Forced Repetitions over Alphabet Lists

Neerja Mhaskar[1] and Michael Soltys[2]

[1] McMaster University
Dept. of Computing & Software
1280 Main Street West
Hamilton, Ontario L8S 4K1, Canada
pophlin@mcmaster.ca
[2] California State University Channel Islands
Dept. of Computer Science
One University Drive
Camarillo, CA 93012, USA
michael.soltys@csuci.edu

Abstract. Thue [14] showed that there exist arbitrarily long square-free strings over an alphabet of three symbols (not true for two symbols). An open problem was posed in [7], which is a generalization of Thue's original result: given an alphabet list $L = L_1, \ldots, L_n$, where $|L_i| = 3$, is it always possible to find a square-free string, $\boldsymbol{w} = \boldsymbol{w}_1\boldsymbol{w}_2 \cdots \boldsymbol{w}_n$, where $\boldsymbol{w}_i \in L_i$? In this paper we show that squares can be forced on square-free strings over alphabet lists iff a suffix of the square-free string conforms to a pattern which we term as an offending suffix. We also prove properties of offending suffixes. However, the problem remains tantalizingly open.

Keywords: strings, square-free, repetition, Thue morphisms

1 Introduction

The study of repetitions in words is an attractive field for both theoretical and applied research. It dates back to the early 20th century and the seminal work of Axel Thue [14,15], who proved the existence of square-free strings over an alphabet of three letters, using iterated morphism. Since his work was not known for a long time, this result was rediscovered by many others independently. For example, the following authors each gave different morphisms to show the existence of a square-free string over a ternary alphabet: [1,10,6,8].

Many different morphisms have been proposed besides Thue's original one. But all these morphisms construct a string over a fixed finite alphabet. A natural generalization of the problem is to allow a (possibly) infinite alphabet of symbols, but to restrict the i-th symbol of the word to come from a particular subset. Thus, we are interested in constructing an arbitrarily long square-free string with constraints imposed on the positions. This generalization has been studied by [7,12,9], among others.

The authors of [7] showed that as long as each position is required to be filled with a symbol from a subset of size at least four, then we can always construct a square-free string over such a list (the i-th symbol of this string comes from the i-th alphabet in the list, and each alphabet is of size at least four). However, the question whether the same holds if the alphabets are restricted to be of size three is still an open question. Note that Thue's original result applies to only a particular case of the problem where all the alphabets in the list are of size three, *and* contain the same elements, but for the generalized case the problem remains open. Also note that if all

Neerja Mhaskar, Michael Soltys: Forced Repetitions over Alphabet Lists, pp. 125–134.
Proceedings of PSC 2016, Jan Holub and Jan Žďárek (Eds.), ISBN 978-80-01-05996-8

subsets are $\{a, b\}$, we cannot construct an arbitrarily long square-free string. In this paper, we show that squares can be forced on certain type of square-free strings over an alphabet list and we give a characterization for such strings.

The outline of the paper is as follows: in Section 2, we give a brief introduction to the terminology. In Section 3, we define a pattern "Offending Suffix," and show that having strings over alphabet lists with a suffix conforming to this pattern is a necessary and sufficient condition to force squares: Theorem 1. In section 4, we give a characterization of square-free strings using borders. In Section 5, we show using rudimentary Kolmogorov complexity that given any alphabet list, there always exists a string $\boldsymbol{w}$ without squares longer than $\frac{1}{5}|\boldsymbol{w}|$. In other words, we are able to eliminate a few huge squares in every case.

2 Background

An *alphabet* is a set of symbols, and Σ is usually used to represent a finite alphabet. The elements of an alphabet are referred to as *symbols* (or *letters*). In this paper, we assume $|\Sigma| \neq 0$. A *string* (or a *word*) over Σ, is an ordered sequence of symbols from it. Formally, $\boldsymbol{w} = \boldsymbol{w}_1\boldsymbol{w}_2 \cdots \boldsymbol{w}_n$, where for each i, $\boldsymbol{w}_i \in \Sigma$, is a string. In order to emphasize the array structure of $\boldsymbol{w}$, we sometimes represent it as $\boldsymbol{w}[1..n]$. The *length* of a string $\boldsymbol{w}$ is denoted by $|\boldsymbol{w}|$. The set of all finite length strings over Σ is denoted by Σ^*. The empty string is denoted by ε, and it is the string of length zero. The set of all finite strings over Σ not containing ε is denoted by Σ^+. We denote Σ_k to be a fixed generic alphabet of k symbols, and $\Sigma^{\geq j}$ to be the set of all strings over Σ of size at least j.

A string $\boldsymbol{v}$ is a *subword* (also known as a *substring* or a *factor*) of $\boldsymbol{w}$, if $\boldsymbol{v} = \boldsymbol{w}_i\boldsymbol{w}_{i+1} \cdots \boldsymbol{w}_j$, where $i \leq j$. If $i = 1$, then $\boldsymbol{v}$ is a *prefix* of $\boldsymbol{w}$ and if $j < n$, $\boldsymbol{v}$ is a *proper prefix* of $\boldsymbol{w}$. If $j = n$, then $\boldsymbol{v}$ is a *suffix* of $\boldsymbol{w}$ and if $i > 1$, then $\boldsymbol{v}$ is a *proper suffix* of $\boldsymbol{w}$. We can express that $\boldsymbol{v}$ is a subword more succinctly using array representation as $\boldsymbol{v} = \boldsymbol{w}[i..j]$. A word $\boldsymbol{v}$ is a *subsequence* of a string $\boldsymbol{w}$ if the symbols of $\boldsymbol{v}$ appear in the same order in $\boldsymbol{w}$. Note that the symbols of $\boldsymbol{v}$ do not necessarily appear contiguously in $\boldsymbol{w}$. Hence, any subword is a subsequence, but the reverse is not true.

A string $\boldsymbol{w}$ is said to have a *repetition* if there exists a subword of $\boldsymbol{w}$ consisting of consecutive repeating factors. The most basic type of repetition is a square and we define it as follows: a string $\boldsymbol{w}$ is said to have a *square* if there exists a string $\boldsymbol{v}$ such that $\boldsymbol{vv}$ is a subword of $\boldsymbol{w}$ and it is *square-free* if no such subword exists. A map $h : \Sigma^* \to \Delta^*$, where Σ and Δ are finite alphabets, is called a *morphism* if for all $x, y \in \Sigma^*$, $h(\boldsymbol{xy}) = h(\boldsymbol{x})h(\boldsymbol{y})$. A morphism is said to be *non-erasing* if for all $\boldsymbol{w} \in \Sigma^*$, $h(\boldsymbol{w}) \geq \boldsymbol{w}$. It is called square-free if $h(\boldsymbol{w})$ is square-free for every square-free word $\boldsymbol{w}$ over Σ.

An *alphabet list* is an ordered list of finite subsets (alphabets), and in our case all the alphabets have the same cardinality. However for the general case we do not need to impose this condition on alphabet lists. Let $L = L_1, L_2, \ldots, L_n$, be an ordered list of alphabets. A string $\boldsymbol{w}$ is said to be a word over the list L, if $\boldsymbol{w} = \boldsymbol{w}_1\boldsymbol{w}_2 \cdots \boldsymbol{w}_n$ where for all i, $\boldsymbol{w}_i \in L_i$. Note that there are no conditions imposed on the alphabets L_i's: they may be equal, disjoint, or have elements in common. The only condition on $\boldsymbol{w}$ is that the i-th symbol of $\boldsymbol{w}$ must be selected from the i-th alphabet of L, i.e., $\boldsymbol{w}_i \in L_i$. The alphabet set for the list $L = L_1, L_2, \ldots, L_n$ is denoted by $\Sigma_L = L_1 \cup L_2 \cup \cdots \cup L_n$. Given a list L of finite alphabets, we can define the set of strings $\boldsymbol{w}$ over L with a regular expression as follows: $R_L := L_1 \cdot L_2 \cdots L_n$. Let $L^+ := L(R_L)$ be the language of all the strings over the list L. For example, if $L_0 = \{\{a,b,c\},\{c,d,e\},\{a,1,2\}\}$, then

$$R_{L_0} := \{a,b,c\} \cdot \{c,d,e\} \cdot \{a,1,2\},$$

and $ac1 \in L_0^+$, but $2ca \notin L_0^+$. Also, in this case $|L_0^+| = 3^3 = 27$.

Given a square-free string $\boldsymbol{w}$ over a list $L = L_1, L_2, \dots, L_n$, we say that the alphabet L_{n+1} *forces a square on* $\boldsymbol{w}$ if for all $a \in L_{n+1}$, $\boldsymbol{w}a$ has a square. Note that, this is not to be confused with forcing a square in $\boldsymbol{w}$. For example, if $L = \{a,b,c\}^7$, and $\boldsymbol{w} = abacaba$, then the alphabet $\{a,b,c\}$ forces a square on $\boldsymbol{w}$, as the strings $\boldsymbol{w}a$, $\boldsymbol{w}b$ and $\boldsymbol{w}c$ all have squares.

We introduce the concept of admissibility of lists. We say that an alphabet list L is *admissible* if L^+ contains a square-free string. For example, the alphabet list $L = \{\{a,b,c\},\{1,2,3\},\{a,c,2\},\{b,3,c\}\}$, is admissible as the string '$a1c3$' over L is square-free.

Let $\mathcal{L}$ represent a *class* of lists; the intention is for $\mathcal{L}$ to denote lists with a given property. For example, we are going to use $\mathcal{L}_{\Sigma_k}$ to denote the class of lists $L = L_1, L_2, \dots L_n$, where for each i, $L_i = \Sigma_k$, and $\mathcal{L}_k$ will denote the class of all lists $L = L_1, L_2, \dots, L_n$, where for each i, $|L_i| = k$, that is, those lists consisting of alphabets of size k. Note that $\mathcal{L}_{\Sigma_k} \subseteq \mathcal{L}_k$. We say that a class of lists $\mathcal{L}$ is admissible if *every* list $L \in \mathcal{L}$ is admissible. An example of admissible class of lists is the class $\mathcal{L}_{\Sigma_3}$ (Thue's result), and $\mathcal{L}_3$ is a class of lists whose admissibility status is unknown, and the subject of investigation in this paper.

A *border* β of a string $\boldsymbol{w}$, is a subword that is both a proper prefix and proper suffix of $\boldsymbol{w}$. Note that the proper prefix and proper suffix may overlap. A string can have many borders. The empty string ε is a border of every string. For example, the string $\boldsymbol{w} = 121324121$ has three borders 1, 121 and the empty string ε. See [13] for many properties of borders.

Given an alphabet Σ, let $\Delta = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \dots, a_1, a_2, a_3, \dots\}$ be variables, where the $\boldsymbol{X}_i$'s range over Σ^*, and the a_i's range over Σ. A *pattern* is a non empty string over Δ^*; for example, $\mathcal{P} = \boldsymbol{X}_1 a_1 \boldsymbol{X}_1$ is a pattern representing all strings where the first half equals the second half, and the two halves are separated by a single symbol. Intuitively, patterns are "templates" for strings. Note that some authors define patterns as being words over variables with no restriction on the size of the variables (see [4]), but we find the definition given here as more amenable to our purpose.

We say that a word $\boldsymbol{w}$ over some alphabet Σ *conforms* to a pattern $\mathcal{P}$ if there is a morphism $h : \Delta^* \longrightarrow \Sigma^*$, such that $h(\mathcal{P}) = \boldsymbol{w}$.

We say that a pattern is *avoidable*, if strings of arbitrary length exist, such that no subword of the string conforms to the pattern, otherwise it is said to be *unavoidable*. For example, the pattern $\boldsymbol{XX}$ is unavoidable for all strings in $\Sigma_2^{\geq 4}$, but there exist strings in Σ_3 of arbitrary length for which it is avoidable (Thue's result, [14]).

The idea of unavoidable patterns was developed independently in [2] and [16]. *Zimin words* (also known as *sesquipowers*) constitute a certain class of unavoidable patterns. The n-th Zimin word, $\mathcal{Z}_n$, is defined recursively over the alphabet Δ of variables of type string as follows:

$$\begin{aligned} \mathcal{Z}_1 &= \boldsymbol{X}_1, \text{and for } n > 1, \\ \mathcal{Z}_n &= \mathcal{Z}_{n-1} \boldsymbol{X}_n \mathcal{Z}_{n-1}. \end{aligned} \tag{1}$$

[16] showed that Zimin words are unavoidable for large classes of words. More precisely, for every n, there exists an N, so that for every word $\boldsymbol{w} \in \Sigma_n^{\geq N}$ there

exists a morphism h so that $h(\mathcal{Z}_n)$ is a subword of $\boldsymbol{w}$. For instance, the pattern $\mathcal{Z}_3 = \boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_1\boldsymbol{X}_3\boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_1$ is unavoidable over Σ_3 for words of length at least 29, as can be checked with an exhaustive search. See [5] for bounds on Zimin word avoidance. For details on Zimin patterns, see [16,3,11,4,5].

3 Offending Suffix Pattern

In this section, we introduce a pattern that we call an "offending suffix", and we show in Theorem 1 that such suffixes characterize in a meaningful way strings over alphabet lists with squares. Let $\mathcal{C}(n)$, an *offending suffix*, be a pattern defined recursively:

$$\begin{aligned} &\mathcal{C}(1) = \boldsymbol{X}_1 a_1 \boldsymbol{X}_1, \text{ and for } n > 1, \\ &\mathcal{C}(n) = \boldsymbol{X}_n \mathcal{C}(n-1) a_n \boldsymbol{X}_n \mathcal{C}(n-1). \end{aligned} \tag{2}$$

To be more precise, given a morphism, $h : \Delta^* \to \Sigma^*$, we call $h(\{a_1, a_2, \ldots, a_n\}) \subseteq \Sigma$ the pivots of h. When all the variables in the set $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n\}$ map to ε, we get the pattern for the shortest possible offending suffix for a list $L \in \mathcal{L}_n$. We call this pattern the *shortest offending suffix*, and employ the notation:

$$\mathcal{C}_s(n) = a_1 a_2 a_1 \cdots a_n \cdots a_1 a_2 a_1. \tag{3}$$

Note that $|\mathcal{C}_s(n)| = 2|\mathcal{C}_s(n-1)| + 1$, where $|\mathcal{C}_s(1)| = 1$, and so, $|\mathcal{C}_s(n)| = 2^n - 1$.

As we are interested in offending suffixes for $\mathcal{L}_3$, we consider mainly:

$$\begin{aligned} &\mathcal{C}(3) = \boldsymbol{X}_3\boldsymbol{X}_2\boldsymbol{X}_1 a_1 \boldsymbol{X}_1 a_2 \boldsymbol{X}_2\boldsymbol{X}_1 a_1 \boldsymbol{X}_1 a_3 \boldsymbol{X}_3\boldsymbol{X}_2\boldsymbol{X}_1 a_1 \boldsymbol{X}_1 a_2 \boldsymbol{X}_2\boldsymbol{X}_1 a_1 \boldsymbol{X}_1, \\ &\mathcal{C}_s(3) = a_1 a_2 a_1 a_3 a_1 a_2 a_1, \end{aligned} \tag{4}$$

and observe that $\mathcal{C}_s(3)a_i$, for $i = 1, 2, 3$, all map to strings with squares.

Pattern $\mathcal{C}$ in (2) bears great resemblance to Zimin words (1) discussed at the end of the previous section. Comparing (1) to (2), one can see that mapping $\boldsymbol{X}_i$ to a_i in (1) yields the same string as mapping $\boldsymbol{X}_i$ to ε in (2). In particular, the shortest offending suffix $\mathcal{C}_s(n)$ can be obtained from the Zimin word $\mathcal{Z}_n$ by mapping X_i's to a_i's. Despite the similarities, we prefer to introduce this new pattern, as the advantage of $\mathcal{C}(n)$ is that it allows for the succinct expression of the most general offending suffix possible.

Given a list L, let $h : \Delta^* \to \Sigma_L^*$, be a morphism. We say that h *respects* a list $L = L_1, L_2, \ldots, L_n$, if h yields a string over L. So, for example, an h that maps each $\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3$ to ε, and also maps $a_1 \mapsto a, a_2 \mapsto b, a_3 \mapsto c$, yields $h(\mathcal{C}(3)) = abacaba$. Such an h respects, for example, a list $L = \{a, e\}, \{a, b\}, \{a, d\}, \{c\}, \{a, e\}, \{b, c, d\}, \{a\}$. In general, papers in the field of string algorithms mix variables over symbols with the symbols themselves, that is, a may stand for both the symbol $a \in \Sigma$, and a variable that takes on values in Σ. In our case, we need to specify exactly what is a variable and what is a symbol.

The main result of the paper, a characterization of squares in strings over lists in terms of offending suffixes, follows.

Theorem 1. *Suppose that $\boldsymbol{w} = \boldsymbol{w}_1\boldsymbol{w}_2\cdots\boldsymbol{w}_{i-1}$ is a square-free string over a list $L = L_1, L_2, \ldots, L_{i-1}$, where $L \in \mathcal{L}_3$. Then, the pivots $L_i = \{a, b, c\}$ force a square on $\boldsymbol{w}$ iff $\boldsymbol{w}$ has a suffix conforming to the offending suffix $\mathcal{C}(3)$.*

Proof. The proof is by contradiction. We assume throughout that our lists are from the class $\mathcal{L}_3$.

($\Leftarrow$) Suppose $\boldsymbol{w} = \boldsymbol{w}_1\boldsymbol{w}_2 \cdots \boldsymbol{w}_{i-1}$ has a suffix conforming to the offending suffix $\mathcal{C}(3)$, where a, b, c are the pivots. Clearly, if we let $L_i = \{a, b, c\}$, then each $\boldsymbol{w}a$, $\boldsymbol{w}b$, $\boldsymbol{w}c$ has a square, and hence by definition L_i forces a square on $\boldsymbol{w}$.

($\Rightarrow$) Suppose, on the other hand, that $L_i = \{a, b, c\}$ forces a square on the word $\boldsymbol{w}$ over $L = L_1, L_2, \ldots, L_{i-1}$. We need to show that $\boldsymbol{w}$ must have a suffix that conforms to the pattern $\mathcal{C}(3)$, with the symbols a, b, c as the pivots. Since L_i forces a square, we know that $\boldsymbol{w}a$, $\boldsymbol{w}b$, $\boldsymbol{w}c$ has a square for a suffix (as $\boldsymbol{w}$ itself was square-free). Let $\boldsymbol{t}a\boldsymbol{t}a$, $\boldsymbol{u}b\boldsymbol{u}b$, $\boldsymbol{v}c\boldsymbol{v}c$ be the squares created by appending a, b and c to $\boldsymbol{w}$, respectively. Here $\boldsymbol{t}$, $\boldsymbol{u}$, $\boldsymbol{v}$ are treated as subwords of $\boldsymbol{w}$.

As all three squares $\boldsymbol{t}a\boldsymbol{t}a$, $\boldsymbol{u}b\boldsymbol{u}b$, $\boldsymbol{v}c\boldsymbol{v}c$ are suffixes of the string $\boldsymbol{w}$, it follows that $\boldsymbol{t}$, $\boldsymbol{u}$, $\boldsymbol{v}$ must be of different sizes, and so we can order them without loss of generality as follows: $|\boldsymbol{t}a\boldsymbol{t}| < |\boldsymbol{u}b\boldsymbol{u}| < |\boldsymbol{v}c\boldsymbol{v}|$. It also follows from the fact that all three are suffixes of $\boldsymbol{w}$, the squares from left-to-right are suffixes of each other. Hence, while $\boldsymbol{t}$ may be empty, we know that $\boldsymbol{u}$ and $\boldsymbol{v}$ are not. We now consider different cases of the overlap of $\boldsymbol{t}a\boldsymbol{t}$, $\boldsymbol{u}b\boldsymbol{u}$, $\boldsymbol{v}c\boldsymbol{v}$, showing in each case that the resulting string has a suffix conforming to the pattern $\mathcal{C}(3)$. Note that it is enough to consider the interplay of $\boldsymbol{u}b\boldsymbol{u}$, $\boldsymbol{v}c\boldsymbol{v}$, as then the interplay of $\boldsymbol{t}a\boldsymbol{t}$, $\boldsymbol{u}b\boldsymbol{u}$ is symmetric and follows by analogy. Also keep in mind that the assumption is that $\boldsymbol{w}$ is square-free; this eliminates some of the possibilities as can be seen below.

1. $\boldsymbol{v} = \boldsymbol{p}\boldsymbol{u}b\boldsymbol{u}$ as shown in Figure 1, where $\boldsymbol{p}$ is a proper non-empty prefix of $\boldsymbol{v}$. Since $\boldsymbol{w}$ is square-free, we assume that $\boldsymbol{p}\boldsymbol{u}b\boldsymbol{u}$ has no square, and therefore $\boldsymbol{p} \neq \boldsymbol{u}$ and $\boldsymbol{p} \neq b$. From this, we get $\boldsymbol{v}c\boldsymbol{v} = \boldsymbol{p}\boldsymbol{u}b\boldsymbol{u}c\boldsymbol{p}\boldsymbol{u}b\boldsymbol{u}$. Therefore, this case is possible.

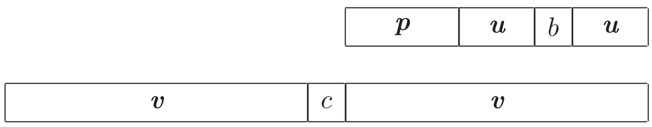

Figure 1. $\boldsymbol{v} = \boldsymbol{p}\boldsymbol{u}b\boldsymbol{u}$

2. $\boldsymbol{v} = \boldsymbol{u}b\boldsymbol{u}$ as shown in Figure 2. Then, $\boldsymbol{v}c\boldsymbol{v} = \boldsymbol{u}b\boldsymbol{u}c\boldsymbol{u}b\boldsymbol{u}$. This case is also possible.

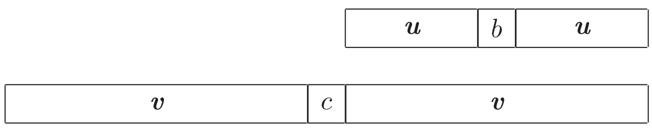

Figure 2. $\boldsymbol{v} = \boldsymbol{u}b\boldsymbol{u}$

3. $c\boldsymbol{v} = \boldsymbol{u}b\boldsymbol{u}$ as shown in Figure 3, then $\boldsymbol{u}_1 = c$. Let $\boldsymbol{u} = c\boldsymbol{s}$, where $\boldsymbol{s}$ is a proper non-empty suffix of $\boldsymbol{u}$, then $\boldsymbol{v}c\boldsymbol{v} = c\boldsymbol{s}bc\boldsymbol{s}cc\boldsymbol{s}bc\boldsymbol{s}$. The subword '$cc$' indicates a square in $\boldsymbol{w}$. This is a contradiction and therefore this case is not possible.

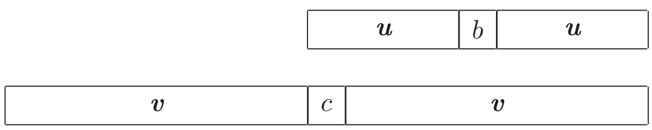

Figure 3. $c\boldsymbol{v} = \boldsymbol{u}b\boldsymbol{u}$

4. $\boldsymbol{vcv} = \boldsymbol{qubu}$ and $|c\boldsymbol{v}| < |\boldsymbol{u}b\boldsymbol{u}|$ as shown in Figure 4, where $\boldsymbol{q}$ is a proper prefix of $\boldsymbol{v}c\boldsymbol{v}$. Let $\boldsymbol{u} = \boldsymbol{p}c\boldsymbol{s}$, where $\boldsymbol{p}$, $\boldsymbol{s}$ are proper prefix and suffix of $\boldsymbol{u}$. Therefore $\boldsymbol{v} = \boldsymbol{s}b\boldsymbol{p}c\boldsymbol{s}$. Since $\boldsymbol{p}$ is also a proper suffix of $\boldsymbol{v}$, one of the following must be true:

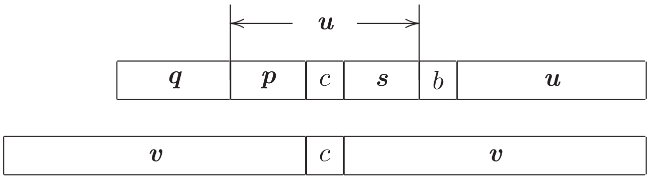

Figure 4. $\boldsymbol{v}c\boldsymbol{v} = \boldsymbol{ngbu}$ and $|c\boldsymbol{v}| < |\boldsymbol{u}b\boldsymbol{u}|$

(a) $|\boldsymbol{s}| = |\boldsymbol{p}|$ and so $\boldsymbol{s} = \boldsymbol{p}$. Since $\boldsymbol{s} = \boldsymbol{p}$, $\boldsymbol{v} = \boldsymbol{s}b\boldsymbol{s}c\boldsymbol{s}$ and $\boldsymbol{v}c\boldsymbol{v} = \boldsymbol{s}b\boldsymbol{s}c\boldsymbol{s}c\boldsymbol{s}b\boldsymbol{s}c\boldsymbol{s}$. The subword '$\boldsymbol{s}c\boldsymbol{s}c$', indicates a square in $\boldsymbol{w}$. This is a contradiction and therefore this case is not possible.
(b) $|\boldsymbol{s}| > |\boldsymbol{p}|$ and so $\boldsymbol{s} = \boldsymbol{rp}$, where $\boldsymbol{r}$ is a proper non-empty prefix of $\boldsymbol{s}$. Substituting $\boldsymbol{rp}$ for $\boldsymbol{s}$, we have $\boldsymbol{v} = \boldsymbol{s}b\boldsymbol{p}c\boldsymbol{s} = \boldsymbol{rp}b\boldsymbol{p}c\boldsymbol{rp}$ and $\boldsymbol{v}c\boldsymbol{v} = \boldsymbol{rp}b\boldsymbol{p}c\boldsymbol{rp}c\boldsymbol{rp}b\boldsymbol{p}c\boldsymbol{rp}$. The subword '$c\boldsymbol{rp}c\boldsymbol{rp}$' indicates a square in $\boldsymbol{w}$. This is a contradiction and therefore this case is not possible.
(c) $|\boldsymbol{s}| < |\boldsymbol{p}|$ and so $\boldsymbol{p} = \boldsymbol{rs}$, where $\boldsymbol{r}$ is a proper non-empty prefix of $\boldsymbol{p}$. Substituting $\boldsymbol{rs}$ for $\boldsymbol{p}$, we have $\boldsymbol{v} = \boldsymbol{s}b\boldsymbol{p}c\boldsymbol{s} = \boldsymbol{s}b\boldsymbol{rs}c\boldsymbol{s}$ and $\boldsymbol{v}c\boldsymbol{v} = \boldsymbol{s}b\boldsymbol{rs}c\boldsymbol{s}c\boldsymbol{s}b\boldsymbol{rs}c\boldsymbol{s}$. The subword '$\boldsymbol{s}c\boldsymbol{s}c$' indicates a square in $\boldsymbol{w}$. This is a contradiction and therefore this case is not possible.

From the above analysis, we can conclude that for L_i to force a square on a square-free string $\boldsymbol{w}$, it must be the case that $\boldsymbol{v} = \boldsymbol{zu}b\boldsymbol{u}$, where $\boldsymbol{z}$ is a prefix (possibly empty) of $\boldsymbol{v}$ and $\boldsymbol{z} \neq \boldsymbol{u}$ and $\boldsymbol{z} \neq b$.

Similarly, we get $\boldsymbol{u} = \boldsymbol{yt}a\boldsymbol{t}$, where $\boldsymbol{y}$ is a prefix (possibly empty) of $\boldsymbol{u}$ and $\boldsymbol{y} \neq \boldsymbol{t}$ and $\boldsymbol{z} \neq \boldsymbol{y}$. Substituting values of $\boldsymbol{u}$ in $\boldsymbol{v}$, we get $\boldsymbol{v} = \boldsymbol{zyt}a\boldsymbol{t}b\boldsymbol{yt}a\boldsymbol{t}$ and $\boldsymbol{v}c\boldsymbol{v} = \boldsymbol{zyt}a\boldsymbol{t}b\boldsymbol{yt}a\boldsymbol{t}c\boldsymbol{zyt}a\boldsymbol{t}b\boldsymbol{yt}a\boldsymbol{t}$. But $\boldsymbol{v}c\boldsymbol{v}$, is a suffix of the square-free string $\boldsymbol{w}$, and it conforms to the offending suffix $\mathcal{C}(3)$ where the elements a, b, c are the pivots.

Therefore, we have shown that if an alphabet L_i forces a square in a square-free string $\boldsymbol{w}$, then $\boldsymbol{w}$ has a suffix conforming to the offending suffix $\mathcal{C}(3)$. $\square$

The following Corollary exploits the fact that an alphabet L_{i+1} *forces* a square on a square-free string $\boldsymbol{v}$ of length i iff $\boldsymbol{v}$ has an offending suffix. But, the size of an offending suffix grows exponentially in the size of the alphabets in the list.

Corollary 2. *If L is a list in $\mathcal{L}_n$ of length at most $2^n - 1$, then L is admissible.*

Proof. Suppose that $L \in \mathcal{L}_n$ and $|L| = 2^n - 1$. We show how to construct a square-free $\boldsymbol{w}$ over L. Let $\boldsymbol{w}_1$ be any one of the three symbols in L_1. Now, inductively for $i \in [2^n - 2]$, assume that $\boldsymbol{v} = \boldsymbol{w}_1\boldsymbol{w}_2 \cdots \boldsymbol{w}_i$ is square-free. If L_{i+1} forces a square on $\boldsymbol{v}$, then by Theorem 1, $\boldsymbol{v}$ must have an offending suffix. But as the shortest possible offending suffix for $|L_{i+1}| = n$ is $\mathcal{C}_s(n)$ of length $2^n - 1$ (see (3)), we get a contradiction since $|\boldsymbol{v}| \leq 2^n - 2$. Thus L_{i+1}, for $i \in [2^n - 2]$, cannot force a square, which means that we can select at least one symbol $\sigma \in L_{i+1}$ so that $\boldsymbol{v}\sigma$ is square free. We proceed this way until $i = 2^n - 2$ and output a square-free string $\boldsymbol{w}$ of length $2^n - 1$ over L. Hence L is admissible. $\square$

From Theorem 1, we know that an alphabet in a list $L \in \mathcal{L}_3$ can force a square on a square-free string $\boldsymbol{w}$ iff $\boldsymbol{w}$ has a suffix $\boldsymbol{s}$ conforming to the offending suffix $\mathcal{C}(3)$. The question is whether $\boldsymbol{s}$ is unique, that is, does the square-free string $\boldsymbol{w}$ contain more that one suffix that conforms to the offending suffix pattern? In Lemma 3, we show that any square-free string $\boldsymbol{w}$ over $L \in \mathcal{L}_3$ has only one suffix $\boldsymbol{s}$ conforming to the offending suffix (w.r.t fixed pivots) if any, that is $\boldsymbol{s}$ is unique.

Lemma 3. *Suppose $\boldsymbol{w}$ is a square-free string over $L = L_1, L_2, \ldots, L_{n-1}$, and $L \in \mathcal{L}_3$. If $\boldsymbol{w}$ has suffixes $\boldsymbol{s}, \boldsymbol{s}'$ conforming to $\mathcal{C}(3)$ with pivots L_n (where $|L_n| = 3$), then $\boldsymbol{s} = \boldsymbol{s}'$.*

Proof. The proof is by contradiction. Suppose that the square-free string $\boldsymbol{w}$ over $L \in \mathcal{L}_3$ has two distinct suffixes $\boldsymbol{s}$ and $\boldsymbol{s}'$ conforming to the offending suffix $\mathcal{C}(3)$ with pivots $L_n = \{a, b, c\}$. That is $\exists h, h(\mathcal{C}(3)) = \boldsymbol{s}$ and $\exists h', h'(\mathcal{C}(3)) = \boldsymbol{s}'$, and $\boldsymbol{s} \neq \boldsymbol{s}'$, and both have pivots in $\{a, b, c\}$. Without loss of generality, we assume that $|\boldsymbol{s}| < |\boldsymbol{s}'|$, and since they are suffixes of $\boldsymbol{w}$, $\boldsymbol{s}$ is a suffix of $\boldsymbol{s}'$. We now examine all possible cases of overlap. Note that $\boldsymbol{s}' = h'(\mathcal{C}(3)) = h'(\boldsymbol{X}_2\mathcal{C}(2)a_3\boldsymbol{X}_2\mathcal{C}(2))$ for some morphism h'. To examine the cases of overlap, let $\boldsymbol{v} = h'(\boldsymbol{X}_2\mathcal{C}(2))$, then $\boldsymbol{s}' = \boldsymbol{v}h'(a_3)\boldsymbol{v}$, where $h'(a_3)$ represents the middle symbol of $\boldsymbol{s}'$. Similarly, the middle symbol of $\boldsymbol{s}$ is represented by $h(a_3)$ for some morphism h. We intentionally use $h'(a_3)$ in $\boldsymbol{s}'$ (and $h(a_3)$ in $\boldsymbol{s}$) as we want to cover all the six different ways in which the variables a_1, a_2, a_3 are mapped to pivots a, b, c.

1. If $|\boldsymbol{s}| \leq \lfloor |\boldsymbol{s}'|/2 \rfloor$, then $\boldsymbol{v} = \boldsymbol{ps}$ (see Figure 5), where $\boldsymbol{p}$ is a prefix of $\boldsymbol{v}$, and $\boldsymbol{ps}h'(a_3)$ is a prefix of $\boldsymbol{s}'$. Observe that, when $|\boldsymbol{s}| = \lfloor |\boldsymbol{s}'|/2 \rfloor$, $\boldsymbol{p} = \varepsilon$. Since $\boldsymbol{s}$ is an offending suffix, we know that $\boldsymbol{s}h'(a_3)$ has a square and hence $\boldsymbol{s}'$ has a square and it follows that $\boldsymbol{w}$ has a square — contradiction.

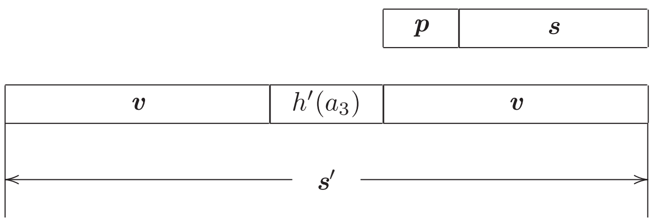

Figure 5. $\boldsymbol{v} = \boldsymbol{ps}$

2. If $|\boldsymbol{s}| = \lfloor |\boldsymbol{s}'|/2 \rfloor + 1$, then $\boldsymbol{s} = h'(a_3)\boldsymbol{u}h(a_3)h'(a_3)\boldsymbol{u}$ (see Figure 6), where $\boldsymbol{u}$ is a non-empty subword of $\boldsymbol{s}$, and $\boldsymbol{v} = \boldsymbol{u}h(a_3)h'(a_3)\boldsymbol{u}$. If the morphisms h and h' map a_3 to the same element in $\{a, b, c\}$, that is $h'(a_3) = h(a_3)$, then $\boldsymbol{s}$ has a square '$h(a_3)h(a_3)$' and therefore $\boldsymbol{w}$ has a square — contradiction. When $h'(a_3) \neq h(a_3)$, without loss of generality, we assume $h'(a_3) = c$ and $h(a_3) = a$, then $\boldsymbol{v} = \boldsymbol{uacu}$ and $\boldsymbol{s}' = \boldsymbol{vcv} = \boldsymbol{uacucuacu}$ has a square '$\boldsymbol{cucu}$' and it follows that $\boldsymbol{w}$ has a square — contradiction.
3. If $|\boldsymbol{s}| > \lfloor |\boldsymbol{s}'|/2 \rfloor + 1$, then $\boldsymbol{s} = \boldsymbol{p}h'(a_3)\boldsymbol{u}h(a_3)\boldsymbol{p}h'(a_3)\boldsymbol{u}$, where $\boldsymbol{p}$ is a non-empty prefix of $\boldsymbol{s}$ and $\boldsymbol{u}$ is a subword (possibly empty) of $\boldsymbol{s}$. Also, $\boldsymbol{v} = \boldsymbol{u}h(a_3)\boldsymbol{p}h'(a_3)\boldsymbol{u}$ and $\boldsymbol{s}' = \boldsymbol{v}h'(a_3)\boldsymbol{v} = \boldsymbol{u}h(a_3)\boldsymbol{p}h'(a_3)\boldsymbol{u}h'(a_3)\boldsymbol{u}h(a_3)\boldsymbol{p}h'(a_3)\boldsymbol{u}$. We can see that $\boldsymbol{s}'$ has a square '$h'(a_3)\boldsymbol{u}h'(a_3)\boldsymbol{u}$', and it follows that $\boldsymbol{w}$ has a square — contradiction.

This ends the proof. □

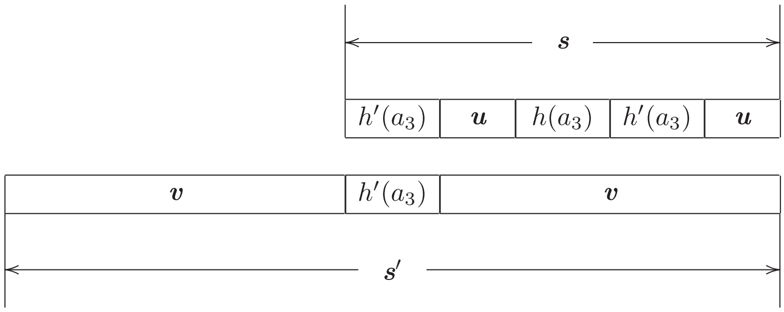

Figure 6. $\boldsymbol{v} = \boldsymbol{u}h(a_3)h'(a_3)\boldsymbol{u}$

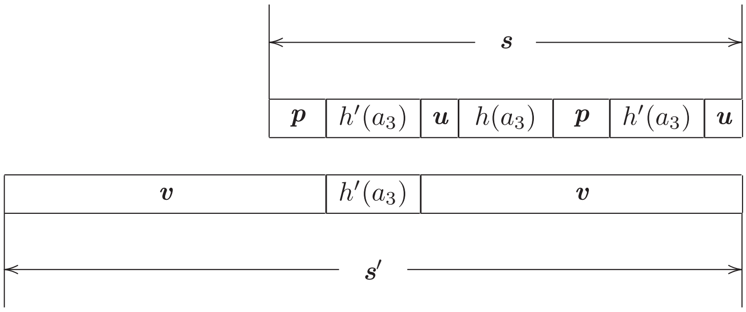

Figure 7. $\boldsymbol{v} = \boldsymbol{u}h(a_3)\boldsymbol{p}h'(a_3)\boldsymbol{u}$

Suppose the class of lists $\mathcal{L}_3$ is inadmissible, and $L \in \mathcal{L}_3$ is a minimum length list that is inadmissible. By Corollary 2 we know that such a list is of length at least eight. Let $L = L_1, L_2, \ldots, L_{n+1}$, where $n \geq 8$, and let $L' = L_1, L_2, \ldots, L_n$, so that $L = L', L_{n+1}$. Then by Theorem 1 every square-free word over L' has a suffix that conforms to the offending suffix $\mathcal{C}(3)$, where the pivot elements are the symbols of the alphabet L_{n+1}. That is, if $\boldsymbol{w}$ is a square-free word over L'^{+}, then there is a non empty suffix $\boldsymbol{s}$ of $\boldsymbol{w}$ and a morphism h such that $h(\mathcal{C}(3)) = \boldsymbol{s}$.

If we are able to replace one of the pivots in $\boldsymbol{s}$ with another element from its respective alphabet, such that the new string $\boldsymbol{w'}$ remains square-free and has no suffix conforming to $\mathcal{C}(3)$, then we can show that L is admissible. Simply, use this $\boldsymbol{w'}$ over L', and append to it a symbol from the alphabet L_{n+1}, such that the resulting string is square-free. We know that such a symbol exists as $\boldsymbol{w'}$ was square-free with no offending suffix.

4 Borders and squares

In this section we relate borders of a string to its squares. There is a vast literature on borders; see for instance [13].

Lemma 4. *A string $\boldsymbol{w}$ is square-free if and only if for every subword $\boldsymbol{s}$ of $\boldsymbol{w}$, if β is a border of $\boldsymbol{s}$, then $|\beta| < \lceil |\boldsymbol{s}|/2 \rceil$.*

Proof. ($\Rightarrow$) Suppose that $\boldsymbol{s}$ is a subword of $\boldsymbol{w}$ and it has a border β such that $|\beta| \geq \lceil |\boldsymbol{s}|/2 \rceil$. From Figure 8 we can see that β must have a prefix $\boldsymbol{p}$ which yields a square $\boldsymbol{pp}$ in $\boldsymbol{s}$ and hence in $\boldsymbol{w}$, and so $\boldsymbol{w}$ is not square-free — contradiction.
($\Leftarrow$) Suppose $\boldsymbol{w}$ has a square $\boldsymbol{s} = \boldsymbol{uu}$. But $\boldsymbol{s}$ is a subword of $\boldsymbol{w}$ and it has a border $\beta = \boldsymbol{u}$ where $|\beta| \geq \lceil |\boldsymbol{s}|/2 \rceil$ — contradiction. □

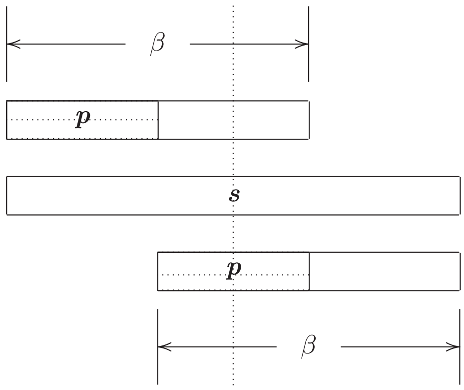

Figure 8. "⇒" direction of the proof for Lemma 4

5 Repetitions and compression

Suppose that we want to encode the $\boldsymbol{w}$'s, as $\langle \boldsymbol{w} \rangle$, in a way that takes advantage of the repetitions in $\boldsymbol{w}$. The intuition, of course, is that strings with long repetitions can be compressed considerably, and so encoded with fewer bits. We can then use the basic Kolmogorov observation about the existence of incompressible strings to deduce that not all strings can have long repetitions. On the other hand, short repetitions are in some sense local, and so they are easier to avoid. Perhaps we can use this approach to prove the existence of square-free strings in $\mathcal{L}_3$.

Assume that the L_i's are ordered, and since each L_i has three symbols, we can encode the contents of each L_i with 2 bits:

Encoding	Symbol
00	1st symbol
01	2nd symbol
10	3rd symbol
11	*separator*

Suppose now that $\boldsymbol{w} = \boldsymbol{w}_1\boldsymbol{v}\boldsymbol{v}\boldsymbol{w}_2$, where $|\boldsymbol{w}| = n, |\boldsymbol{v}| = \ell$, that is, $\boldsymbol{w}$ is a string over L of length n containing a square of length ℓ. Then, we propose the following scheme for encoding $\boldsymbol{w}$'s: $\langle \boldsymbol{w} \rangle := \langle \boldsymbol{w}_1 \rangle 11 \langle \boldsymbol{v} \rangle 11 \langle \boldsymbol{w}_2 \rangle$. A given $\boldsymbol{w}$ does not necessarily have a unique encoding, as it may have several squares; but we insist that the encoding always picks a maximal square (in length). Note also that $\langle \boldsymbol{w} \rangle$ encodes $\boldsymbol{w}$ over L as a string over $\Sigma = \{0, 1\}$.

Note that $|\langle \boldsymbol{w} \rangle| = 2(n - 2\ell) + 2\ell + 4$, where the term $2(n - 2\ell)$ arises from the fact that $|\boldsymbol{w}_1| + |\boldsymbol{w}_2|$ have $n - 2\ell$ symbols (as strings over L), and each such symbol is encoded with two bits, hence $2(n - 2\ell)$. The two separators $11, 11$ take 4 bits, and the length of $\boldsymbol{v}$ is ℓ (as a string over L), and so it takes 2ℓ bits.

It is clear that we can extract $\boldsymbol{w}$ out of $\langle \boldsymbol{w} \rangle$ (uniquely), and so $\langle \cdot \rangle$ is a valid encoding; for completeness, let f decode strings: $f : \Sigma^* \longrightarrow L^+$ work as follows:

$$f(\langle \boldsymbol{w} \rangle) = f(\langle \boldsymbol{w}_1 \rangle 11 \langle \boldsymbol{v} \rangle 11 \langle \boldsymbol{w}_2 \rangle) = \boldsymbol{w}_1\boldsymbol{v}\boldsymbol{v}\boldsymbol{w}_2,$$

and if the input is not a well-formed encoding, say it is 111111, then we let f output, for instance, the lexicographically first string over L^+.

On the other hand, $\langle \cdot \rangle : L^+ \longrightarrow \Sigma^*$ encodes strings by finding the longest square $\boldsymbol{v}$ in a given $\boldsymbol{w}$ (if there are several maximal squares, it picks the first one, i.e., the one where the index of first symbol of $\boldsymbol{v}\boldsymbol{v}$ is smallest), yielding $\boldsymbol{w}_1\boldsymbol{v}\boldsymbol{v}\boldsymbol{w}_2$, and outputting $\langle \boldsymbol{w}_1 \rangle 11 \langle \boldsymbol{v} \rangle 11 \langle \boldsymbol{w}_2 \rangle$.

Suppose now that for a given $L = L_1, L_2, \ldots, L_n$ every string has a maximal square of size at least ℓ_0. We want to bound how big can ℓ_0 be; to this end, we want to find ℓ_0 such that:

$$2^{2(n-2\ell_0)+2\ell_0+4} < 3^n. \tag{5}$$

The reason is that the term on the left counts the maximal number of possible encodings given the assumption that every string over L has a square of size at least ℓ_0, while the term on the right is the size of $|L^+|$. The inequality expresses that if ℓ_0 is assumed to be too big, then we won't be able to encode all the 3^n strings in L^+.

Since (5) can be simplified to $2^{2n-2\ell_0+4} < 3^n$, and using $\log_2$ on both sides we obtain: $2n - 2\ell_0 + 4 < \log_2 3n < 1.6n$, which gives us $n - \ell_0 + 2 < 0.8n$, and so $\ell_0 > n - 0.8n + 2 > 0.2n$. Thus, given any $L = L_1, L_2, \ldots, L_n$, there always is a $\boldsymbol{w} \in L^+$ with a square no longer than $\frac{1}{5}n$. Can we strengthen this technique to give a Kolmogorov style proof to prove that $\mathcal{L}_3$ is admissible?

References

1. S. Arson: *Proof of the existence of asymmetric infinite sequences (russian).* Mat. Sbornik, 2 1937, pp. 769–779.
2. D. R. Bean, A. Ehrenfeucht, and G. F. McNulty: *Avoidable patters in strings of symbols.* Pacific Journal of Mathematics, 85(2) 1979, pp. 261–294.
3. J. Berstel, A. Lauve, C. Reutenauer, and F. V. Saliola: *Combinatorics on Words: Christoffel Words and Repetitions in Words*, American Mathematical Society, 2008.
4. J. Berstel and D. Perrin: *The origins of combinatorics of words.* Electronic Journal of Combinatorics, 28 2007, pp. 996–1022.
5. J. Cooper and D. Rorabaugh: *Bounds on zimin word avoidance.* Electronic Journal of Combinatorics, 21(1) 2014.
6. J. D. Currie: *Which graphs allow infinite non-repetitive walks?* Discrete Mathematics, 87 1991, pp. 249–260.
7. J. Grytczuk, J. Kozik, and P. Micek: *A new approach to nonrepetitive sequences.* arXiv:1103.3809, December 2010.
8. J. Leech: *A problem on strings of beads.* Mathematical Gazette, December 1957, p. 277.
9. N. Mhaskar and M. Soltys: *Non-repetitive string over alphabet list*, in WALCOM: Algorithms and Computation, vol. 8973 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2015, pp. 270–281.
10. M. Morse and G. A. Hedlund: *Unending chess, symbolic dynamics and a problem in semigroups.* Duke Math. J, 11 1944, pp. 1–7.
11. N. Rampersad and J. Shallit: *Repetitions in words.* May 2012.
12. J. Shallit: *A second course in formal languages and automata theory*, Cambridge Univeristy Press, 2009.
13. B. Smyth: *Computing Patterns in Strings*, Pearson Education, 2003.
14. A. Thue: *Über unendliche Zeichenreichen.* Norsek Vid. Selsk. Srk., I Mat. Nat. Kl., 7 1906, pp. 1–22.
15. A. Thue: *Über die gegenseitige lage gleicher teile gewisser Zeichenreihen.* Kra. Vidensk. Selsk. Skrifter., I. Mat. Nat. Kl., 1 1912, pp. 1–67.
16. A. I. Zimin: *Blocking sets of terms.* Mat. Sbornik, 119 1982, pp. 363–375.