Journal of Integer Sequences, Vol. 7 (2004),
Article 04.2.7

The Number of Ternary Words Avoiding Abelian Cubes Grows Exponentially

Ali Aberkane & James D. Currie[1]
Department of Mathematics and Statistics
University of Winnipeg
Winnipeg, Manitoba R3B 2E9
Canada
aberkane@iml.univ-mrs.fr
currie@uwinnipeg.ca

Narad Rampersad
School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
Canada
nrampersad@math.uwaterloo.ca

Abstract

We show that the number of ternary words of length n avoiding abelian cubes grows faster than r^n, where $r = 2^{1/24}$.

1 Introduction

The study of words avoiding repetitions is an area of combinatorics on words reaching back to at least the turn of the century and the work of Thue. A word of the form xx, x some non-empty word, is called a *square*. A word of the form xxx, x some non-empty word, is called a *cube*. A word containing no squares (cubes) is called *square-free* (*cube-free*).

[1]This author's research was supported by an NSERC operating grant.

The longest square-free words over the 2-letter alphabet $\{a, b\}$ are aba and bab. Nevertheless, in 1906 Thue showed that over a 3-letter alphabet, there are arbitrarily long square-free words [17]. Thue also showed that there are arbitrarily long cube-free words over a 2-letter alphabet.

How many square-free words over $\{a, b, c\}$ are there? An exact answer remains elusive. Let $c(n)$ be the number of ternary square-free words of length n. In 1997 [15] it was still of interest even to calculate $c(46)$. A recent article [13] gives values to $c(110)$. Exponential upper and lower bounds on $c(n)$ have been steadily improved in a series of articles by various authors [4, 3, 2, 10, 13, 19]. The current best lower bound, given in [19], is $c(n) \geq 110^{n/42} = (1.118419\ldots)^n$. According to Grimm [13], an exponential upper bound has base $8416550317984^{\frac{1}{108}} = 1.317277\ldots$

In algebraic problems, commutativity is usually a simplifying assumption. However, for word repetitions, the commutative problems have been hard to crack. An *abelian square* is a word x_1x_2 where x_2 can be obtained from x_1 by rearranging letters. The study of abelian squares was initiated by Erdos [11], who asked whether an infinite sequence over a finite alphabet existed which was square-free in an abelian sense; no two adjacent blocks were to be permutations of one another. Not only 1234 1234 is forbidden in such a sequence, but also 1234 2134, since 1234 and 2134 are permutations of each other. A sequence avoiding squares in this abelian sense was discovered by Evdokimov [12], but he used an alphabet of 25 letters. The alphabet size was reduced to 5 letters by Pleasants [16], and not until 1992, to a 4-letter alphabet by Keränen [14]. One checks that on a 3-letter alphabet there are only finite strings which are square-free in this abelian sense, so that Keränen's result is optimal. Dekking [9] showed that the smallest alphabet on which cubes were avoidable in this abelian sense had 3 symbols, while 2 symbols were necessary and sufficient to avoid fourth powers in the abelian sense.

Carpi [5, 6] showed that the number of words over $\{0, 1, 2, 3\}$ avoiding abelian squares grows exponentially with length. Recently one of the authors [7] showed that the number of binary words avoiding abelian fourth powers grows exponentially with length. In the following article we "finish off" the abelian growth problems by solving the following problem from [8]:

Problem 1.1 Show that the number of abelian cube-free ternary words grows exponentially with length.

The number of ternary words avoiding abelian cubes for $n = 0, 1, \ldots 13$ is

$$1, 3, 9, 24, 66, 180, 468, 1206, 3150, 7998, 20124, 50520, 124044, 303906$$

and is sequence A096168 in the Encyclopedia of Integer Sequences.

2 Preliminaries

We say that $x \sim y$ for words x and y if the number of occurrences each letter is identical in x and in y. For example, $123342 \sim 321342$. We say that word w *encounters an abelian cube* if w has a subword $x_1x_2x_3$ with $x_i \sim x_{i+1}$, $i = 1, 2$, $x_1 \neq \epsilon$.

We use $\mathbb{Z}$ to denote the set of integers, and R^n to denote the set of $1 \times n$ matrices (i.e., row vectors) with entries in ring R.

A *multi-valued substitution* is a function $h : \Sigma^* \to \mathcal{P}(T^*)$ where $\mathcal{P}(T^*)$ is the set of subsets of T^*. Let $v \in \Sigma^*$, $v = v_1 v_2 v_3 \ldots v_k$, $v_i \in \Sigma$, $i = 1, 2, \ldots, k$. We say that word w is an image of word v under multi-valued substitution h if we can write $w = w_1 w_2 w_3 \ldots w_k$ where $w_i \in h(v_i)$, $i = 1, 2, 3, \ldots, k$. We write $w \in h(v)$. In the case that $\Sigma = T$, we call h a *multi-valued substitution on* Σ.

Fix a natural number m, and let Σ be the alphabet $\{0, 1, \ldots, m-1\}$. If w is a word , we define $f(w) = [|w|_0, |w|_1, \ldots, |w|_{m-1}]$. In other words, $f(w)$ is a row vector which counts the occurrences of $0, 1, \ldots, m-1$ in w, and for $w, v \in \Sigma^*$ we have $w \sim v$ exactly when $f(w) = f(v)$. Suppose that h is a multi-valued substitution on Σ such that for each $i \in \Sigma$ we have $f(u) = f(v)$ whenever $u, v \in h(i)$. In this case we say that h is *single-valued up to permutation*. Let the *incidence matrix* M of h be the $m \times m$ matrix with ith row $f(v)$, any $v \in h(i)$. We observe that if w is an image of v under h, then $f(w) = f(v)M$.

Now fix $m = 3$ and $\Sigma = \Sigma$. Consider the multi-valued substitution h on Σ given by

$$\begin{aligned} h(0) &= \{001002\} \\ h(1) &= \{110112\} \\ h(2) &= \{002212, 122002\} \end{aligned}$$

The corresponding matrix M is thus

$$\begin{bmatrix} 4 & 1 & 1 \\ 1 & 4 & 1 \\ 2 & 1 & 3 \end{bmatrix}$$

Remark 2.1 Let $v \in \mathbb{Z}^3$. It is an exercise in linear algebra to check that $uM = v$ has a solution $u \in \mathbb{Z}^3$ if and only if

$$v \begin{bmatrix} 1 \\ 13 \\ 19 \end{bmatrix} \in 36\mathbb{Z}^1.$$

Let $L = \{w : uwv \in h^n(0)$ for some positive integer n, some words $u, v\}$. Thus L is the set of words contained in some image of 0 under iterations of h. Set L is closed under h, and each word of L is a subword of a word of $h(L)$. If $w \in L$ then for some letters $a, b \in \Sigma$ we have $awb \in L$. We will prove

Theorem 2.2 (Main Theorem) *Set L contains no abelian cubes.*

Remark 2.3 It is not the case that h is abelian cube-free. That is, words of $h(u)$ may contain abelian cubes, even u does not. For example,

$$001002110112001002110112 \in h(0101), \text{ but } 0101$$

contains no abelian cubes, while 0010 021101120010021 10112 contains an abelian cube, since $02110 \sim 11200 \sim 10021$.

3 Templates and Parents

A *template* is a 6-tuple $[a, b, c, d, v_1, v_2]$ where $a, b, c, d \in \{\epsilon, 0, 1, 2\}$ and $v_1, v_2 \in \mathbb{Z}^3$. We say that a word w *realizes* template $[a, b, c, d, v_1, v_2]$ if we can write $w = aX_1bX_2cX_3d$ where $f(X_2) - f(X_1) = v_1$, $f(X_3) - f(X_2) = v_2$.

Remark 3.1 Suppose that $X \in L$, $|X| \geq 5$. We can write $X = a''Yb'$ where

- for some $Z \in L$, $Y \in h(Z)$
- for some $A, B \in \{0, 1, 2, \epsilon\}$, there exist words a', b'' such that $a'a'' \in h(A)$, $b'b'' \in h(B)$.

Let $t_1 = [a, b, c, d, v_1, v_2]$ and $t_2 = [A, B, C, D, w_1, w_2]$ be templates. We say that t_2 is a *parent* of t_1 if $A, D \neq \epsilon$, and for some words $a', a'', b', b'', c', c'', d', d''$ we have $a'aa'' \in h(A)$, $b'bb'' \in h(B)$, $c'cc'' \in h(C)$, $d'dd'' \in h(D)$, while

$$\begin{aligned} v_1 - f(b''c') + f(a''b') &= w_1 M \\ v_2 - f(c''d') + f(b''c') &= w_2 M. \end{aligned}$$

Given a template t_1, we may calculate all its parents. The set of candidates for A, B, C, D and hence for $a', a'', b', b'', c', c'', d', d''$ is finite, and may be searched exhaustively. Since M is invertible, a choice of values for $a', a'', b', b'', c', c'', d', d''$, together with given values v_1 and v_2, determines w_1 and w_2. Note that not all computed values for w_1, w_2 are in $\mathbb{Z}^3$, some templates have no parents. For example, an exhaustive search shows that $[2, 1, 0, 1, [0, 0, 0], [0, 0, 0]]$ has no parents.

Lemma 3.2 *Let $t_1 = [a, b, c, d, v_1, v_2]$ and $t_2 = [A, B, C, D, w_1, w_2]$ be templates, with t_2 a parent of t_1. Suppose that some word $u_2 \in L$ realizes t_2. Then some subword u_1 of a word of $h(u_2)$ realizes t_1. The word u_1 is in L.*

Proof: Of course if u_1 is a subword of a word of $h(u_2)$, and u_2 is in L, then u_1 is in L.

Write $u_2 = AZ_1BZ_2CZ_3DZ_4$ where $f(Z_2) - f(Z_1) = w_1$, $f(Z_3) - f(Z_2) = w_2$. Since t_2 is a parent of t_1, find words $a', a'', b', b'', c', c'', d', d''$ so that $a'aa'' \in h(A)$, $b'bb'' \in h(B)$, $c'cc'' \in h(C)$, $d'dd'' \in h(D)$, and

$$\begin{aligned} v_1 - f(b''c') + f(a''b') &= w_1 M \\ v_2 - f(c''d') + f(b''c') &= w_2 M. \end{aligned}$$

Choose any words Y_1, Y_2, Y_3 with $Y_i \in h(Z_i)$. This means that

$$a'aa''Y_1b'bb''Y_2c'cc''Y_3d'dd'' \in h(AZ_1BZ_2CZ_3D).$$

Let $u_1 = aa''Y_1b'bb''Y_2c'cc''Y_3d'd$. Then u_1 realizes t_1, which is verified by letting $X_1 = a''Y_1b'$, $X_2 = b''Y_1c'$, $X_3 = c''Y_1d'$. We see that $u_1 = aX_1bX_2cX_3d$, and

$$\begin{aligned} f(X_2) - f(X_1) &= f(b''Y_1c') - f(a''Y_1b') \\ &= f(b''c') - f(a''b') + f(Y_2) - f(Y_1) \\ &= f(b''c') - f(a''b') + f(Z_2)M - f(Z_1)M \\ &= f(b''c') - f(a''b') + (f(Z_2) - f(Z_1))M \\ &= f(b''c') - f(a''b') + w_1 M \\ &= v_1. \end{aligned}$$

Similarly, $f(X_3) - f(X_2) = v_2$. Thus u_1 realizes t_1, as desired .$\square$

Lemma 3.3 *Let $t_1 = [a,b,c,d,v_1,v_2]$ be a template. Suppose that some word $u_1 \in L$ realizes t_1, with $|u_1| > 17$. Suppose further that*

$$[-1] \leq v_1 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, v_2 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, (v_1+v_2) \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \leq [1].$$

Then there is a template $t_2 = [A,B,C,D,w_1,w_2], A, B \neq \epsilon$ which is a parent of t_1, and a word $u_2 \in L$ which realizes t_2.

Proof: Clearly, u_2 will be shorter than u_1.

Write $u_1 = aX_1bX_2cX_3d$ where $f(X_2) - f(X_1) = v_1$, $f(X_3) - f(X_2) = v_2$. The condition on v_1, v_2 and $v_1 + v_2$ implies that $|X_1| - 1 \leq |X_2|, |X_3| \leq |X_1| + 1$. Thus

$$\begin{aligned} 17 &\leq |u_1| \\ &= |aX_1bX_2cX_3d| \\ &= |a| + |X_1| + |b| + |X_2| + |c| + |X_3| + |d| \\ &\leq 3|X_1| + 2, i = 1,2,3. \end{aligned}$$

We deduce that $|X_1| \geq 5$. Similarly, $|X_2|, |X_3| \geq 5$. Rewriting $X_1 = a''Y_1b'$, etc, we find there must be a word $a'u_1d'' \in h(L)$ with

$$\begin{aligned} a'u_1d'' &= a'aX_1bX_2cX_3dd'' \\ &= a'aa''Y_1b'bb''Y_2c'cc''Y_3d'dd'' \\ &\in h(u_2) \end{aligned}$$

for some word

$$u_2 = AZ_1BZ_2CZ_3D \in L$$

satisfying

$$\begin{aligned} Y_i &\in h(Z_i), \; i = 1,2,3 \\ A,B,C,D &\in \{\epsilon, 0, 1, 2\} \\ b'bb'' &\in h(B) \\ c'cc'' &\in h(C) \end{aligned}$$
$$aa'' \text{ is a suffix of an element of } h(A)$$
$$d'd \text{ is a prefix of an element of } h(D).$$

If a solution u_2 of these conditions has $A = \epsilon$, then by force, $aa'' = \epsilon$. Since ϵ is also a suffix of $h(0)$, another solution $\hat{u}_2$ exists, differing from u_2 only by changing A to 0. We may therefore assume that $A \neq \epsilon$. Similarly, assume $D \neq \epsilon$. Given a solution u_2 of our conditions, let $w_1 = f(Z_2) - f(Z_1), w_2 = f(Z_3) - f(Z_2)$. As in the previous lemma,

$$\begin{aligned} v_1 - f(b''c') + f(a''b') &= w_1 M \\ v_2 - f(c''d') + f(b''c') &= w_2 M. \end{aligned}$$

It follows that $t_2 = [A, B, C, D, w_1, w_2]$ is a parent of t_1, realized by a word $u_2 \in L$.□

A template $t = [a, b, c, d, v_1, v_2]$ *implies a cube* if either

$$f(a) - f(b) = v_1 \text{ and } f(b) - f(c) = v_2 \textbf{ OR } f(b) - f(c) = v_1 \text{ and } f(c) - f(d) = v_2.$$

Suppose that u realizes $t = [a, b, c, d, v_1, v_2]$, and t implies a cube. Write $u = aX_1bX_2cX_3d$ where $v_1 = f(X_2) - f(X_1)$, $v_2 = f(X_3) - f(X_2)$. We suppose that $f(a) - f(b) = v_1$ and $f(b) - f(c) = v_2$. (The other case is similar.) In this case, u contains the abelian cube $aX_1bX_2cX_3$. We have $f(aX_1) - f(bX_2) = f(a) - f(b) - (f(X_2) - f(X_1)) = v_1 - v_1 = [0, 0, 0]$. Thus $aX_1 \sim bX_2$. Similarly, $bX_2 \sim cX_3$, $aX_1bX_2cX_3$ is an abelian cube, as claimed.

A set T of templates is *closed* if whenever $t_1 \in T$ and t_2 is a parent of t_1, then $t_2 \in L$ if t_2 does not imply a cube.

Let $t_c = [\epsilon, \epsilon, \epsilon, \epsilon, [0, 0, 0], [0, 0, 0]]$. An abelian cube realizes t_c. To show that L is abelian cube-free, we start by finding a closed set T_c of templates containing t_c, and checking that whenever $t = [a, b, c, d, v_1, v_2] \in T_c$, then

$$[-1] \leq v_1 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, v_2 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, (v_1 + v_2) \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \leq [1].$$

Suppose now that u_1 is the shortest word of L realizing a template t_1 of T_c. If $|u_1| \geq 17$, by Lemma 3.3, a parent t_2 of t_1 is realized by a word $u_2 \in L$ which is shorter than u_1. The minimality of $|u_1|$ implies that $t_2 \notin T_c$. Since T_c is closed, this means that t_2 implies a cube. It follows that u_2 realizes $t_c \in T_c$. This is a contradiction. Thus $|u_1| < 17$.

We then generate the set of all words of L of length 16. A finite search shows that none of these words realizes a template in T_c. It follows that no word u_1 can exist, so that L is abelian cube-free.

4 Computations

Given a template t, the MAPLE procedure `parents` in Appendix 1 generates all of t's parents which do not imply a cube. Given a set `seed` of templates, the procedure `grow` generates the set `closure`, the smallest closed set containing `seed`. A Pentium 4 running MAPLE 7 at 2.53 GHz ran `grow` on $t_c = [\epsilon, \epsilon, \epsilon, \epsilon, [0, 0, 0], [0, 0, 0]]$ in 4.6 seconds. We find that $|T_c| = 103$. The output T_c is also observed to have the property that if $t = [a, b, c, d, v_1, v_2] \in T_c$, then

$$[-1] \leq v_1 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, v_2 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, (v_1 + v_2) \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \leq [1].$$

(We thus observe *a posteori* both that T_c is finite, and that its various v_1, v_2 have this desirable property.) Another observation is that the only $t \in T_c$ for which any of a, b, c, d is ϵ is $t = t_c$.

Since images of letters under h have length 6, we see that any word w of L with $|w| = 16$ is a subword of $h(a_1a_2a_3a_4)$ for some word $a_1a_2a_3a_4 \in L$ with the a_1, a_2, a_3, a_4 letters. We see in turn, that $a_1a_2a_3a_4$ will be a subword of $h(\{00, 01, 02, 10, 11, 12, 20, 21, 22\})$, and thus of $h(112001002122)$, since each of $00, 01, 02, 10, 11, 12, 20, 21, 22$ is a subword of 112001002122. Word 112001002122 is itself a subword of a word in $h(102)$.

We obtain the set all of words of length 16 in L as follows:

1. Apply h to $\{112001002122\}$ to obtain a set `imageSet`
2. Collect all subwords of length 4 from the words of `imageSet` to give a set `length4Words`. We find that `length4Words`, and hence L, contains exactly 27 words of length 4.
3. Apply h to `length4Words` to obtain `imageSet4`. There are 80 words in this set.
4. Build `testSet` by collecting all subwords of length 16 from the words of the resulting `imageSet4`. We thus find that there are 278 words of length 16 in L.

For each template t in T_c and each word w of length 16 in L, viz., for each $w \in$ `testSet`, we verify that no subword of w realizes t. This is done in two stages:

1. For $w \in$ `testSet` and $t \in T_c - \{t_c\}$, we verify that no subword of w realizes t using procedure `occurs`. (The procedure `occurs` takes advantage of the fact that if $[a, b, c, d, v_1, v_2] \in T_c - \{t_c\}$, then $a, b, c, d \neq \epsilon$.)
2. For $w \in$ `testSet`, we verify that no subword of w realizes t_c using procedure `cubeOccurs`.

The total time for these verifications is 60.4 seconds. The computations establish the Main Theorem.

5 Exponential Growth

Let w be the prefix of a word of $h^\omega(0)$ such that $|w| = n$. We can write $w \in h(v)p$ where v is a prefix of a word of $h^\omega(0)$, and $|p| \leq 5$. By our Main Theorem, every word in $h(v)p$ avoids abelian cubes. Since each image of a letter under h contains a 2, $|v|_2 \geq (|v| - 5)/6$. Since $h(2)$ contains 2 words, we see that $h(v)$, and hence $h(v)p$ contains at least $2^{(|v|-5)/6}$ words. Also, $n = |w| = |h(v)p| \leq 6|v| + 5$. Then $2^{(|v|-5)/6} \geq 2^{((n-5)/6-5)/6} = 2^{-35/36}2^{n/36}$. It follows that $h(v)p$ contains at least cr^n abelian cube-free words of length n, where $c = 2^{-35/36}$ and $r = 2^{1/36}$.

The base $2^{1/36}$ may be sharpened. By a slight variation on Theorem 8.4.6 in [1], letter frequencies in $h^\omega(0)$ can be obtained from the normalized eigenvector of the incidence matrix M corresponding to the dominant eigenvalue. (By Perron-Frobenius theory, this eigenvalue has multiplicity 1 and its corresponding eigenvector is non-negative.) We find that for M the dominant eigenvalue is 6 and the corresponding normalized eigenvector is [5/12,1/3,1/4].

This means that the asymptotic frequency of 2's is 1/4. Using $|v|_2 \sim |v|/4$ in the previous paragraph would show the number of abelian cube-free words of length n over $\{0,1,2\}$ to be $\Omega(2^{n/24})$, rather than $\Omega(2^{n/36})$.

Theorem 5.1 (Second Main Theorem) *The number of abelian cube-free words of length n over $\{0,1,2\}$ is $\Omega(2^{n/24})$.*

6 Acknowledgments

Special thanks to T. I. Visentin for a suggestion which significantly speeded up the MAPLE code. Thanks also to the anonymous referee.

Appendix: Code

A MAPLE worksheet with all code referred to in this paper is available from

`www.uwinnipeg.ca/~currie/AbelianCubes.mws`

Here are the most important pieces:

The following MAPLE code sets up an array `split` which is used by `parents`:

```
For a letter a, split[a]={[f(a'),f(a"),alpha]:h(alpha)=a'aa"}.
Here 3 stands for the empty/missing letter \epsilon. We let
split[3]={[f(a'),f(a"),alpha]:h(alpha)=a'a"}. We represent these
sets by lists.}

 >split[0]:=[[[0,0,0],[3,1,1],0],[[1,0,0],[2,1,1],0],
[[2,1,0],[1,0,1],0],[[3,1,0],[0,0,1],0],[[0,2,0],[0,2,1],1],
[[0,0,0],[1,1,3],2],[[1,0,0],[0,1,3],2],[[0,1,2],[1,0,1],2],
[[1,1,2],[0,0,1],2]];
>split[1]:=[[[2,0,0],[2,0,1],0],[[0,0,0],[1,3,1],1],
[[0,1,0],[1,2,1],1],[[1,2,0],[0,1,1],1],[[1,3,0],[0,0,1],1],
[[2,0,2],[0,0,1],2],[[0,0,0],[2,0,3],2]];
>split[2]:=[[[2,0,1],[0,1,1],2],[[4,1,0],[0,0,0],0],
[[1,4,0],[0,0,0],1],[[2,0,0],[0,1,2],2],[[2,1,2],[0,0,0],2],
[[0,1,0],[2,0,2],2],[[0,1,1],[2,0,1],2]];}
>split[3]:=[[[0,0,0],[0,0,0],3],[[1,2,0],[0,2,1],1],
[[0,0,0],[0,0,0],3],[[0,0,0],[4,1,1],0],[[1,0,0],[3,1,1],0],
[[2,0,0],[2,1,1],0],[[2,1,0],[2,0,1],0],[[3,1,0],[1,0,1],0],
[[4,1,0],[0,0,1],0],[[0,0,0],[1,4,1],1],[[0,1,0],[1,3,1],1],
[[0,2,0],[1,2,1],1],[[1,3,0],[0,1,1],1],[[1,4,0],[0,0,1],1],
[[0,0,0],[2,1,3],2],[[1,0,0],[1,1,3],2],[[2,0,0],[0,1,3],2],
[[2,0,1],[0,1,2],2],[[2,0,2],[0,1,1],2],[[2,1,2],[0,0,1],2],
[[0,1,0],[2,0,3],2],[[0,1,1],[2,0,2],2],[[0,1,2],[2,0,1],2],
[[1,1,2],[1,0,1],2]];
```

Given a template t, the MAPLE procedure parents generates all of t's parents which do not imply a cube.

```
> parents:=proc(template)

Given a template t1 = [a, b, c, d, v1, v2], the following
procedure finds the set of all parents of t. We rewrite u1 =
aX1bX2Cx3d  as u1=aa"Y1b'bb"Y2c'cc"Y3d'd in all possible ways. (We
use the symbol '3' for the empty/missing letter \epsilon, so if b
= 3, then aX1bX2cX3d = aX1X2cX3d, for example.) For a given way of
rewriting u1, we should have
       v1-f(b"c') + f(a"b') = f(Y2)-f(Y1),
       v2-f(c"d') + f(b"c') = f(Y3) -f(Y2)
To find potential parents, we let a', a" range over possible
``splits"  of h(a), etc, and test whether each of v1 - f(b"c') +
f(a"b') and v2 - f(c"d') + f(b"c') is an integer combination of
parikh vectors of h(0), h(1), h(2). In this case we solve
(wi)M=f(Y(i+1))-f(Yi). The parent is then t2 = [A,B,C,D,w1,w2]
where h(A)=a'aa", etc. The values of A, B, C, D are available as
the third component of the ``split" array. We suppress parents
which imply cubes.

> a:=template[1];b:=template[2];c:=template[3];
> d:=template[4];v1:=template[5];v2:=template[6];
> parentSet:={};
> for A in split[a] do for B in split[b] do for C in split[c] do
> for D in split[d] do
>    if not((A[3]=3) or (D[3]=3)) then
>>      As:=A[2];Bp:=B[1];Bs:=B[2];Cp:=C[1];Cs:=C[2];Dp:=D[1];
>       if (
> ((v1[1]+As[1]+Bp[1]-Bs[1]-Cp[1]+13*(v1[2]+As[2]+Bp[2]-Bs[2]-Cp[2])
> +19*(v1[3]+As[3]+Bp[3]-Bs[3]-Cp[3]) )mod 36 = 0) and
> ((v2[1]+Bs[1]+Cp[1]-Cs[1]-Dp[1]+13*(v2[2]+Bs[2]+Cp[2]-Cs[2]-Dp[2])
> +19*(v2[3]+Bs[3]+Cp[3]-Cs[3]-Dp[3]) )mod 36 = 0)   ) then

To speed up this procedure, some linear algebra is done ``long
hand".  The above condition is that v1[1,13,19]^T ,v1[1,13,19]^T
are in 36Z^3.  The following computation is wi = ( f(Y(i+1))-f(Yi)
)M^{-1}:

> w1:=[11/36*(v1[1]+As[1]+Bp[1]-Bs[1]-Cp[1])-1/36*(v1[2]+
> As[2]+Bp[2]-Bs[2]-Cp[2])-7/36*(v1[3]+As[3]+Bp[3]-Bs[3]-
> Cp[3]),-1/18*(v1[1]+As[1]+Bp[1]-Bs[1]-Cp[1])+5/18*(v1[2]+
> As[2]+Bp[2]-Bs[2]-Cp[2])-1/18*(v1[3]+As[3]+Bp[3]-Bs[3]-
> Cp[3]),-1/12*(v1[1]+As[1]+Bp[1]-Bs[1]-Cp[1])-1/12*(v1[2]+
```

```
> As[2]+Bp[2]-Bs[2]-Cp[2])+5/12*(v1[3]+As[3]+Bp[3]-Bs[3]-
> Cp[3])];
> w2:=[11/36*(v2[1]+Bs[1]+Cp[1]-Cs[1]-Dp[1])-1/36*(v2[2]+
> Bs[2]+Cp[2]-Cs[2]-Dp[2])-7/36*(v2[3]+Bs[3]+Cp[3]-Cs[3]-
> Dp[3]),-1/18*(v2[1]+Bs[1]+Cp[1]-Cs[1]-Dp[1])+5/18*(v2[2]+
> Bs[2]+Cp[2]-Cs[2]-Dp[2])-1/18*(v2[3]+Bs[3]+Cp[3]-Cs[3]-
> Dp[3]),-1/12*(v2[1]+Bs[1]+Cp[1]-Cs[1]-Dp[1])-1/12*(v2[2]+
> Bs[2]+Cp[2]-Cs[2]-Dp[2])+5/12*(v2[3]+Bs[3]+Cp[3]-Cs[3]-
> Dp[3])];
```

```
 We will ignore parents which imply cubes
```

```
>             check1:=[0,0,0]; check1[A[3]+1]:=1;
>             check2:=[0,0,0]; if not(B[3]=3) then
>                check2[B[3]+1]:=1 fi;
>             check3:=[0,0,0]; if not(C[3]=3) then
>                check3[C[3]+1]:=1 fi;
>             check4:=[0,0,0]; check4[D[3]+1]:=1;
>             diff1:=check1-check2;
```

```
Here 'diff1' is f(A) - f(B), for deciding if t2 implies a cube
```

```
>             diff2:=check2-check3;
>             diff3:=check3-check4;
>             cube:=((w1=diff1)and(w2=diff2)) or
>              ((w1=diff2)and(w2=diff3));
>             if not(cube) then
>                parentSet:=parentSet union
>                 {[A[3],B[3],C[3],D[3],w1,w2]};
>             fi;
>       fi;
>    fi;
> od;od;od;od;
> return(parentSet);
> end;
```

Given a set `seed` of templates, the procedure `grow` generates the set `closure`, the smallest closed set containing `seed`:

```
> grow:=proc(T)
> seed:=T; closure:={};
> while not(seed={}) do
>    for t in seed do
>       closure:=closure union {t};
>       candidatesForNewParents:=parents(t);
```

```
>       seed:=(seed union candidatesForNewParents) minus
>        closure;
>    od
> od;
> return(closure);
> end;
```

For $t \in T_c - \{t_c\}$, and $w \in$ `testSet`, we verify that no subword of w realizes t using procedure `occurs`:

```
> occurs:=proc(template,aWord)
Notice that in the set Tc returned by procedure ``grow", there are
no templates other than tc for which a,b,c or d are 3. In this
procedure, we will therefore assume a,b,c,d are letters. A
separate check for tc (which uses empty letters) is in another
procedure. The current procedure takes [a,b,c,d,v1,v2] and a word
and looks for an occurrence of the pattern aX1bX2cX3d in the word.
Again, the ``totals" of v1, v2 and v1+v2 have absolute values at
most 1. We search using ``for loops" on length(X1), length(X2),
length(X3). We have |aWord| >= 4 + |X1| +|X2| +|X3|
>= 4 + 3|X1| -2, so that |X1| <= (|aWord| - 2)/3.
> a:=template[1];b:=template[2];c:=template[3];d:=template[4];
> v1:=template[5];v2:=template[6];L:=length(aWord);
> for L1 from 0 to (L-2)/3 do
>  for L2 from max(0,L1-1) to L1+1 do
>   for L3 from max(0,L1-1,L2-1) to min(L1+1,L2+1) do
>    for iStart from 1 to L-(L1+L2+L3+3) do
>     if (
> (numString(a)=substring(aWord,iStart..iStart))
> and (numString(b)=substring(aWord,iStart+L1+1..iStart+L1+1))
> and (numString(c)=substring(aWord,iStart+L1+L2+2..iStart+L1+L2+2))
> and (numString(d)=substring(aWord,iStart+L1+L2+L3+3..iStart+L1+L2
>     +L3+3)))
>     then
>      X1:=substring(aWord,iStart+1..iStart+L1);
>      X2:=substring(aWord,iStart+L1+2..iStart+L1+L2+1);
>      X3:=substring(aWord,iStart+L1+L2+3..iStart+L1+L2+L3+2);
>      if ((f(X2)-f(X1)=v1) and (f(X3)-f(X2)=v2)) then
>       return(true)
>      fi
>     fi
>    od
>   od
>  od
> od;
```

```
> return(false)
> end;
```

For $w \in$ `testSet`, we verify that no subword of w realizes t_c using procedure `cubeOccurs`:

```
> cubeOccurs:=proc(aWord)
> L:=length(aWord);
> for L1 from 1 to L/3 do
>    for iStart from 1 to L-3*L1+1 do
>       X1:=substring(aWord,iStart..iStart+L1-1);
>       X2:=substring(aWord,iStart+L1..iStart+2*L1-1);
>       X3:=substring(aWord,iStart+2*L1..iStart+3*L1-1);
>       if ((f(X1)=f(X2)) and (f(X3)=f(X2))) then
>          return(true)
>       fi
>    od
> od;
> return(false)
> end;
```

References

[1] J.-P. Allouche & J.O. Shallit, *Automatic Sequences: Theory, Applications, Generalizations*, Cambridge University Press, 2003.

[2] M. Baake, V. Elser and U. Grimm, The entropy of square-free words, *Math. Comput. Modelling* **26** (1997), 13–26.

[3] F. J. Brandenburg, Uniformly growing k-th power-free homomorphisms, *Theoret. Comput. Sci.* **23** (1983), 69–82.

[4] J. Brinkhuis, Non-repetitive sequences on three symbols, *Quart. J. Math. Oxford* **(2) 34** (1983), 145–149.

[5] A. Carpi, On the number of Abelian square-free words on four letters, *Disc. Appl. Math*, **81** (1998) 155–167.

[6] A. Carpi, On Abelian squares and substitutions, *Theoret. Comp. Sci.*, **218** (1999) 61–81.

[7] J. D. Currie, The number of binary words avoiding abelian fourth powers grows exponentially, *Theoret. Comp. Sci.* To appear.

[8] J. D. Currie, Pattern avoidance: themes and variations, *Theoret. Comp. Sci.*. To appear.

[9] F. M. Dekking. Strongly non-repetitive sequences and progression-free sets, *J. Comb. Theory Ser. A* **27** (1979), 181-185.

[10] S. B. Ekhad and D. Zeilberger, There are more than $2^{(n/17)}$ n-letter ternary square-free words, *J. Integer Seq.* **1** (1998) 98.1.9.

[11] P. Erdös, Some unsolved problems, *Magyar Tud. Akad. Mat. Kutató. Int. Közl.* **6** (1961), 221–254.

[12] A. A. Evdomikov, Strongly asymmetric sequences generated by a finite number of symbols, *Dokl. Akad. Nauk. SSSR* **179** (1968), 1268–1271; *Soviet Math. Dokl.* **9** (1968), 536–539.

[13] U. Grimm, Improved bounds on the number of ternary square-free words, *J. Integer Seq.* **4** (2001) 01.2.7.

[14] V. Keränen, Abelian squares are avoidable on 4 letters, *Automata, Languages and Programming: Lecture notes in Computer Science* **623** (1992) Springer-Verlag, 41–52.

[15] J. Noonan & D. Zeilberger, The Goulden-Jackson cluster method: extensions, applications and implementations, *J. Difference Eq. Appl.* **5** (1999), 355–377.

[16] P. A. B. Pleasants, Non-repetitive sequences, *Proc. Cambridge Philos. Soc.* **68** (1970) 267–274.

[17] A. Thue, Über unendliche Zeichenreihen, *Norske Vid. Selsk. Skr. I. Mat. Nat. Kl. Christiana* **7** (1906) 1–22.

[18] A. Thue, Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen, *Norske Vid. Selsk. Skr. I. Mat. Nat. Kl. Christiana* No. 1 (1912) 1–67.

[19] X. Sun, New lower-bound on the number of ternary square-free words, *J. Integer Seq.* **6** (2003) 03.3.2.

2000 *Mathematics Subject Classification*: Primary 05A20; Secondary 68R15.
Keywords: Combinatorics on words, abelian repetitions, enumeration.

(Concerned with sequence A096168.)

Received April 7 2004; revised version received June 16 2004. Published in *Journal of Integer Sequences*, June 19 2004.

Return to Journal of Integer Sequences home page.