \in V^+ \) is accepted by Algorithm 4.1 if and only if \( p = \pi(B) \)
for some \( B \in B(V) \).

**Proof.** By induction on \( \alpha(p) = n + 1 = |V| \) for \( p \in V^+ \). Clearly, if
\( V = \{x\} \), then \( p \) is accepted by Algorithm 4.1 if and only if \( p = \pi(B) = x \)
for \( B = (\emptyset, x, \emptyset) \in B(\{x\}) \).

According to Definitions 4.1 and 4.2, if \( p \in V^+ \) is accepted by Algo-
mith 4.1, then a suitable \( B \in B(V) \) may be generated from the isola-
ted symbols chosen at each recursive step of an accepting computation.
Suppose \( p \) is accepted so there exists \( x \in V \) such that \( p = p_1xp_2 \) and
\( p_1, p_2 \in (V \setminus \{x\})^+ \). We know that Algorithm 4.1 accepts \( p_1, p_2 \leq p \) with
\( \alpha(p_1), \alpha(p_2) \leq n \) so there exist \( B_1, B_2 \in B(V \setminus \{x\}) \) with \( \pi(B_1) = p_1 \) and
\( \pi(B_2) = p_2 \). We let \( B = (B_1, x, B_2) \in B(V) \) with \( \pi(B) = \pi(B_1)x\pi(B_2) = p_1xp_2 \).

Conversely, suppose \( p = \pi(B) \) for some \( B = (B_L, x, B_R) \in B(V) \). Assume
that \( y \neq x \) is the isolated symbol chosen at the first call of Algo-
mith 4.1 so that \( p = p_1yp_2 \) with \( p_1, p_2 \in (V \setminus \{y\})^+ \). To apply
the induction argument, we must show that there exist \( B_1, B_2 \in B(V \setminus \{y\}) \)
such that \( \pi(B_1) = p_1 \) and \( \pi(B_2) = p_2 \). We know that a unique \( y \) node
must occur in \( B \) and, without loss of generality, assume that \( y \) is in \( B_R \).
We let \( p = q_1xq_2yp_2 \) where \( p_1 = q_1xq_2 \) and \( \pi(B_R) = q_2yp_2 \), \( \pi(B_L) = q_1 \) for
\( B_R, B_L \in B(V \setminus \{x\}) \). By assumption, \( \pi(B_R) \) is accepted by Algorithm 4.1,
and hence so are its substrings \( q_2 \) and \( p_2 \). Likewise, for \( \pi(B_L) \) and \( q_1 \). We
note that because \( y \) is isolated in \( p \), \( y \) does not occur in \( q_1, q_2, \) or \( p_2 \). Hence,
there exist \( L, R, R' \in B(V \setminus \{x, y\}) \) such that \( \pi(L) = q_1 \), \( \pi(R) = q_2 \),
and \( \pi(R') = p_2 \). Let \( B_1, B_2 \in B(V \setminus \{y\}) \) be \( B_1 = (L, x, R) \) and \( B_2 = R' \).
Then \( \pi(B_1) = \pi(L)x\pi(R) = q_1xq_2 = p_1 \) and \( \pi(B_2) = \pi(R') = p_2 \). Since
by assumption \( p_1 \) and \( p_2 \) are accepted by Algorithm 4.1, so is \( p = p_1yp_2 \). □

Hence, the strings accepted by Algorithm 4.1 are exactly those \( p = \pi(B) \)
for some \( B \in B(V) \). We had mentioned previously that the execution of a specific implementation of Algorithm 4.1 depends on the choice of an
isolated symbol. For instance, the procedure Isolated(p) given in Figure 3
returns the first. By the proof of Lemma 4.2, the particular isolated symbol
chosen by a specific implementation impacts only the rate at which a string
is accepted. We know from Lemma 4.1 that every unavoidable \( p \in V^+ \)
is accepted by Algorithm 4.1 and from Lemma 4.2 that there exists at
least one \( B \in B(V) \) with \( \pi(B) = p \). To determine whether Algorithm 4.1
decides string unavoidability, we need to answer the question: do there exist \( B \in \mathcal{B}(V) \) such that \( \pi(B) \) is avoidable even though it is accepted by Algorithm 4.1?

**Definition 4.3.** Define \( \mathcal{T}(p) = \{ B \in \mathcal{B}(V) : \pi(B) = p \} \).

**Example 4.1.** As we saw in Figure 5 on page 14, for \( V = \{x, y, z\} \) and \( p = xyzxz \), \( \mathcal{T}(p) = \{ B = (x, y, z, x), B' = ((x, y, x), z, x) \} \).

**Lemma 4.3.** Suppose that \( p \) is unavoidable. Then \( \mathcal{T}(p) \neq \emptyset \).

As illustrated by Figure 6, it is immediately obvious that the converse to Lemma 4.3 is false. Although \( |\pi(B)| \leq 2^k - 1 \) when \( |V| = k \), there are many \( B \) where \( |\pi(B)| = 2^k - 1 \) but \( \pi(B) \) is not isomorphic to \( Z_k \). However, we can easily satisfy Condition 3.3 simply by restricting consideration to \( B \in \mathcal{B}(V) \) having strictly less than \( 2^k - 1 \) nodes. When \( |V| = 3 \), all \( B \in \mathcal{B}(V) \) satisfying this additional length constraint yield one of the 14 nonisomorphic unavoidable \( \pi(B) \in V^+ \) listed in Table 1. When \( |V| = 4 \), however, it is possible to determine empirically that there are \( \pi(B) \) which satisfy all four conditions, yet which are avoidable.

Specifically, among strings having four distinct symbols, there are 1700 nonisomorphic \( p \) with nonempty \( \mathcal{T}(p) \) and which hence satisfy a generalization of the isolated symbol requirement of Condition 3.2. However, we know that there are only 438 unavoidable strings with 4 symbols. Of the remaining 1262 avoidable strings, 23 fail the length criteria. Only 133 of the rest also meet Condition 3.4 by having an unlocked \( G(p) \) with two or more connected components. Finally, of those 133 exactly 31 are of the form \( p = x_1qx_j \) where both \( x_1 \), \( q \) and \( qx_j \) are unavoidable, implying that all proper substrings of \( p \) are unavoidable as well. Thus, as listed in Table 2,
there exist 31 nonisomorphic $p$, where $\alpha(p) = 4$, which satisfy our four necessary conditions for unavoidability but which are actually avoidable.

These minimal counterexamples will be used to generate arbitrary strings forcing false positives from Algorithm 4.1. We know that efficient string searching techniques would rapidly disqualify any $q$ containing one of a finite number of problematic $p = \pi(B)$ as a substring. Hence, the operation of using two V-trees as the right and left subtrees to form a third is not adequate for this purpose. However, we exploit another operation, that of “expanding” a node by replacing it with another binary tree. V-tree consistency above and below the expanded node(s) is easily insured by introducing new node labels. Moreover, we prove that the analogous operation preserves string unavoidability.

5. REPLACEMENT PRESERVES UNAVOIDABILITY

We want to define an operation on strings which corresponds to our intuitive notion of replacing all instances of a symbol $y$ in a string $p \in V^+$ with another string $q \in U^+$.

**Definition 5.1.** Let $p \in V^+$ and $q \in U^+$. Suppose $y$ is a symbol in $p$ with $(V \setminus \{y\}) \cap U = \emptyset$. Define $\rho(p, y, q)$ to be the string obtained by replacing every instance of $y$ in $p$ by the string $q$.

**Example 5.1.** Let $V = \{x, y, z\}$, $p = xyzyx$, $U = \{y, w\}$ and $q = yw$. Then $\rho(p, y, q) = xywzywx$ whereas $\rho(q, y, p) = xyzyzw$.

As illustrated in Figure 7, because $(V \setminus \{y\}) \cap U = \emptyset$, a $(V \cup U)$-tree for $\rho(p, y, q)$ can be produced from a $V$-tree corresponding to $p$ and a $U$-tree associated with $q$. Intuitively, we simply replace every instance of a $y$ node by the appropriate $U$-tree and attach the original left and right subtrees to the leftmost and rightmost children of the inserted tree.
$B \in B(V)$ with $\pi(B) = p = xyzyx$.

$B' \in B(U)$ with $\pi(B') = q = yw$

The substrings $p_L = xy$ and $p_R = yx$ are used inductively to obtain $B'' \in B(V \cup U)$ with $\pi(B'') = \rho(p, y, q) = xyzyxw$.

To obtain $B'' \in B(V \cup U)$ with $\pi(B'') = \rho(q, y, p) = xyzyw$, the rightmost subtree $(\emptyset, x, \emptyset)$ of $B$ is replaced by $(\emptyset, x, w)$.

**FIG. 7.** The trees corresponding to the strings in Example 5.1 according to the construction given in Lemma 5.1.
Lemma 5.1. Consider $\rho(p, y, q)$ when $p = \pi(B)$ for $B \in \mathcal{B}(V)$, $q = \pi(B')$ for $B' \in \mathcal{B}(U)$, and $(V \setminus \{y\}) \cap U = \emptyset$ with $y \leq p$. There exists $B'' \in \mathcal{B}(V \cup U)$ such that $\rho(p, y, q) = \pi(B'')$.

Proof. Let $B = (B_L, x, B_R) \in \mathcal{B}(V)$ and suppose $x = y$. Let $q = wq'v$ for $u, v \in U$ and consider the corresponding subtrees $(\emptyset, u, \emptyset)$ and $(L_v, v, \emptyset)$ in $B' \in \mathcal{B}(U)$. We obtain $B'' \in \mathcal{B}(V \cup U)$ from $B' \in \mathcal{B}(U)$ by replacing the subtree $(\emptyset, u, \emptyset)$ by $(B_L, u, B_u)$ and the subtree $(L_v, v, \emptyset)$ by $(L_v, v, B_R)$. By construction, $\pi(B'') = \pi(B_L)\pi(B')\pi(B_R) = \rho(p, y, q)$.

Now we induct on $|p| = n + 1$ for $n \leq 2^{|V|-2}$. Suppose $p = p_Lzp_R$ for $z \neq y$, $p_L = \pi(B_L)$ and $p_R = \pi(B_R)$ with $|p_L|, |p_R| \leq n$. Let $l = \rho(p_L, y, q)$ if $y \leq p_L$ and $p_L$ otherwise. Likewise, let $r = \rho(p_R, y, q)$ if $y \leq p_R$ and $p_R$ otherwise. We have that $\rho(p, y, q) = lzx$. By induction, if necessary, there exists $B''_L, B''_R \in \mathcal{B}(V \cup U)$ such that $\rho(p_L, y, q) = \pi(B''_L)$ and $\rho(p_R, y, q) = \pi(B''_R)$. Otherwise, let $B''_L = B_L$ and $B''_R = B_R$. Then for $B'' = (B''_L, x, B''_R) \in \mathcal{B}(V \cup U)$, $\pi(B'') = \pi(B''_L)x\pi(B''_R) = \rho(p, y, q)$. 

Thus, $\rho(p, y, q)$ captures the idea of replacing every node $y$ in a $V$-tree corresponding to $p$ by a $U$-tree associated with $q$. The restriction that $(V \setminus \{y\}) \cap U = \emptyset$ optimally insures the consistency of the resulting $(V \cup U)$-tree for all possible cases. Not only does the $p$ operation preserve acceptance under Algorithm 4.1, but it also preserves the unavailability of strings $p$ and $q$.

Theorem 5.1. $\rho(p, y, q)$ is unavoidable if and only if both $p$ and $q$ are.

Proof. Let $r = \rho(p, y, q)$ for $y \in V^+$, $q \in U^+$ and $(V \setminus \{y\}) \cap U = \emptyset$.

The necessary unavailability of both $p$ and $q$ can be easily seen. Clearly, $q \leq r$ so $r$ cannot be unavoidable if $q$ is avoidable. Suppose that $\rho(p, y, q)$ is unavoidable. Then for any finite alphabet $A_k$ there exists an integer $l$ such that $r \mid w$ for all $w \in A_k^+$ with $|w| \geq l$. Given $r \mid w$, let $\phi : V \cup U \to A_k^+$ be the mapping such that $\phi(r) \leq w$. Define $\phi' : V \to A_k^+$ by $\phi'(x) = \phi(x)$ for $x \in V, x \neq y$ and $\phi'(y) = \phi(q)$. Then $\phi'(p) = \phi(r) \leq w$ and $p$ cannot be avoidable either.

Assume, then, that both $p$ and $q$ are unavoidable. We construct a $\sigma$-deletion sequence which reduces $r$ to the empty string from the complete $\sigma$-deletion sequences of $p$ and $q$, respectively $\sigma_{E_{ij}}$ for $1 \leq i \leq k$ and $\sigma_{G_{ij}}$, for $1 \leq j \leq m$. We suppose, without loss of generality, that $y \in U$ and that $\sigma_{E_{m-1}}(\ldots(\sigma_{G_i}(q) \ldots) = y$.

Let $\equiv$ be the equivalence relation on the vertices of a minimal $\sigma$-graph where two vertices are equivalent if and only if they are in the same connected component.
We show that the $\sigma_{G_j}$, except for the final $\sigma_{G_m}$, can be inserted into the $\sigma_{F_r}$ sequence immediately preceding the first $\sigma_{F_{r'}}$ for which $[a, y] \neq [y, b]$ in the minimal $\sigma$-graph of $\sigma_{F_{r'}} \ldots \sigma_{F_1}(p) \ldots$. If we can $\sigma$-delete the $\sigma_{F_r}$ from $r$ for $1 \leq i < k'$ followed by the $\sigma_{G_j}$ for $1 \leq j < m$, then we have

$$\sigma_{G_{m-1}} \ldots (\sigma_{G_1}(\sigma_{F_{r'}} \ldots (\sigma_{F_1}(r) \ldots))) \ldots = \sigma_{F_{r'}} \ldots (\sigma_{F_1}(p))$$

and hence the $\sigma_{F_r}$ for $k' \leq i \leq k$ would finish reducing $r$ to $\varepsilon$.

Without loss of generality, we need consider only whether $[a, y]$ and $[y, b]$ are in the same connected component of $G(p)$, the minimal $\sigma$-graph of $p$.

This is because if $[a, y] = [y, b]$ in $G(p)$ and if $F_1$ may be $\sigma$-deleted from $r$, then $\sigma_{F_1}(r) = \rho(\sigma_{F_1}(p), y, q)$. Whereas, if $[a, y] \neq [y, b]$ in $G(p)$ and if $G_1$ can be $\sigma$-deleted from $r$, then we know that $\sigma_{G_1}(r) = \rho(p, y, \sigma_{G_1}(q))$.

We begin by considering the relationship among the minimal $\sigma$-graphs of $r$, $p$, and $q$ as sketched in Figure 8 on page 21. Suppose that $q = sq_1 t$, and recall that $xyz \leq p$ if and only if $xqz \leq r$. Hence, $G(r)$ can be obtained from $G(p)$ by removing the two vertices $[a, y]$, $[y, b]$, inserting $G(q)$, and replacing all edges $([a, y], [z, b])$, $([a, x], [y, b])$ with $([a, t], [z, b])$, $([a, x], [s, b])$. Because $(V \setminus \{y\}) \cap U = \emptyset$, these new edges are the only connection between what remains of $G(p)$ and the $G(q)$ subgraph in $G(r)$.

Now suppose that $[a, y] \neq [y, b]$ in $G(p)$. We show that $G_1$ can be $\sigma$-deleted from $r$ by providing sets $A, B \subseteq (V \cup U)$ such that $A, B$ satisfy the two-window criteria for $r$ and $G_1 \subseteq B \setminus A$.

$$A = \{ x \in V \cup U : [a, x] \equiv [g, b] \in G(r) \text{ for some } g \in G_1 \}$$
$$B = \{ y \in V \cup U : [y, b] \equiv [g, b] \in G(r) \text{ for some } g \in G_1 \}$$

Note that, as the projections of the right and left sides of the connected component(s) containing $[g, b]$ in $G(r)$, we know that $A$ and $B$ minimally satisfy the two-window criteria. Clearly, $G_1 \subseteq B$. We must verify that $G_1 \cap A = \emptyset$.

Suppose not. Then there exists $g, h \in G_1 \subset U$ such that $[a, h] \equiv [g, b]$ in $G(r)$. However, because $G_1$ may be $\sigma$-deleted from $q$, $[a, h] \neq [g, b]$ in $G(q)$. Hence, $[a, h]$ and $[g, b]$ cannot be connected through the edges of the $G(q)$ subgraph in $G(r)$. The path from $[a, h]$ to $[g, b]$ in $G(r)$ must involve vertices outside the $G(q)$ subgraph; it must be that $[a, h]$ is connected to either $[a, s]$ or $[t, b]$ and that $[g, b]$ is connected to $[a, s]$ or $[t, b]$ in $G(q)$ for $q = sq_1 t$. If $[a, h] \equiv [a, s] \equiv [g, b]$, and likewise for $[t, b]$, then $[a, h]$ and $[g, b]$ are in the same connected component of $G(q)$, which is a contradiction. However, it cannot be that $[a, s]$ is connected to $[t, b]$ through edges in the rest of $G(r)$ since, by assumption, $[a, y] \neq [y, b]$ in $G(p)$. Hence, $G_1 \subseteq B \setminus A$ and $G_1$ may be $\sigma$-deleted from $r$.

Suppose instead that $[a, y] \equiv [y, b]$ in $G(p)$, implying that $y \notin F_1$. Since $[a, y]$ and $[y, b]$ are already connected in $G(p)$, the insertion of $G(q)$ to form
\(\mathcal{G}(r)\)

\(\mathcal{G}(p)\)

\(\mathcal{G}(q)\)

\(\mathcal{G}(r)\) can introduce no new connections between the vertices of what remains of \(\mathcal{G}(p)\). Hence, essentially the same argument as before shows that \(F_1 \subset V\) may be \(\sigma\)-deleted from \(r\) in this case.

We have shown that essential connected components remain disconnected in \(\mathcal{G}(r)\) if they were in \(\mathcal{G}(q)\) or \(\mathcal{G}(p)\). Hence, \(F_1\) may be \(\sigma\)-deleted from \(r\) if \([a, y] \equiv [y, b]\) in \(\mathcal{G}(p)\) and \(G_1\) may be \(\sigma\)-deleted from \(r\) if that is not the case. Thus, \(r\) is reduced to \(\varepsilon\) by the appropriate application of the complete \(\sigma\)-deletion sequences for \(q\) and \(p\). Consequently, the unavoidability of both \(p\) and \(q\) is sufficient, as well as necessary, for the unavoidability of \(\rho(p, y, q)\).

Since unavoidability is preserved under replacement, we use this technique to generate arbitrarily many different strings foiling Algorithm 4.1. Although these strings meet the four necessary conditions for unavoidability given in Section 3, by construction they are avoidable.

**Theorem 5.2.** Let \(p \in V^+\) be avoidable. Suppose further that

1. \(p\) has nonempty \(T(p)\),
2. \(p\) has unlocked \(\mathcal{G}(p)\),
3. \(|p| < 2^{\alpha(p)} - 1\),
4. and all proper substrings of \(p\) are unavoidable.
Then there exist infinitely many avoidable $r$ which also satisfy the four conditions above.

Proof. Recall that if all proper substrings of $p$ are unavoidable, then $p$ must be of the form $p = xp_1z$ for $x, z \in V$, where $xp_1$ and $p_1z$ are unavoidable for $p_1 \in V^*$. Choose an unavoidable $q \in U^+$ such that $\{y\} = V \cap U$ with $y \neq x, y \neq z$. Observe that any unavoidable $q$ satisfies the four properties listed above. Let $r = \rho(p, y, q)$.

According to Lemma 5.1, $\mathcal{T}(r) \neq \emptyset$. One of the implications of the methods used to prove Theorem 5.1 is that if both $p$ and $q$ have unlocked minimal $\sigma$-graphs, then $G(r)$ will also have at least two connected components. Let $\alpha(p) = k$ and $\alpha(q) = m$. Then $r$ has $k + m - 1$ distinct symbols. By assumption $|q| < 2^m - 1$, $|p| < 2^k - 1$ and we know that at most every other symbol of $p$ can be $y$. Hence, let $i$ be the number of instances of $y$ in $p$ which must be strictly less than $2^{k-1}$. Consequently, $|r| = |q| + |p| - i < [(2^m - 1) - 1](2^{k-1}) + (2^k - 1) < 2^{k+m-1} - 1$. Finally, since $\rho(p, y, q) = x\rho(p_1 z, y, q) = \rho(xp_1, y, q)z$ all proper substrings of $r$ must be unavoidable, but $r$ itself cannot be.

Starting with an avoidable $p$ which satisfies all four criteria and any set of unavoidable $q_i$, we can construct arbitrarily many different $r_k = \rho \cdots (\rho(p, y_1, q_1) \ldots y_k, q_k)$, with $p \not\subset r_k$, which likewise satisfy the four criteria, but which are also avoidable. □

All of the 31 strings listed in Table 2 meet the assumptions of Theorem 5.2, and hence provide multiple base cases for such an inductive construction. Thus, Algorithm 4.1 would return a false positive for an arbitrary number of distinct avoidable strings. Moreover, the four known necessary conditions cannot be used to rectify this failure as these counterexamples meet all four conditions.

As a final note, we observe that the strings $\rho(p, y, q)$ do have $p$ as a subsequence; the symbols of $p$ appear sequentially, but not consecutively, in $\rho(p, y, q)$. While a substring is obtained by removing zero or more symbols from the beginning and end of the string, a subsequence is created by deleting zero or more symbols from anywhere in the string. Finding the longest common subsequence of two strings is a well-known example of the efficiency of dynamic programming. However, as the following results will enable us to demonstrate in Section 7, subsequences in general — without the restriction that they be obtained from a $\sigma$-deletion — are not useful in deciding unavoidability.

6. AN OPTIMAL COMBINED ALGORITHM
We begin by considering again Condition 3.2, which states that each unavoidable string must have an isolated symbol. Thus, if \( p \in V^+ \) is unavoidable there exists \( x \in V \) such that \( p = p_1xp_2 \in V^+ \) where \( p_1, p_2 \in (V \setminus \{x\})^+ \). Since both \( p_1 \) and \( p_2 \) must be unavoidable strings, each one has its own isolated symbol. However, since the same symbol may not be isolated in both \( p_1 \) and \( p_2 \), the choice of an isolated symbol across different branches of Algorithm 4.1’s recursive levels was made independently. This corresponded to the lack of any \( V \)-tree requirement that the root nodes of subtrees match, necessitated by the existence of unavoidable strings like \( xyzzz \). In the previous section, though, we proved that Algorithm 4.1 and the \( V \)-tree characterization were not sufficient to decide string unavoidability.

Now, we will consider another generalization of Conditions 3.2 and 3.1. Previously, we could not require that the same symbol be isolated across all substrings in the recursion of Algorithm 4.1 without generating unacceptable false negatives. In Algorithm 6.1, we have a more sophisticated procedure which requires that the same symbol be isolated or nonexistent across all the current substrings. Since the results of Section 5 demonstrated that a simple recursive application of Condition 3.2 is not sufficient to determine unavoidability, the following algorithm is the best way to apply the isolated symbol requirement simultaneously across all substrings in the recursion.

**Algorithm 6.1.** Begin with \( q_1 = p \neq \varepsilon, q_2 = \ldots = q_n = \varepsilon \) for \( |p| = n \). If \( q_i = \varepsilon \) for all \( 1 \leq i \leq n \), accept. Else if there exists \( x \) such that \( x \) is isolated in \( q_j \) for some \( j, 1 \leq j \leq n \), and \( x \) is isolated or nonexistent in \( q_i \) for all \( 1 \leq i \leq n \), then for any \( i \) with \( q_i = q'_ixq''_i \) update \( q_i = q'_i \) and, if \( q''_i \neq \varepsilon \), \( q_{i+|q'_i|+1} = q''_i \) and repeat.

**Example 6.1.** We consider the processing of \( p = xyzzz \) under Algorithm 6.1 according to the contents of \( q_i \) for \( 1 \leq i \leq |p| \) as listed in the following table. In Step 1, \( y \) is chosen as the first isolated symbol from the initial \( q_1 = p, q_2 = \ldots = q_5 = \varepsilon \). Then in Step 2, \( z \) is chosen since it is isolated in \( q_3 \) and nonexistent in all other \( q_i \). In Step 3, the remaining symbol \( x \) is removed from the \( q_i \) and we are left with all empty strings by Step 4.

<table>
<thead>
<tr>
<th>Step #</th>
<th>( q_1 )</th>
<th>( q_2 )</th>
<th>( q_3 )</th>
<th>( q_4 )</th>
<th>( q_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( xyzzz )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
</tr>
<tr>
<td>2</td>
<td>( x )</td>
<td>( \varepsilon )</td>
<td>( xzzz )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
</tr>
<tr>
<td>3</td>
<td>( x )</td>
<td>( \varepsilon )</td>
<td>( x )</td>
<td>( \varepsilon )</td>
<td>( x )</td>
</tr>
<tr>
<td>4</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
</tr>
</tbody>
</table>
Algorithm 3 (Complex(p)).

\begin{algorithm}
\textbf{begin}
\textbf{sum} := 0; \textbf{i} := 0; \textbf{l} := |p|; \textbf{k} := \alpha(p); \\
\textbf{while} \textbf{i} < \textbf{l} \textbf{do}
\quad \textbf{sum} := \textbf{sum} + \textbf{p}[\textbf{i}]; \\
\quad \textbf{i} := \textbf{i} + 1; \\
\textbf{end while}
\textbf{isolates} := \text{Isolated}(p, l, k); \\
\textbf{while} \textbf{sum} > 0 \textbf{and} \textbf{isolates} \neq 0 \textbf{do}
\quad \textbf{i} := 0; \\
\quad \textbf{while} \textbf{i} < \textbf{l} \textbf{do}
\quad \quad \textbf{if} \textbf{p}[\textbf{i}] = \textbf{isolates} \textbf{then}
\quad \quad \quad \textbf{p}[\textbf{i}] := 0; \\
\quad \quad \quad \textbf{i} := \textbf{i} + 1; \\
\quad \textbf{end while}
\quad \textbf{sum} := 0; \textbf{i} := 0; \\
\quad \textbf{while} \textbf{i} < \textbf{l} \textbf{do}
\quad \quad \textbf{sum} := \textbf{sum} + \textbf{p}[\textbf{i}]; \\
\quad \quad \textbf{i} := \textbf{i} + 1; \\
\quad \textbf{end while}
\quad \textbf{isolates} := \text{Isolated}(p, l, k);
\textbf{end while}
\textbf{if} \textbf{sum} = 0 \textbf{then}
\quad \textbf{return} \text{TRUE};
\textbf{return} \text{FALSE};
\textbf{end}
\end{algorithm}

\textbf{FIG. 9.} A pseudocode implementation of Algorithm 6.1, where the input strings have again been normalized to the alphabet \{1, 2, 3, \ldots, k\}. A subroutine for calculating the length of a string \(p\), \(|p|\), and the number of symbols, \(\alpha(p)\), is assumed. The complexity of this procedure including the subroutine from Figure 10 is \(O(n^3)\).
Algorithm 4 (isolated(p, l, k)).

begin
  i := 0;
  while i < l do
    j := 0;
    while j < k do
      cnt[i][j] := 0;
      j := j + 1;
    end while
    i := i + 1;
  end while
  i := 0; j := 0;
  while j < l do
    if p[j] = 0 then
      i := i + 1;
      j := j + 1;
    end while
    j := 0;
  end while
  i := 0; fnd := FALSE; num := 0;
  while num < 2 and i < l do
    num := cnt[i][j];
    if num = 1 and fnd := FALSE then
      fnd := TRUE;
      i := i + 1;
    end while
    if fnd = TRUE and i = l and num < 2 then
      return j + 1;
    end if
    j := j + 1;
  end while
return 0;
end

FIG. 10. A subroutine for the pseudocode implementation of Algorithm 6.1 in Figure 9, where the input has been normalized to strings over \{1, 2, 3, \ldots, k\}.
Fig. 11. The two possible augmented V-trees for $p = xyxzx$, $L = (y, (x, z), (x, x, x))$ and $L' = ((x), (y, z), (x, x, x))$, where the ε nodes have been left unlabeled.

We note that if $q_i$ is nonempty, then it is a substring of $p$ beginning at position $i$. Algorithm 6.1 accepts a string $p$ if and only if it can be repeatedly broken into substrings at the same symbol for a given level of the recursion. As with Algorithm 4.1, one possible pseudocode implementation of Algorithm 6.1 can be found in Figures 9 and 10. However, unlike Lemma 4.1, which follows directly from Conditions 3.2 and 3.1, it is not immediately obvious that Algorithm 6.1 accepts all unavoidable strings $p$. In order to demonstrate that this more complex procedure also produces no false negatives, we appropriately modify our notion of V-trees by incorporating a “placeholder” node. We call a level of a rooted tree all the nodes which are the same number of edges from the root. The following definition is given in terms of levels, which more easily accommodates the requirement that all the non-placeholder nodes at a particular level correspond to the same symbol.

Definition 6.1. [augmented V-trees] Let $\nu$ be an ordering of $V$. Define the set of augmented V-trees as $\mathcal{L}(V) = \{(L_1, L_2, \ldots, L_{|V|}) \mid L_i = (l_{i, 1}, l_{i, 2}, \ldots, l_{i, c_i}) \text{ with } l_{i, j} \in \{\nu(i), \varepsilon\} \text{ for } 1 \leq j \leq c_i \text{ and } c_i = 2 \cdot |\{j : l_{i-1, j} = \nu(i-1)\}| + |\{j : l_{i-1, j} = \varepsilon\}|. \}$ Let $c_1 = 1$ and require that for every $1 \leq i \leq |V|$ there exists $1 \leq j \leq c_i$ such that $l_{i, j} = \nu(i)$.

As illustrated in Figure 11, we think of $L \in \mathcal{L}(V)$ as being a rooted binary tree with $|V|$ levels, $L_1, \ldots, L_{|V|}$. Each level $i$ is associated with an element $\nu(i)$ of $V$ and is specified by an ordered list of nodes $(l_{i, 1}, l_{i, 2}, \ldots, l_{i, c_i})$. The first level always consists of one node, $\nu(1)$. At each subsequent level $i$, a node may be labeled either $\nu(i)$ or $\varepsilon$ so long as not all $l_{i, j}$ are $\varepsilon$. Matching children to parent nodes from left to right, a node in the $i$th level labeled $\nu(i)$ has two children in the $(i + 1)$st level while an $i$th level $\varepsilon$ node has only one child on the $(i + 1)$st level. Thus, the $\varepsilon$ nodes function as “placeholders” for an eventual $\nu(j)$ node. Because $L_1 = (\nu(1))$, we have that $c_i \leq 2^{i-1}$ for $1 \leq i \leq |V|$. Hence, the maximum number of nodes in an augmented V-tree is $2^{|V|} - 1$. 
As before, we uniquely associated a string $\pi(L)$ to every augmented $V$-tree. We first define an appropriate notion of inserting instances of a particular symbol into a given string.

**Definition 6.2.** For a symbol $z$ not already occurring in a string $p$ and $J \subseteq \{1, 2, \ldots, |p| + 1\}$, define $\lambda(p, z, J)$ to be the string obtained from $p$ by inserting $z$ between the $(j - 1)$st and $j$th positions in $p$ for all $j \in J$.

**Example 6.2.** Let $p = xyzzz$, $w \not\subseteq p$, and $J = \{1, 4, 6\}$. Then $\lambda(p, w, J) = wxyzwzw$.

If $1 \in J$, then an instance of $z$ is added before the first symbol of $p$, while if $|p| + 1 \in J$, one is appended after the last. Note that $p$ exists as a subsequence of $\lambda(p, z, J)$, as does any subsequence of $p$ itself. In the degenerate case when $J = \emptyset$, we have that $\lambda(p, z, J) = p$.

**Definition 6.3.** Let $L \in \mathcal{L}(V)$ with ordering $\nu$. If $|V| = 1$, then $\pi(L) = \nu(1)$. For $|V| = k \geq 2$, define $\pi : \mathcal{L}(V) \rightarrow V^+$ by $\pi(L) = \lambda(\ldots(\lambda(\nu(1), \nu(2), J_2), \nu(3), J_3), \ldots), \nu(k), J_k)$ where $j \in J_i$ if and only if $\nu(i) = \nu(i)$ for $1 \leq j \leq c_i$.

**Example 6.3.** As in Figure 11, consider $V = \{x, y, z\}$, $p = xyzzz$, and $L = ((y), (\varepsilon, z), (x, x, x)), L' = ((z), (y, \varepsilon), (x, x, x)) \in \mathcal{L}(V)$. Then $\pi(L) = ((y, z, \{2\}), x, \{1, 2, 3\}) = p$ and $\pi(L') = ((z, y, \{1\}), x, \{1, 2, 3\}) = p$.

We show inductively that the mapping $\pi$ is well defined for $L \in \mathcal{L}(V)$. Let $p_{i+1} = \lambda(p_i, \nu(i + 1), J_{i+1}) = \lambda(\ldots(\lambda(\nu(1), \nu(2), J_2), \ldots), \nu(i + 1), J_{i+1})$ for the first $1 \leq i < k$ levels of $L$ and $p_1 = \nu(1)$. Since $\nu$ is an ordering of $V$, $\nu(j) \not\subseteq p_i$ for $j > i$. All that remains to show is that $J_{i+1} \subseteq \{1, 2, \ldots, |p_i| + 1\}$. We know that $j \in J_{i+1}$ implies that $1 \leq j \leq c_{i+1}$. Clearly, $|p_1| + 1 = 2 = c_2$ so that $J_2 \subseteq \{1, 2\}$. Assume $|p_{i-1}| + 1 = c_i$. Then, $c_{i+1} = 2|J_i| + (c_i - |J_i|) = c_i + |J_i| = |p_{i-1}| + 1 + |J_i| = |p_i| + 1$.

One advantage of augmented $V$-trees is immediately clear, in contrast to the $V$-tree problem illustrated in Figure 6 on page 16. If $L \in \mathcal{L}(V)$ contains no $\varepsilon$ nodes, we have that $\pi(L)$ is isomorphic to $Z_{|V|}$. Hence, there is no need to impose an additional length restriction on $\pi(L)$ to meet Condition 3.3. Moreover, we will see that strings associated with augmented $V$-trees are precisely those recognized by Algorithm 6.1.

**Lemma 6.1.** $p \in V^+$ with $|p| = |V|$ is accepted by Algorithm 6.1 if and only if $p = \pi(L)$ for some $L \in \mathcal{L}(V)$.
Proof. Tracing out an accepting computation by Algorithm 6.1 yields an associated augmented \( V \)-tree with a node labeled by the corresponding isolated symbol \( x \) when a substring is split and \( \varepsilon \) otherwise. Conversely, suppose that \( p = \pi(L) \) for some \( L \in \mathcal{L}(V) \). We must show that if \( x \neq \nu(i) \) is chosen as the \( i \)th isolated symbol, then the algorithm still completes an accepting computation. Suppose that \( \nu(1), \ldots, \nu(i-1) \) have been chosen as the first \( i \) isolated symbols and consider the substrings \( q_1, \ldots, q_n \) of \( p \) for \( |p| = n \) generated by Algorithm 6.1. We claim that \( \nu(i) \) occurs at most once in each substring \( q_1, \ldots, q_n \). Suppose instead that there are at least two instances of \( \nu(i) \) in \( q_j \) for some \( 1 \leq j \leq n \). By construction, there exists \( 1 \leq j', j'' \leq i-1 \) such that \( \nu(j') \nu(j'') \leq p \) and that \( \nu(1), \ldots, \nu(i-1) \leq q_j \). As in the discussion following Definition 6.3, consider \( p_{i-1} = \lambda(\ldots(\lambda(\nu(1), \nu(2), J_2) \ldots), \nu(i-1), J_{i-1}) \). We have that \( \nu(j') \nu(j'') \leq p_{i-1} \). However, at most one instance of \( \nu(i) \) can be inserted between \( \nu(j') \) and \( \nu(j'') \) to form \( p_i = \lambda(p_{i-1}, \nu(i), J_i) \). Hence, there can be at most one instance of \( \nu(i) \) in \( q_j \). Thus, if the algorithm has further subdivided \( q_1, \ldots, q_n \), it is still the case that \( \nu(i) \) is isolated or nonexistent in each one. Consequently, to complete an accepting computation of \( p, \nu(i) \) may be chosen as an isolated symbol at any point in Algorithm 6.1 after \( \nu(1), \ldots, \nu(i-1) \) have been used.

As with the corresponding result for \( V \)-trees, the proof shows that although the execution of a particular implementation of Algorithm 6.1 is dependent on the choice of an isolated symbol, it does not affect the outcome. In order to show that all unavoidable strings are accepted by Algorithm 6.1, we demonstrate that they can be associated with an augmented \( V \)-tree. Intuitively, an appropriate \( L \) is constructed from a \( \sigma \)-deletion sequence which reduces \( p \) to \( \varepsilon \) as in Figure 12. Working backward through the \( \sigma \)-deletion sequence, at each stage the deleted symbols are assigned locations within levels of the expanding augmented \( V \)-tree. Although it has not been emphasized here, we note that other results [6, 7, 8] show that the number of symbols needing to be \( \sigma \)-deleted from a string \( p \) can range from \( 1 \) to \( |B \setminus A| \), for two-window sets \( A \) and \( B \). Hence, unlike the example given in Figure 12, the following proof must address the issue of \( \sigma \)-deletions where \( |F| > 1 \).

**Theorem 6.1.** Let \( p \in V^+ \) with \( \alpha(p) = V \) be unavoidable. Then there exists \( L \in \mathcal{L}(V) \) such that \( \pi(L) = p \).

**Proof.** We build an appropriate \( L \) inductively according to a given complete \( \sigma \)-deletion sequence of \( p, \sigma_{F_m}(\ldots \sigma_{F_2}(\sigma_{F_1}(p)) \ldots) = \varepsilon \). For \( 1 \leq i \leq m \leq |V| \), let \( p_i = \sigma_{F_i}(\ldots \sigma_{F_2}(\sigma_{F_1}(p)) \ldots) \).
TABLE 2.
The 31 non-isomorphic avoidable \( p \in \{w, x, y, z\}^+ \) with \( \alpha(p) = 4 \) which have nonempty \( T(p) \), unlocked \( G(p) \), \( |p| < 2^4 - 1 \), and only unavoidable proper substrings.

<table>
<thead>
<tr>
<th>p</th>
<th>( \pi(L_1, L_2, L_3, L_4) )</th>
<th>( L_4 = (\varepsilon, y, y, \varepsilon, y, \varepsilon) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x y z w y x y )</td>
<td>( \pi(L_1, L_2, L_3, L_4) )</td>
<td>( L_4 = (\varepsilon, y, y, \varepsilon, y, \varepsilon) )</td>
</tr>
<tr>
<td>( x y z y w y x y )</td>
<td>( \pi(L_1, L_2, L_3, L_4) )</td>
<td>( L_4 = (\varepsilon, y, y, \varepsilon, y, \varepsilon) )</td>
</tr>
<tr>
<td>( x y z x y z y z )</td>
<td>( \pi(L_1, L_2, L_3, L_4) )</td>
<td>( L_4 = (\varepsilon, y, y, \varepsilon, y, \varepsilon) )</td>
</tr>
<tr>
<td>( x y z y w z y x y )</td>
<td>( \pi(L_1, L_2, L_3, L_4) )</td>
<td>( L_4 = (\varepsilon, y, y, \varepsilon, y, \varepsilon) )</td>
</tr>
<tr>
<td>( x y z x w y z y z )</td>
<td>( \pi(L_1, L_2, L_3, L_4) )</td>
<td>( L_4 = (\varepsilon, y, y, \varepsilon, y, \varepsilon) )</td>
</tr>
<tr>
<td>( x y z y w z x y z )</td>
<td>( \pi(L_1, L_2, L_3, L_4) )</td>
<td>( L_4 = (\varepsilon, y, y, \varepsilon, y, \varepsilon) )</td>
</tr>
<tr>
<td>( x y z x w y z y z )</td>
<td>( \pi(L_1, L_2, L_3, L_4) )</td>
<td>( L_4 = (\varepsilon, y, y, \varepsilon, y, \varepsilon) )</td>
</tr>
</tbody>
</table>

\[ F_1 = \{ y \} \]
\[ F_1 = \{ y \} \]
\[ p_1 = \sigma_{F_1} x x x y z = x x x y z = \pi(L_1, L_2, L_3, L_4) \] for \( L_4 = (\varepsilon, y, y, \varepsilon, y, \varepsilon) \)

\[ F_2 = \{ x \} \]
\[ F_2 = \{ x \} \]
\[ p_2 = \sigma_{F_2} x x x y z = x x x y z = \pi(L_1, L_2, L_3, L_4) \] for \( L_4 = (\varepsilon, y, y, \varepsilon, y, \varepsilon) \)

\[ F_3 = \{ w \} \]
\[ F_3 = \{ w \} \]
\[ p_3 = \sigma_{F_3} x x x y z = x x x y z = \pi(L_1, L_2, L_3, L_4) \] for \( L_4 = (\varepsilon, y, y, \varepsilon, y, \varepsilon) \)

\[ F_4 = \{ z \} \]
\[ F_4 = \{ z \} \]
\[ p_4 = \sigma_{F_4} x x x y z = x x x y z = \pi(L_1, L_2, L_3, L_4) \] for \( L_4 = (\varepsilon, y, y, \varepsilon, y, \varepsilon) \)

FIG. 12. The complete \( \sigma \)-deletion sequence of \( p = x y z y w x z x \) as given in Example 2.9 on page 9 can be used to construct \( L \in L(V) \) such that \( \pi(L) = p \).
Suppose $F_i$ has been $\sigma$-deleted from $p_{i-1}$ to form $p_i = \sigma_{F_i}(p_{i-1})$. For every $y \in F_i$, there exists $x, z \in V \cup \{\varepsilon\}$ such that $xyz \in p_{i-1}$ and $xz \in p_i$. Hence, to reconstruct $p_{i-1}$ from $p_i$ by “inserting” the symbols of $F_i$, at most one instance of $y \in F_i$ can be reinserted between any two symbols of $p_i = \sigma_{F_i}(p_{i-1})$ to reform $p_{i-1}$. Furthermore, this places bounds on the size of $F_i$.

In particular, since $p_m = \varepsilon$, we know that $|F_m| = 1$ and that $p_{m-1} = y_1$ for some $y_1 \in V$. Hence, let $\nu(1) = y_1$ and $L_1 = ((\nu(1)))$. We have then that $\pi(L_1) = p_{m-1}$.

Similarly, it must be that $|F_{m-1}| \leq 2$. Suppose that $F_{m-1} = \{y_2\}$. There can be at most two instances of $y_2$ in $p_{m-2}$. Let $l_{2,1} = \nu(2) = y_2$ if $y_2y_1 \leq p_{m-2}$ and $l_{2,2} = \nu(2)$ if $y_1y_2 \leq p_{m-2}$; otherwise let $l_{2,j} = \varepsilon$. Clearly then, $\pi((L_1, L_2)) = \lambda(\nu(1), \nu(2), J_2) = p_{m-2}$. If $F_{m-1} = \{y_2, y_3\}$, then we may assume that $p_{m-2} = y_2y_3y_2$. Let $\nu(2) = y_2$ with $L_2 = (\nu(2), \varepsilon)$, and let $\nu(3) = y_3$ with $L_3 = (\varepsilon, \varepsilon, \nu(3))$. Again, by construction, $\pi((L_1, L_2, L_3)) = \lambda(\nu(1), \nu(2), J_2, \nu(3), J_3) = p_{m-2}$. Note, for $j = 1 + |F_{m-1}|$, we have that $c_{j+1} = 2 \cdot \{|g : l_{j,g} = \nu(j)\}| + \{|g : l_{j,g} = \varepsilon\} \geq p_{m-2} + 1$.

Suppose the last $i$ $\sigma$-deletions in the complete sequence for $p$ have been used to specify the first $j$ levels of $L$, where $j = \sum_{h=0}^{i-1} |F_{m-h}|$, and that

$$\pi((L_1, \ldots, L_j)) = \lambda(\ldots, \lambda(\nu(1), \nu(2), J_2), \ldots, \nu(j), J_j) = p_{m-i} = \sigma_{F_{m-i-1}}(\ldots, \sigma_{F_1}(p)\ldots)$$

Suppose further that $c_{j+1} = |p_{m-i}| + 1$.

Consider $p_{m-i-1}$ where $i < m$. Let $F_{m-i} = \{y_{j+1}, y_{j+2}, \ldots, y_{j+i}\}$. We will construct $L_{j+1}, \ldots, L_{j+i}$ such that

$$\pi((L_1, \ldots, L_j, L_{j+1}, \ldots, L_{j+i})) = p_{m-i-1} = \sigma_{F_{m-i-1}}(\ldots, \sigma_{F_1}(p)\ldots)$$

Suppose the $y_{j+i}$ first appear in increasing index order in $p_{m-i-1}$ and define $\nu(j + h) = y_{j+i}$ for $h = 1, \ldots, e$. Let $d_h$ be the number of $y_{j+i}$ in $p_{m-i-1}$. Note that $\sum_{h=1}^e d_h \leq |p_{m-i}| + 1$ since each instance in $p_{m-i-1}$ of a $y$ from $F_{m-i}$ must be separated from all others.

Beginning with $h = 1$, we construct $L_{j+1}$ as follows. Let $l_{j+1,g} = \nu(j+1)$ if $y_{j+i} \in p_{m-i-1}$ between the $(g-1)$st and $g$th symbols of $p_{m-i}$, and $l_{j+1,g} = \varepsilon$ otherwise, for $1 \leq g \leq |p_{m-i}| + 1 = c_{j+1}$. We repeat this process for subsequent levels $L_{j+2}, \ldots, L_{j+i}$ with one crucial difference. For $1 < h \leq e$, we need to “shift” the second index of $l_{j+i,g}$ according to the number of instances of $y_{j+i}$ present in $L_{j+i}$ for $1 \leq f < h$.

Consider an instance of $\nu(j + h)$ in $p_{m-i-1}$ which occurs between the $(g-1)$st and $g$th symbols of $p_{m-i}$. Let $s(\nu(j + h), g)$ be the number of $\nu(j + f)$ for $1 \leq f < h$ preceding that instance of $\nu(j + h)$ in $p_{m-i-1}$. For $L_{j+i}$ with $1 < h \leq e$, we define $l_{j+i+g,n(\nu(j+h),g)} = \nu(j + h)$ for an instance
of \( \nu(j + h) \) in \( p_{m-i-1} \) between the \((g - 1)\)st and \(g\)th symbols of \( p_{m-i} \), and \( l_{j+h,f} = \varepsilon \) otherwise.

By construction, the descendants of any \( \nu(j + h) \) node in levels \( L_{j+h+1} \) through \( L_{j+e} \) will always be \( \varepsilon \) nodes. Hence, we have that \( c_{j+h+1} = c_{j+h} + d_h \) for \( 1 \leq h \leq e \). Thus, \( c_{j+h+1} = c_{j+1} + \sum_{f=1}^{h} d_f \).

According to the definition of \( L_{j+1} \), \( \lambda \) inserts the \( \nu(j + 1) \) symbols into \( p_{m-i} \) appropriately. Hence, \( \pi(L_1, \ldots, L_j, L_{j+1}) = \lambda(p_{m-i}, y_{j+1}, J_{j+1}) \).

For \( 2 \leq h \leq e \), any instance of \( \nu(j + h) \) which should appear between the \((g - 1)\) and \(g\)th symbols of \( p_{m-i} \) would be correctly inserted between the \((g - 1 + s(\nu(j + h), g))\)th and \((g + s(\nu(j + h), g))\)th symbols of \( \lambda(\ldots \lambda(\nu(1), \nu(2), J_2) \ldots \nu(j + h - 1), J_{j+h-1}) \). Moreover, \( c_{j+e+1} = c_{j+1} + \sum_{h=1}^{e} d_h = |p_{m-i}| + 1 + \sum_{h=1}^{e} d_h = |p_{m-i-1}| + 1 \). Thus, by construction, \( \lambda(\ldots \lambda(\nu(1), \nu(2), J_2) \ldots \nu(j + e), J_{j+e}) = p_{m-i-1} \).

Hence, based on the complete \( \sigma \)-deletion sequence of an unavoidable \( p \in V^{+} \), \( \alpha(p) = V \), we can always construct \( L \in \mathcal{L}(V) \) such that \( \pi(L) = p \).

Consequently, Algorithm 6.1 produces no false negatives.

**Corollary 6.1.** If \( p \) is unavoidable, then \( p \) is accepted by Algorithm 6.1.

As before, there is an exact (although non-unique) correspondence between augmented \( V \)-trees and unavoidable strings when \( |V| = 3 \). When \( |V| = 4 \), there are a total of 865 nonisomorphic \( p = \pi(L) \) for \( L \in \mathcal{L}(V) \).

Recall that 438 of those are known to be unavoidable. Of the remaining 427, there are none having unlocked \( G(p) \), although 65 are of the form \( p = x qxj \) where both \( x q \) and \( qxj \) are unavoidable. For \( |V| \leq 4 \) then, the prospects with augmented \( V \)-trees are markedly better; with only the small additional computation of checking whether \( G(p) \) has more than one connected component, unavoidability may be decided by Algorithm 6.1 when \( \alpha(p) \leq 4 \).

With \( |V| = 5 \), however, the situation has deteriorated rapidly. We know there to be 55,572 nonisomorphic \( p \) with \( \alpha(p) = 5 \). But there are a grand total of 2,384,331 other nonisomorphic \( p = \pi(L) \) for some \( L \in \mathcal{L}(V) \).

Of those, 60,702 have unlocked \( G(p) \), of which 3720 are also of the form \( p = x_1 qxj \) where \( x_1 q \) and \( qxj \) are both unavoidable. A sample of such strings may be found in Table 3. Also, 5311 of the strings with locked \( G(p) \) have only unavoidable proper substrings.

Once again, given any string which satisfies all four of the necessary unavoidability criteria but which is itself avoidable, we can construct infinitely many others. A level \( L_i \) of a \( V \)-tree, and so each node \( \nu(i) \), may be expanded by another augmented \( U \)-tree so long as \((V \setminus \{\nu(i)\}) \cap U = \emptyset \).

This sufficient condition is exactly the restriction imposed on \( \rho(p, z, q) \) to
\section*{Table 3.}
A selection of the 3720 nonisomorphic avoidable $p = \pi(L)$ for $L \in \mathcal{L}(\{u, v, x, y, z\})$ which have unlocked $\mathcal{O}(p)$ and unavoidable proper substrings.

<table>
<thead>
<tr>
<th>$xyzzxyuxyu yzxzyxuz$</th>
<th>$xyzzxyuuxyzzyuxz$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$xyzzxyuxyu xzyvxy$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxxy yvxxxyu x y$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxzxyvxy uy y$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxzyx yzxy x y$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxzxyvxy uy y$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxzy xzvxyuyx y$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxzyuxy yzxyx y$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxzyxy xuyzxy y$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxzyux yzvxy xy$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxxyx zvxxxyuy y$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxzyvx yzvxy xy$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxxy vxxxyuy y$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxzyvxy xzvxy y$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxzyux yzvxy xy$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxzyux yzvxy xy$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
<tr>
<td>$xyzzxyuxzyux yzvxy xy$</td>
<td>$xyzzxyuxyvzxyz$</td>
</tr>
</tbody>
</table>
preserve unavoidability when \( p \) and \( q \) are. Node "expansion" is simply a matter of inserting additional levels and then incorporating appropriately sized blocks of \( \varepsilon \) nodes to broaden subsequent levels.

**Theorem 6.2.** Let \( p \in V^+ \) be avoidable. Suppose further that

1. there exists \( L \in \mathcal{L}(V) \) with \( \pi(L) = p \),
2. \( p \) has unlocked \( G(p) \),
3. and all proper substrings of \( p \) are unavoidable.

Then there exist infinitely many avoidable \( r \) which also satisfy the three conditions above.

The proof is essentially the same as Theorem 5.2, except that the length restriction is automatically satisfied by any \( p = \pi(L) \). As before, the base case \( p \) is not a substring of \( \rho(p, y, q) \) or any subsequent \( r \). Hence, Algorithm 6.1 returns a false positive for the 3720 minimal counterexamples when \( \alpha(p) = 5 \), and arbitrarily many more for \( |V| > 5 \).

### 7. Subsequences and Unavoidability

The question of subsequences, rather than simply substrings, was raised at the end of Section 5. As defined in the previous section, augmented \( V \)-trees have a very natural connection to string subsequences through their description in terms of different levels. We know already that \( p \) is unavoidable if and only if there exists an unavoidable subsequence \( q \), where the symbols of a free set \( F \) have been \( \sigma \)-deleted from \( p \) to form \( q \). We might wonder whether there is a more general relationship between subsequences and unavoidability, without regard to free sets and \( \sigma \)-deletions.

Empirically, we investigate the prospects with strings containing five distinct symbols. We know that there are 438 nonisomorphic unavoidable \( p \) with \( \alpha(p) = 4 \) and \( 4 \leq |p| \leq 15 \). We let \( V = \{u, v, x, y, z\} \) and for each unavoidable \( p \in \{u, v, y, z\} \), we construct \( r = \lambda(p, x, J) \) for all \( J \subseteq \{1, 2, \ldots, |p| + 1\} \). Eliminating redundancies from the resulting strings over \( V \) yields 1,000,589 nonisomorphic avoidable \( r = \lambda(p, x, J) \). By construction, for each avoidable \( r = \lambda(p, x, J) \) there exists \( L \in \mathcal{L}(V) \) such that \( \pi(L) = r \). Furthermore, of the 1,000,589 nonisomorphic avoidable \( r = \lambda(p, x, J) \), 19,864 have \( G(r) \) with at least two connect components and 6389 have the form \( r = x_1 q x_j \) where \( x_1 q \) and \( q x_j \) are unavoidable. Of those strings, 1700 have both unlocked \( G(r) \) and only unavoidable proper substrings. Thus, for \( L = (L_1, L_2, L_3, L_4, L_5) \in \mathcal{L}(V) \) and \( L' = (L_1, L_2, L_3, L_4) \), the unavoidability of \( \pi(L') \) by no means insures the unavoidability of \( \pi(L) \). Furthermore, we have seen that for unavoidable
$p \in \{u, v, y, z\}$ and $|p| \geq 7$ (except $p = uuuyuzu$), there exist both avoidable and unavoidable $r = \lambda(p, x, J)$, although the distribution of those with unlocked $G(r)$ and/or only unavoidable proper substrings is less general.

Furthermore, only 54,766 nonisomorphic unavoidable $\lambda(p, x, J)$ are obtained in this fashion, although there are 55,572 unavoidable strings with five symbols. Those missing include almost all of the unavoidable strings having a unique worthwhile $\sigma$-deletion where $|F| > 1$. There are 799 strings with five distinct symbols having a unique worthwhile $\sigma$-deletion where $|F| = 2$. The eight strings which have nonempty $B \cap A$ for two-window sets $A$ and $B$ are formed by $\lambda(p, x, J)$. However, the other 791 cannot be. The remaining fifteen missing unavoidable strings all have two worthwhile $\sigma$-deletions, both having free sets of size 2. Again, we note that a worthwhile $\sigma$-deletion may require a free set $F \subset B \setminus A$ whose size takes on any value between 1 and $|B \setminus A| \in [6, 7, 8]$.

Clearly, forming strings by inserting a new symbol into unavoidable $p$ with $\alpha(p) = 4$ results both in many additional avoidable strings as well as missing unavoidable ones. If we consider also avoidable $\pi(L)$ for $L \in L(\{u, v, y, z\})$, we see that for every such $L$ there exists some $J$ such that $\lambda(\pi(L), x, J)$ is unavoidable. From this we begin to conclude that subsequences are not, in general, useful in determining unavoidability.

**Theorem 7.1.** Suppose $q$ is a subsequence of a string $p$. Then $p$ and $q$ can be any combination of avoidable/unavoidable.

**Proof.** Observe that $p = \lambda(q, z, J)$ for $J = \{1, 2, \ldots, |q| + 1\}$ is unavoidable if and only if $q$ is, since $F = \{z\}$ is the only possible worthwhile $\sigma$-deletion.

Let $p \in V^+$ be unavoidable and $\alpha(p) = |V| = k \geq 5$. Suppose that $p$ has a unique worthwhile $\sigma$-deletion with $|F| > 1$ and that, for every $x \in V$, $[a, x]$ and $[x, b]$ are in different connected components of $G(p)$. By the results of [6, 7, 8], such strings are known to exist. Now consider $L = (L_1, L_2, \ldots, L_{k-1}, L_k) \in L(V)$ with ordering $\nu$ such that $p = \pi(L)$. Then $q = \pi(L')$ is avoidable for $L' = (L_1, L_2, \ldots, L_{k-1}) \in L(V \setminus \{\nu(k)\})$. By assumption, it must be that there exist $A, B \subseteq V$ satisfying the two-window criteria such that $\nu(k) \in B \setminus A$. Hence, $\nu(k)$ can be $\sigma$-deleted from $p$. However, $p$ has exactly one worthwhile $\sigma$-deletion and $|F| > 1$. Thus, $\pi(L') = \sigma(\nu(k))|p)$ must be avoidable.

We know there exist unavoidable $r$ with $\alpha(r) = 4$ such that, for some $J$, $\lambda(r, x, J)$ is avoidable. Suppose that $q$ is unavoidable with $r \leq q$. Then there exists $J' \subseteq \{1, \ldots, |q| + 1\}$ such that $\lambda(r, x, J) \leq \lambda(q, x, J')$. Thus $p = \lambda(q, x, J')$ is avoidable, although the subsequence $q$ is unavoidable.
The proof remains true even if $p$ and $q$ are required to satisfy generalizations of the four necessary conditions for unavoidability given in Section 3. Hence, beginning with $k = 5$, the avoidability or unavoidability of an arbitrary subsequence $q$ of a string $p$ with $\alpha(p) = k$ says nothing about the unavoidability of $p$. Unless, of course, $q = \sigma_F(p)$ for some free set $F$. Thus, except those associated with $\sigma$-deletions, subsequences in general are not useful in deciding unavoidability.

8. A FURTHER V-TREE RESTRICTION

On a final note, we return to the idea of splitting $p$ as $p = q_1 y q_2$ for an isolated symbol $y$ where we also impose the restriction that if $q_1$ and $q_2$ are both nonempty strings, then there must exist the same isolated symbol in both. Before we had remarked that this would not work for an unavoidable string such as $xyzzx$, whereas we are now in a position to say something more significant. Consider an augmented $V$-tree $L$ where we do not allow any $\varepsilon$ node to be the parent of a non-$\varepsilon$ node. Then $\pi(L)$ satisfies the property outlined above. For $V = \{u, v, y, z\}$, there are 540 such nonisomorphic strings $\pi(L)$ for $L \in \mathcal{L}(V)$, although there are only 438 unavoidable strings with four symbols. Of the 540 such $\pi(L)$, only 268 are unavoidable. Hence, such a restriction to the augmented $V$-trees only increases the number of false negatives, without eradicating the false positives.

9. CONCLUSIONS

We recall the four necessary conditions for the unavoidability of a string $p \in V^+$ with $\alpha(p) = |V| = k$ currently under consideration:

Condition 3.1: if $p$ is unavoidable and $q \preceq p$, then $q$ is unavoidable.

Condition 3.2: $p$ is unavoidable only if $p$ has an isolated symbol.

Condition 3.3: $p$ is unavoidable only if either $|p| < 2^k - 1$ or $p$ is isomorphic to $Z_k$ under a one-to-one mapping $\phi : V \to A_k$.

Condition 3.4: $p$ is unavoidable only if $\mathcal{G}(p)$ is not locked.

In terms of Condition 3.4, we know there exist strings, such as the unavoidable $p = xyzuvvy$ and the avoidable $q = xyzuvzvy$, where $\mathcal{G}(p) = \mathcal{G}(q)$. Hence, we do not expect any property of $\mathcal{G}(p)$ beyond its having at least two connected components to significantly help in determining the unavoidability of $p$. Likewise, as long as $\alpha(p) \leq |p| \leq 2^{\alpha(p)-1}$, the bound on the length of $p$ given in Condition 3.3 is of little use in deciding string unavoidability.

So we return the two conditions which were the original motivation for this work: Condition 3.1 and Condition 3.2. Here we have considered possi-
ble combinations and extensions of the requirements that \( p \) have an isolated symbol and no avoidable substrings. We have seen the existence of many nonisomorphic avoidable \( p = xqy \) where \( x, y \in V, p, q \in V^+ \) and both \( xq \) and \( qy \) are unavoidable. Since we know that all proper substrings of a string \( p \) can be unavoidable, but \( p \) is not, this suggests that unavoidability is not a string property which can be optimized locally. In conjunction, substring unavoidability and the isolated symbol property lead us to consider \( V \)-trees — regular, augmented, and/or restricted being the three distinct possible recursive extensions depending on whether the same symbol was isolated across the substrings of the recursion or not. We have shown that none of them is adequate to characterize an unavoidable string, and thus demonstrated the insufficiency of any combination of extensions of these four necessary conditions.

Finally, the argument in Section 7 regarding the (un)avoidability of subsequences can be interpreted as implying that there is no local \( V \)-tree property which would determine string unavoidability either. Globally, however, a \( V \)-tree is just a different way of representing particular types of strings in \( V^+ \). Thus, any global \( V \)-tree property which might decide unavoidability (although we do not believe there to be any) would represent a new necessary and sufficient condition on unavoidability not resulting from the four considered here.

REFERENCES


