Powers and Anti-Powers in Binary Words

by

Samin Riasat

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2019

© Samin Riasat 2019
I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.
Abstract

Fici et al. recently introduced the notion of anti-powers in the context of combinatorics on words. A power (also called tandem repeat) is a sequence of consecutive identical blocks. An anti-power is a sequence of consecutive distinct blocks of the same length. Fici et al. showed that the existence of powers or anti-powers is an unavoidable regularity for sufficiently long words. In this thesis we explore this notion further in the context of binary words and obtain new results.
Acknowledgements

I would like to thank my supervisor Jeffrey Shallit for supervising this thesis and for giving me many interesting problems to think about. I would also like to thank Dan Brown and Lila Kari for serving as readers for this thesis. Finally, I would like to thank my family and friends without whose constant support this thesis would not have been possible.
# Table of Contents

<table>
<thead>
<tr>
<th>List of Tables</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
</tbody>
</table>

## 1 Introduction
1.1 Ramsey Theory and Combinatorics on Words .................................................. 1  
1.2 Avoiding Anti-Powers .......................................................................................... 3  
1.3 Abelian Anti-Powers ......................................................................................... 4

## 2 Preliminaries
2.1 Notions and Notations ...................................................................................... 5  
2.2 A Classical Result on Words with Borders ......................................................... 6  
2.3 Parikh Vectors ................................................................................................. 7  
2.4 Previous Work on Anti-Powers ......................................................................... 7

## 3 A Study in $N(k, r)$
3.1 Existence ........................................................................................................ 9  
3.2 Asymptotic Behaviour ....................................................................................... 10  
  3.2.1 Upper Bound .............................................................................................. 10  
  3.2.2 Lower Bound ............................................................................................. 11  
3.3 The Case $r = 3$ .............................................................................................. 12  
3.4 The Case $r > 3$ .............................................................................................. 13
4 Words Avoiding Anti-Powers

4.1 Classifying All Words Avoiding 3-Anti-Powers .......................... 15
4.2 The Characteristic Sequence of Powers of 4 ............................... 20
4.3 The Cantor Word .............................................................. 22
4.4 Remarks ................................................................. 24

5 Abelian Powers and Abelian Anti-Powers .............................. 26

5.1 The Proof of Theorem 1.4 ................................................ 26

6 Open Problems .................................................................... 28

6.1 Computing $N(k, r)$ ........................................................... 28
6.2 Classifying Words with No Anti-Powers ................................. 29
6.3 Computing $A(k, r)$ ........................................................... 29

References ................................................................. 30

APPENDICES ................................................................. 32

A Table of Values of $N(k, r)$ ................................................. 33

B C++ Program to Compute $N(k, r)$ ................................. 34
List of Tables

A.1 Values of $N(k,r)$ .................................................. 33
List of Figures

3.1 The Proof of Theorem 1.2 .................................................. 10
3.2 The Proof of Eq. (3.3) ....................................................... 11
3.3 The Proof of Lemma 3.2 .................................................... 14

4.1 Automaton Generating $c_4$ ................................................ 20
4.2 Automaton Generating $s$ .................................................. 22
Chapter 1

Introduction

1.1 Ramsey Theory and Combinatorics on Words

Ramsey theory may be considered as the branch of combinatorics that studies unavoidable regularities in large combinatorial objects. A classical example is the unavoidability of a monochromatic triangle when the edges of a complete graph on 6 vertices are coloured using two colours. In the early 70s Erdős, Simonovits and Sós initiated the study of anti-Ramsey theory, which is the study of regularities concerning all-distinct objects [6]. A much-studied regularity in the context of combinatorics on words is a power.

Definition 1.1. A $k$-power is a word of the form $w^k$ for some non-empty word $w$.

For example, murmur is a 2-power over the English alphabet.

To contrast the notion of a power, Fici et al. [6] recently introduced the notion of an anti-power.

Definition 1.2. An $r$-anti-power is a word of the form $w_1 \cdots w_r$, where the $w_i$ are words such that $|w_i| = |w_j|$ and $w_i \neq w_j$ for every pair $(i, j)$ with $i \neq j$.

For example, mormon is a 2-anti-power over the English alphabet.

With these two definitions at hand one can talk about anti-Ramsey theory in the context of words. To set the ground, let us recall a version of Ramsey’s celebrated theorem. Here $K_n$ denotes a complete graph on $n$ vertices.
Theorem 1.1 (Ramsey [12], 1930). Given integers \( k > 1 \) and \( r > 1 \) there exists an integer \( R = R(k, r) \) such that every red-blue edge-colouring of a \( K_R \) contains a red \( K_k \) or a blue \( K_r \).

Examples of these numbers (now known as Ramsey numbers) are \( R(k, 2) = k \) and \( R(3, 3) = 6 \). These are not hard to verify.

Fici et al. [6] proved an analogous result for powers and anti-powers.

Theorem 1.2 (Fici et al. [6], 2018). Given integers \( k > 1 \) and \( r > 1 \) there exists an integer \( N = N(k, r) \) such that every binary word of length \( N \) contains a \( k \)-power or an \( r \)-anti-power.

Analogous examples of these numbers are \( N(k, 2) = k \) and \( N(3, 3) = 9 \). These are again not hard to verify.

Fici et al. [6] also showed for \( k > 2 \) that

\[
k^2 - 1 \leq N(k, k) \leq k^3 \binom{k}{2}.
\]

In a recent preprint Burcroff [3] improved the above bounds to

\[
2k^2 - 2k \leq N(k, k) \leq (k^3 - k^2 + k) \binom{k}{2}
\]

for \( k > 3 \).

It seems that almost nothing else is known about the numbers \( N(k, r) \) apart from a few values [6, 13]. This is perhaps not surprising, since numbers produced by Ramsey-type results generally tend to be difficult to compute. For instance, very few Ramsey numbers are known to this day [11]. Nevertheless, we computed a list of values of \( N(k, r) \) (see Appendix A) using a C++ program (see Appendix B). The following patterns were observed by J. Shallit by means of a similar computation.

Conjecture 1.1 (Shallit, unpublished). The following relations hold.

1. \( N(k, 3) = 2k \) for \( k \geq 7 \).
2. \( N(k, 4) = 4k \) for \( k \geq 11 \).
3. \( N(k, 5) = 6k + 4 \) for \( k \geq 10 \).
In general, for fixed \( r > 2 \), \( N(k, r) = (2r - 4)k + O(1) \).

In this thesis we show that Part 1 of Conjecture 1.1 is true and that Part 3 is false, while Part 2 remains unresolved. That Part 3 is false follows from \( N(15, 5) = 95 \) and \( N(25, 5) = 155 \) with corresponding examples 04(01)1402(01)1402 and 04(01)2402(01)2402 of longest binary words avoiding \( k \)-powers and \( r \)-anti-powers. See Appendices A and B for details.

More specifically, we prove the following theorem in Chapter 3.

**Theorem 1.3.** The following relations hold.

1. For \( r \geq 2 \),
   (a) \( N(k, r) \leq r(kr - k + r){r \choose 2} \).
   (b) \( N(k, r) \geq (r - 1)k \) for \( k > r - 2 \).

   In particular, for fixed \( r \geq 2 \), \( N(k, r) = \Theta(k) \).

2. \( N(k, 3) = 2k \) for \( k \geq 7 \).

### 1.2 Avoiding Anti-Powers

In Chapter 4 we take a deeper look into words avoiding \( r \)-anti-powers for some small values of \( r \). In Section 4.1 we classify all finite and infinite binary words avoiding 3-anti-powers. Such a classification seems difficult for \( r \geq 4 \). Nevertheless, one can give interesting examples of infinite binary words avoiding \( r \)-anti-powers for specific values of \( r \). For instance, the characteristic sequence of the powers of 4 is

\[ c_4 = 0100100000000000100 \cdots. \]

That is, \( c_4[n] = 1 \) if \( n \) is a power of 4, and \( c_4[n] = 0 \) otherwise. We show in Section 4.2 that \( c_4 \) does not contain 4-anti-powers using the automatic theorem-proving software Walnut [10].

In a follow-up paper Fici et al. [5] showed that the Cantor word (also known as the Sierpiński word) does not contain 11-anti-powers. The Cantor word \( s \) is the limit as \( n \to \infty \) of the sequence \((s_n)_{n \geq 0}\) of words defined by \( s_0 = 0 \) and \( s_{n+1} = s_n1^{3^n}s_n \). So \( s = 0101^30101^90101^30101^270 \cdots \).

J. Shallit observed empirically that 11 can be improved to 10. We show using Walnut that this is indeed the case in Section 4.3.
1.3 Abelian Anti-Powers

The last part of this thesis briefly concerns abelian anti-powers. Fici, Postic and Silva [5] extended the notion of anti-powers to the abelian setting as follows. Let $P(w)$ denote the Parikh vector of the word $w$. (See Section 2.3 for details.)

**Definition 1.3.** An **abelian k-power** is a word of the form $w_1 \cdots w_k$, where the $w_i$ are words such that $|w_1| = \cdots = |w_k|$ and $P(w_1) = \cdots = P(w_k)$.

**Definition 1.4.** An **abelian r-anti-power** is a word of the form $w_1 \cdots w_r$, where the $w_i$ are words such that $|w_i| = |w_j|$ and $P(w_i) \neq P(w_j)$ for every pair $(i, j)$ with $i \neq j$.

Let $A = A(k, r)$ denote the least positive integer such that every binary word of length $A$ contains an abelian $k$-power or an abelian $r$-anti-power. It is not known whether $A(k, r)$ is finite or even exists [5]. Assuming existence, since any word avoiding abelian $k$-powers and abelian $r$-anti-powers must also avoid $k$-powers and $r$-anti-powers, one obtains the trivial lower bound $A(k, r) \geq N(k, r)$, whence $A(k, r) \geq (r - 1)k$ for $k > r - 2$ by Theorem 1.3.

In fact, computation suggests that $A(k, 3) = k^2$. We show in Chapter 5 that this is indeed a lower bound.

**Theorem 1.4.** $A(k, 3) \geq k^2$ for $k \geq 1$, assuming that $A(k, 3)$ exists.
Chapter 2

Preliminaries

2.1 Notions and Notations

A \textit{semigroup} is a set $S$ equipped with a binary operation, expressed here as concatenation, satisfying the following two properties.

- $a, b \in S \implies ab \in S$.
- $a, b, c \in S \implies abc = a(bc) = (ab)c$.

If, in addition, $S$ contains an element $e$ such that $ea = ae = a$ for all $a \in S$, then $S$ is called a \textit{monoid} with \textit{identity} $e$. Any subset of $S$ that is also a semigroup is called a \textit{subsemigroup} of $S$.

Given a set $\Sigma$ we can construct a semigroup $\Sigma^*$ as follows. For a non-negative integer $n$ and elements $a_1, \ldots, a_n \in \Sigma$, let $w = a_1 \cdots a_n \in \Sigma^*$.

- We call $\Sigma$ the \textit{alphabet} and $w$ a \textit{word over} $\Sigma$.
- We write $w[i] = a_i$ and call $w[i]$ a \textit{letter} of $w$.
- We write $w[i..j] = a_i \cdots a_j$ for $1 \leq i \leq j \leq n$ and call $w[i..j]$ a \textit{subword} (or \textit{factor} or \textit{substring}) of $w$.
- If $v$ is a subword of $w$, we say $w$ \textit{contains} $v$.
• If $\Sigma = \{0, 1\}$, we call $w$ a binary word.

• If $n = 0$, we write $w = \epsilon$ and call $\epsilon$ the empty word. Observe that $\Sigma^*$ is a monoid with identity $\epsilon$.

• The length of $w$, denoted $|w|$, is $n$.

• For $a \in \Sigma$ we denote by $|w|_a$ the size of the set $\{i : a_i = a\}$, i.e., the number of occurrences of the letter $a$ in $w$. Observe that $\sum_{a \in \Sigma} |w|_a$.

• The set of all non-empty words over $\Sigma$ is denoted $\Sigma^+$. That is, $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$. Observe that $\Sigma^+$ is a subsemigroup of $\Sigma^*$.

• A word $u \in \Sigma^*$ is a prefix (resp. suffix) of $w$ if $w = uv$ (resp. $w = vu$) for some $v \in \Sigma^*$. Observe that $\epsilon$ is a prefix and a suffix of $w$.

• A word is a border of $w$ if it is both a prefix and a suffix of $w$.

Likewise, we can construct the set $\Sigma^\omega$ of all (right-)infinite words on $\Sigma$ by letting $a_0a_1 \cdots \in \Sigma^\omega$ for any infinite sequence of elements $a_0, a_1, \ldots \in \Sigma$. The relevant definitions from the above list also apply to $\Sigma^\omega$. In addition, we write $w^\omega$ for the infinite word $ww \cdots$.

### 2.2 A Classical Result on Words with Borders

The following result may be viewed as a division algorithm for words.

**Theorem 2.1** (Lyndon and Schützenberger [9], 1962). Let $x, y, z \in \Sigma^+$. Then $xy = yz$ if and only if there exist $u \in \Sigma^+, v \in \Sigma^*$ and an integer $t \geq 0$ such that $x = uv$, $z = vu$ and $y = (uv)^tu = u(vu)^t$.

**Proof.** The non-trivial direction is only if:

• If $|x| > |y|$, then $y$ is a prefix of $x$ and a suffix of $z$. Writing $x = yv$ and $z = wy$ for some $v, w \in \Sigma^*$ gives $xy = yvy$ and $yz = ywy$. Then $xy = yz$ gives $w = v$. Taking $u = y$ gives $x = uv$ and $z = vu$ for $u \in \Sigma^+$ and $v \in \Sigma^*$. 


• If \(|x| \leq |y|\), then \(x\) is a prefix of \(y\), so we may write \(y = xw\) for some \(w \in \Sigma^*\). Then \(xxw = xwz\), i.e., \(xw = wz\), which is equivalent to the original equation, but with \(|w| = |y| - |x|\). Repeating this process finitely many times we can therefore write \(y = x^t u\) and \(xu = uz\) for some integer \(t > 0\) and word \(u \in \Sigma^*\) with \(|u| < |x|\). If \(u = \epsilon\), then \(x = z\) and \(y = x^t\). Otherwise, by the previous case, \(x = uv\) and \(z = vu\) for \(u \in \Sigma^+\) and \(v \in \Sigma^*\).

Thus \(x = uv\), \(z = vu\) and \(y = (uv)^t u = u(vu)^t\) for some words \(u \in \Sigma^+, v \in \Sigma^*\) and integer \(t \geq 0\), as desired.

Corollary 2.1. Let \(x, y \in \Sigma^+\). Then \(xy = yx\) if and only if there exist \(z \in \Sigma^+\) and positive integers \(k, \ell\) such that \(x = z^k\) and \(y = z^\ell\).

Proof. We proceed by induction on \(|xy|\). If \(|xy| = 2\), then \(x, y \in \Sigma\). Then \(xy = yx\) if and only if \(x = y\), as desired.

Assume now that \(|xy| > 2\). By Theorem 2.1, there exist \(u \in \Sigma^+, v \in \Sigma^*\) and an integer \(t \geq 0\) such that \(x = uv = vu\) and \(y = (uv)^t u = u(vu)^t\). If \(v = \epsilon\) then we are done. Otherwise, since \(|uv| = |x| < |xy|\), there exist \(z \in \Sigma^+\) and positive integers \(k, \ell\) such that \(u = z^k\) and \(v = z^\ell\) by the inductive hypothesis. Then \(x = z^{k+\ell}\) and \(y = z^{\ell(k+\ell)+k}\), as desired.

2.3 Parikh Vectors

Sometimes we may want to impose an order on the alphabet \(\Sigma\). (For us this will always be the natural order on \(\Sigma\).) In such cases we call \(\Sigma\) an ordered alphabet.

Definition 2.1. For an ordered alphabet \(\Sigma = \{a_1, \ldots, a_n\}\), the Parikh vector of \(w\) is

\[P(w) = (|w|_{a_1}, \ldots, |w|_{a_n}).\]

2.4 Previous Work on Anti-Powers

Since the conception of the notion there has been a surge of activities regarding anti-powers in words. Other than those already mentioned in the introduction, Defant [4] and Gaetz [7] studied anti-power prefixes and subwords of the Thue-Morse word.
An infinite word $w$ is *aperiodic* if it is not eventually periodic, and it is *recurrent* if every finite factor of $w$ occurs infinitely often in $w$. Fici et al. [6] asked for the maximum $k$ such that every aperiodic recurrent word must contain a $k$-anti-power, and they proved that this maximum must be 3, 4 or 5. Berger and Defant [2] resolved this question by demonstrating that the maximum is 5.

Badkobeh et al. [1] and Kociumaka et al. [8] studied algorithms for computing anti-powers in words. Badkobeh et al. gave the first algorithm to find all $k$-anti-powers in a word of length $n$, which runs in $O(n^2/k)$ time and $O(n)$ space. Following this, Kociumaka et al. gave an algorithm that computes the number $C$ of $k$-anti-power factors of a word of length $n$ in $O(nk \log k)$ time and reports all of them in $O(nk \log k + C)$ time. They also gave the construction in $O(n^2/r)$ time of a data structure of size $O(n^2/r)$, for any $r \in \{1, \ldots, n\}$, which answers anti-power queries in $O(r)$ time.
Chapter 3

A Study in $N(k,r)$

3.1 Existence

In this section we give a proof of Theorem 1.2 based on ideas by Fici et al. [6]. The argument is independent of the underlying alphabet. We shall use the following lemma of Fici et al. [6] for which we give a new proof.

**Lemma 3.1.** Let $v$ be a border of a word $w$, and let $w = uv$. If $n$ is an integer such that $|w| \geq n|u|$, then $u^n$ is a prefix of $w$.

**Proof.** Write $w = uv = vu'$. By Theorem 2.1, $u = u_1v_1$ and $v = (u_1v_1)^t u_1$ for some $u_1 \in \Sigma^+$, $v_1 \in \Sigma^*$ and integer $t \geq 0$. Thus $w = (u_1v_1)^{t+1} u_1 = u^{t+1} u_1$ and the result follows.

Let $x$ be a sufficiently long word avoiding $r$-anti-powers, whose length will be specified in Section 3.2.1. Let

$$M = (r - 1) \binom{r}{2}, \quad m = (k + 1)M. \quad (3.1)$$

Consider $U_{j,\ell} = x[j\ell + 1..(j + 1)\ell]$ for $0 \leq j \leq r - 1$. Observe that $U_{j,\ell}$ is a block of size $\ell$. Since $U_{0,\ell} \cdots U_{r-1,\ell}$ is not an $r$-anti-power, there exist $i$ and $j$ with $0 \leq i < j \leq r - 1$ such that $U_{i,\ell} = U_{j,\ell}$. Consider the pairs $(i,j)$ associated with $\ell$ for $m \leq \ell \leq m + \binom{r}{2}$. By the pigeonhole principle, two of the pairs must coincide. Hence there exist $i, j, \ell_1, \ell_2$ with $m \leq \ell_1 < \ell_2 \leq m + \binom{r}{2}$ and $0 \leq i < j \leq r - 1$ such that $U_{i,\ell_1} = U_{j,\ell_1}$ and $U_{i,\ell_2} = U_{j,\ell_2}$. 

9
Using Eq. (3.1) we therefore obtain $(i + 1)\ell_1 > i\ell_2 + 1$ and $(j + 1)\ell_1 > j\ell_2 + 1$. Let $w = x[i\ell_2 + 1..(i + 1)\ell_1]$ and $v = x[j\ell_2 + 1..(j + 1)\ell_1]$. Observe that

$$|v| = (j + 1)\ell_1 - j\ell_2 < (i + 1)\ell_1 - i\ell_2 = |w|.$$

Since $v$ is a prefix of $U_{j,\ell_2} = U_{i,\ell_2}$ and a suffix of $U_{j,\ell_1} = U_{i,\ell_1}$, it follows that $v$ is a border of $w$. Writing $w = uv$ we have

$$1 \leq |u| = |w| - |v| = \ell_1 - i(\ell_2 - \ell_1) - \ell_1 + j(\ell_2 - \ell_1) = (j - i)(\ell_2 - \ell_1) \leq (r - 1)\binom{r}{2} = M$$

so that

$$|w| > |v| = \ell_1 - j(\ell_2 - \ell_1) \geq m - (r - 1)\binom{r}{2} = m - M = kM \geq k|u|.$$

Thus, by Lemma 3.1, $u^k$ is a prefix of $w$, i.e., a factor of $x$, as desired.

### 3.2 Asymptotic Behaviour

In this section we prove the first part of Theorem 1.3.

#### 3.2.1 Upper Bound

The argument given in Section 3.1 works when

$$|x| \geq r \left( m + \binom{r}{2} \right) = r \left( (k + 1)(r - 1)\binom{r}{2} + \binom{r}{2} \right) = r(kr - k + r)\binom{r}{2}.$$
Therefore

\[ N(k, r) \leq r(kr - k + r) \binom{r}{2}. \] (3.2)

### 3.2.2 Lower Bound

Here we show that

\[ N(k, r) \geq (r - 1)k \] (3.3)

for \( k > r - 2 \geq 0 \).

We use the greedy algorithm to construct the word \( v = (0^{k-1}1)^{r-2}0^{k-1} \). Observe that

\[ |v| = (r - 1)k - 1. \]

We claim that \( v \) contains neither a \( k \)-power nor an \( r \)-anti-power. This will give the desired bound.

We proceed by contradiction.

- If \( v \) contains an \( r \)-anti-power \( u_1 \cdots u_r \), then at most \( r - 2 \) of \( u_1, \ldots, u_r \) can contain a 1. But then at least two of \( u_1, \ldots, u_r \) must be equal, a contradiction.

- If \( v \) contains a \( k \)-power \( w^k \), then \( w \) cannot contain a 1 since the number of 1s in \( v \) is \( r - 2 < k \). Thus \( w^k \) must consist entirely of 0s. But there is no block of \( k \) consecutive 0s in \( v \).

This completes the proof.
3.3 The Case $r = 3$

In this section we prove the second part of Theorem 1.3, namely

$$N(k, 3) = 2k$$

for $k \geq 7$.

Using Eq. (3.3) it suffices to show that any word of length $2k$ contains a $k$-power or a $3$-anti-power. We proceed by induction on $k$. For the base case we need to show that any binary word of length 14 contains a 7-power or a 3-anti-power. This follows from Table A.1 and may be verified by brute force, possibly using a computer search. So assume that the result holds for some $k \geq 7$.

Consider a binary word $y = xab$ of length $2k + 2$ for $a, b \in \Sigma = \{0, 1\}$. Without loss of generally, $y$ begins with a 0. By the inductive hypothesis, $x$ contains either a 3-anti-power—in which case we are done—or $w^k$, where $w \in \{0, 1, 00, 01\}$. We assume the latter.

If $w = 00$, then $y$ contains $0^{k+1}$ so we are done.

If $w = 01$ then $y = (01)^{k-4}0(101)(010)(1ab)$. If $y$ does not contain a 3-anti-power, then we must have $1ab = 101$. But then $y = (01)^{k+1}$ which contains a $(k + 1)$-power.

Otherwise $x = uw^kv$ for $u, v \in \Sigma^*$ and $w = c \in \Sigma$, so $y = uc^kvb$. Note that if $u$ ends in a $c$ or $vab$ begins with a $c$ then $y$ contains $c^{k+1}$ and we are done. So we may assume otherwise.

Case 1: $u = \epsilon$. Assume that $v = \bar{c}v'$. Then $y = c^k\bar{c}v'ab = c^2y'$, where $y' = c^{k-2}\bar{c}v'ab$. By the inductive hypothesis, $y'$ contains either a 3-anti-power—in which case we are done—or a $k$-power. Then $\bar{c}v' = \bar{c}^k$ or $v'a = a^k$ or $v'ab = da^k$ for some $d \in \Sigma$. Then $y = c^k\bar{c}^kab$ or $c^k\bar{c}a^kb$ or $c^k\bar{c}da^k$. If $a = \bar{c}$ in the first two cases, or $a = d$ in the last case, then $y$ contains $a^{k+1}$ and we are done. So we may assume that $y = c^k\bar{c}^cb$ or $c^k\bar{c}c^k\bar{c}b$ or $c^k\bar{c}^2c^k$ or $c^k\bar{c}\bar{c}c^k$.

- If $y = c^k\bar{c}^cb$ then $y$ contains the 3-anti-power $c^j\bar{c}^cb$, where $j \in \{1, 2, 3\}$ such that $j + k + 2 \equiv 0 \pmod{3}$.
- If $y = c^k\bar{c}c^k$ then either $b = c$ or $b = \bar{c}$. If $b = c$ then $y$ contains $c^{k+1}$. Otherwise $y$ contains the 3-anti-power $c^j\bar{c}c^kB$, where $j \in \{0, 1, 2\}$ such that $j + k + 2 \equiv 0 \pmod{3}$.
- If $y = c^k\bar{c}c^k$ then $y$ contains the 3-anti-power $c\bar{c}c\bar{c}c\bar{c}$.
- If $y = c^k\bar{c}\bar{c}c^k$ then $y$ contains the 3-anti-power $cc\bar{c}c\bar{c}$.
Case 2: \( u \neq \epsilon \). Then \( u = u'c \) and \( vab = cv' \), so \( y = u'cv^kcv' \). Consider a suffix \( u'' \) of \( u'c \) and a prefix \( v'' \) of \( v' \) such that \( \ell = |u''v''| \in \{0,1,2\} \) and \( k + \ell + 2 \equiv 0 \mod 3 \). Then \( y \) contains the 3-anti-power \( u''cv^kcv'' \).

This completes the proof.

3.4 The Case \( r > 3 \)

As per the proof in Section 3.3 one might expect that a similar argument be carried out for any \( r \geq 3 \). However, the number of cases to deal with grows rapidly with \( r \). As a result, this method soon becomes impractical. Nevertheless, one could try to deal with the cases by other means. For instance, with \( w \) as in Section 3.3 the following lemma shows that it suffices to consider only \(|w| < r\).

Lemma 3.2. Let \( w \) be a non-empty binary word of length \( \ell \geq 2 \), and let \( k > \ell \). Then \( w^k \) contains a \( 2(k-1) \)-power or an \( \ell \)-anti-power.

Proof. Let \( w = w[1..\ell] \) and \( v_i = w[i..\ell]w[1..i] \) for \( i = 1, \ldots, \ell \). If \( v_i = v_j \) for some \( 1 \leq i < j \leq \ell \), then \( xy = yx \), where \( x = w[i..j-1] \) and \( y = w[j..\ell]w[1..i-1] \). Hence, there exist a non-empty binary word \( z \) and integers \( p, q > 0 \) such that \( x = z^p \) and \( y = z^q \) by the corollary to Theorem 2.1. Then

\[
\begin{align*}
    w^k &= w[1..i-1](w[i..\ell]w[1..i-1])^kw[i..\ell] \\
    &= w[1..i-1](xy)^{k-1}w[i..\ell] \\
    &= w[1..i-1]z^{(p+q)(k-1)}w[i..\ell],
\end{align*}
\]

which contains \( z^{2(k-1)} \).

If the \( v_i \) are all distinct, then \( w^k \) contains \( w^{\ell+1} = v_1 \cdots v_\ell \), which is an \( \ell \)-anti-power. This completes the proof. \( \Box \)
Figure 3.3: The Proof of Lemma 3.2
Chapter 4

Words Avoiding Anti-Powers

Throughout this chapter $a$ will denote an arbitrary element in $\Sigma = \{0, 1\}$. The binary complement of $a$ is denoted $\overline{a}$, so $\overline{a} = 1 - a$.

As mentioned in the introduction, it is not difficult to see that $N(k, 2) = k$, since the only binary words avoiding 2-anti-powers are of the form $a^i$. From this observation it also follows that the only infinite binary words avoiding 2-anti-powers are of the form $a^\omega$. So we consider $r \geq 3$ below.

4.1 Classifying All Words Avoiding 3-Anti-Powers

Using arguments similar to those in Section 3.3 we can prove the following result, which was first observed by J. Shallit.

**Theorem 4.1.** Let $n \geq 12$ be an integer such that $n \equiv 0 \pmod{3}$. Then there are exactly $2^n + 12$ binary words of length $n$ avoiding 3-anti-powers, given by the following list.

1. $a^n$
2. $a^i\overline{a}a^{n-1-i}$ for $1 \leq i < n$
3. $a^{n-2}\overline{a}^2$
4. $a^{n-3}\overline{a}a\overline{a}$
5. $a^2\overline{a}^{n-2}$
6. \((a\bar{a})^{n/2}\) if \(n\) is even, \((a\bar{a})^{(n-1)/2}a\) if \(n\) is odd

7. \(a\bar{a}a^{n-3}\)

8. \(a\bar{a}^{n-1}\)

Proof. We proceed by induction on \(n\). The base case \(n = 12\) may be verified by brute force, so assume that the result holds for some \(n \geq 12\) with \(n \equiv 0 \pmod{3}\).

Consider a binary word \(wu\) of length \(n + 3\), where \(w\) avoids 3-anti-powers and \(|w| = n\). Then

\[
u \in \{a^3, a^2\bar{a}, a\bar{a}a, a\bar{a}^2, \bar{a}a\bar{a}, a^2a, \bar{a}^3\}.
\]

By the inductive hypothesis, \(w\) belongs to the list in the statement of the theorem. We now observe the following.

- If \(w = a^n\), then \(wu\) does not contain a 3-anti-power if and only if \(u \notin \{\bar{a}^2a, \bar{a}^3\}\).
- If \(w = a^i\bar{a}a^{n-1-i}\) with \(1 \leq i < n\), then \(wu\) does not contain a 3-anti-power if and only if \(u = a^3\).
- If \(w = a^{n-2}\bar{a}^2\), then \(wu\) contains a 3-anti-power for every choice of \(u\).
- If \(w = a^{n-3}a\bar{a}\), then \(wu\) contains a 3-anti-power for every choice of \(u\).
- If \(w = a^2\bar{a}^{n-2}\), then \(wu\) does not contain a 3-anti-power if and only if \(u = \bar{a}^3\).
- If \(n\) is even and \(w = (a\bar{a})^{n/2}\), then \(wu\) does not contain a 3-anti-power if and only if \(u = a\bar{a}\).
- If \(n\) is odd and \(w = (a\bar{a})^{(n-1)/2}a\), then \(wu\) does not contain a 3-anti-power if and only if \(u = \bar{a}\bar{a}\).
- If \(w = a\bar{a}a^{n-3}\), then \(wu\) does not contain a 3-anti-power if and only if \(u = \bar{a}^3\).
- If \(w = a\bar{a}^{n-1}\), then \(wu\) does not contain a 3-anti-power if and only if \(u = \bar{a}^3\).

In every case, \(wu\) belongs to the list in question. Therefore we are done.

Consequently, we can obtain similar lists for \(n \equiv 1 \pmod{3}\) and \(n \equiv 2 \pmod{3}\).
Theorem 4.2. Let $n \geq 12$ be an integer such that $n \equiv 1 \pmod{3}$. Then there are exactly $2n + 14$ binary words of length $n$ avoiding 3-anti-powers, given by the following list.

1. $a^n$
2. $a^i\bar{a}a^{n-1-i}$ for $1 \leq i < n$
3. $a^{n-2}\bar{a}^2$
4. $a^{n-3}\bar{a}\bar{a}$
5. $a^2\bar{a}^{n-2}$
6. $(\bar{a}a)^{n/2}$ if $n$ is even, $(\bar{a}a)^{(n-1)/2}\bar{a}$ if $n$ is odd
7. $\bar{a}\bar{a}a^n-3$
8. $a^n-2\bar{a}$
9. $a^n-1$

Proof. Such a word must be of the form $wa$ or $\bar{w}a$, where $w$ is a word of length $n - 1$ given by Theorem 4.1. So it must belong to the following list.

<table>
<thead>
<tr>
<th>$wa$</th>
<th>$\bar{w}a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a^n$</td>
<td>$a^{n-1}\bar{a}$</td>
</tr>
<tr>
<td>$a^i\bar{a}a^{n-1-i}$</td>
<td>$a^i\bar{a}a^{n-2-i}\bar{a}$</td>
</tr>
<tr>
<td>$1 \leq i &lt; n - 1$</td>
<td>$1 \leq i &lt; n - 1$</td>
</tr>
<tr>
<td>$a^{n-3}\bar{a}^2a$</td>
<td>$a^{n-3}\bar{a}^3$</td>
</tr>
<tr>
<td>$a^{n-4}\bar{a}a^n$</td>
<td>$a^{n-4}\bar{a}\bar{a}a^n$</td>
</tr>
<tr>
<td>$a^2\bar{a}^{n-3}a$</td>
<td>$a^{22}\bar{a}^{n-2}$</td>
</tr>
<tr>
<td>$(\bar{a}a)^{(n-1)/2}a$</td>
<td>$(\bar{a}a)^{22}(n - 1/2)\bar{a}$</td>
</tr>
<tr>
<td>$(n$ odd), $(\bar{a}a)^{n/2}\bar{a}^2$</td>
<td>$(n$ odd), $(\bar{a}a)^{n/2}$</td>
</tr>
<tr>
<td>$(n$ even)</td>
<td>$(n$ even)</td>
</tr>
<tr>
<td>$\bar{a}\bar{a}\bar{a}a^{n-4}a$</td>
<td>$\bar{a}\bar{a}a^n-3$</td>
</tr>
<tr>
<td>$\bar{a}a^{n-2}a$</td>
<td>$\bar{a}a^{n-1}$</td>
</tr>
</tbody>
</table>

- If $n = 3j + 1$, then $a^i\bar{a}a^{n-2-i}\bar{a} = a^i\bar{a}a^{3j-1-i}\bar{a}$, which contains the 3-anti-power $a^{i-1}\bar{a}a^{3j-1-i}\bar{a}$, for $1 \leq i < n - 3$.
- $a^{n-3}\bar{a}^2a$ contains the 3-anti-power $(aa)(\bar{a}a)(\bar{a}a)$.
- $a^{n-3}\bar{a}^3$ contains the 3-anti-power $(aa)(\bar{a}a)(\bar{a}a)$.
\begin{itemize}
\item $a^{n-4}a\bar{a}a$ contains the 3-anti-power $(aaa)(\bar{aa})(\bar{aa})$.
\item $a^{n-4}a\bar{a}a\bar{a}$ contains the 3-anti-power $(aa)(\bar{aa})(\bar{aa})$.
\item $a^2a^{n-3}a$ contains the 3-anti-power $aa^{n-3}a$.
\item $(\bar{a}a)^{n/2}a^2$ (n even) contains the 3-anti-power $(\bar{aa})(a\bar{aa})(\bar{aa})$.
\item $(\bar{a}a)^{(n-1)/2}a$ (n odd) contains the 3-anti-power $(a\bar{aa})(\bar{aa})(\bar{aa})$.
\item $a\bar{aa}a^{n-4}a$ contains the 3-anti-power $(\bar{aa}(n-1)/3)(\bar{a}^{(n-1)/3})(\bar{a}^{(n-4)/3}a)$.
\end{itemize}

The rest of the possibilities can be easily seen to avoid 3-anti-powers. This concludes the proof.

\begin{theorem}
Let $n \geq 12$ be an integer such that $n \equiv 2 \pmod{3}$. Then there are exactly $2n + 22$ binary words of length $n$ avoiding 3-anti-powers, given by the following list.

1. $a^n$
2. $a^i\bar{a}a^{n-1-i}$ for $1 \leq i < n$
3. $a^{n-2}\bar{a}^2$
4. $a^{n-3}\bar{a}a$
5. $a^2\bar{a}^{n-3}a$
6. $a^2\bar{a}^{n-2}$
7. $a\bar{a}a^{n-3}\bar{a}$
8. $(\bar{a}a)^{n/2}$ if $n$ is even, $(\bar{a}a)^{(n-1)/2}a$ if $n$ is odd
9. $a\bar{aa}a^{n-3}$
10. $a\bar{a}^{n-3}a^2$
11. $a\bar{a}^{n-3}a\bar{a}$
12. $a\bar{a}^{n-2}a$
13. $a\bar{a}^{n-1}$
\end{theorem}
Proof. Such a word must be of the form \(wa\) or \(w\tilde{a}\), where \(w\) is a word of length \(n - 1\) given by Theorem 4.2. So it must belong to the following list.

<table>
<thead>
<tr>
<th>(wa)</th>
<th>(w\tilde{a})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a^n)</td>
<td>(a^{n-1}\tilde{a})</td>
</tr>
<tr>
<td>(a^i\tilde{a}a^{n-1-i}) ((1 \leq i &lt; n - 1))</td>
<td>(a^i\tilde{a}a^{n-2-i}\tilde{a} ((1 \leq i &lt; n - 1))</td>
</tr>
<tr>
<td>(a^{n-3}a^2\tilde{a})</td>
<td>(a^{n-3}\tilde{a}^3)</td>
</tr>
<tr>
<td>(a^{n-4}\tilde{a}a\tilde{a})</td>
<td>(a^{n-4}\tilde{a}a\tilde{a})</td>
</tr>
<tr>
<td>(a^2\tilde{a}a^{n-3}\tilde{a})</td>
<td>(a^2\tilde{a}a^{n-2})</td>
</tr>
<tr>
<td>((\tilde{a}a)^{(n-1)/2}a) ((n odd)), ((a\tilde{a})^{n/2}a^2) ((n even))</td>
<td>((\tilde{a}a)^{(n-1)/2}\tilde{a} ((n odd)), ((a\tilde{a})^{n/2}) ((n even))</td>
</tr>
<tr>
<td>(a\tilde{a}a^{n-4}\tilde{a}\tilde{a})</td>
<td>(a\tilde{a}a^{n-3}\tilde{a}\tilde{a})</td>
</tr>
<tr>
<td>(a\tilde{a}a^{n-3}a^2\tilde{a}\tilde{a})</td>
<td>(a\tilde{a}a^{n-3}a\tilde{a}\tilde{a})</td>
</tr>
<tr>
<td>(a\tilde{a}a^{n-2}\tilde{a}\tilde{a})</td>
<td>(a\tilde{a}^{n-1})</td>
</tr>
</tbody>
</table>

- If \(n = 3j + 2\), then \(a^i\tilde{a}a^{n-2-i}\tilde{a} = a^i\tilde{a}a^{3j-i}\tilde{a}\), which contains the 3-anti-power \(a^{i-2}\tilde{a}a^{3j-i}\tilde{a}\), for \(2 \leq i < n - 3\).
- \(a^{n-3}a^2\tilde{a}\) contains the 3-anti-power \((aa)(a\tilde{a})(\tilde{a}a)\).
- \(a^{n-3}\tilde{a}^3\) contains the 3-anti-power \((aa)(a\tilde{a})(\tilde{a}a)\).
- \(a^{n-4}\tilde{a}a\tilde{a}a\) contains the 3-anti-power \((aaa)(aa\tilde{a})(a\tilde{a}a)\).
- \(a^{n-4}\tilde{a}a\tilde{a}a\) contains the 3-anti-power \((aa)(a\tilde{a})(\tilde{a}a)\).
- \((\tilde{a}a)^{n/2}a^2\) \((n even)\) contains the 3-anti-power \((\tilde{a}a\tilde{a})(\tilde{a}a\tilde{a})(\tilde{a}a\tilde{a})\).
- \((a\tilde{a})^{(n-1)/2}\tilde{a}\) \((n odd)\) contains the 3-anti-power \((a\tilde{a}\tilde{a})(\tilde{a}a\tilde{a})(a\tilde{a}\tilde{a})\).
- \(a\tilde{a}a^{n-4}\tilde{a}a\) contains the 3-anti-power \((a\tilde{a}^{(n-5)/3})(\tilde{a}a^{(n-2)/3})(\tilde{a}a^{(n-5)/3})\).

The rest of the possibilities can be easily seen to avoid 3-anti-powers. This concludes the proof.

As an immediate corollary of these results we obtain the following classification of infinite binary words avoiding 3-anti-powers.

**Theorem 4.4.** The only infinite binary words avoiding 3-anti-powers are given by the following list.
1. $a^\omega$

2. $a^i\bar{a}a^\omega$ for $1 \leq i$

3. $a^2\bar{a}^\omega$

4. $(a\bar{a})^\omega$

5. $a\bar{a}a\bar{a}^\omega$

6. $a\bar{a}^\omega$

4.2 The Characteristic Sequence of Powers of 4

In this section we show that $c_4$ avoids 4-anti-powers.

It is not difficult to see that $c_4$ is generated by the automaton in Figure 4.1 below, which reads the base-4 representation of $n$ from left to right and produces $c_4[n]$ based on the state reached.

![Automaton Generating $c_4$](image)

Figure 4.1: Automaton Generating $c_4$

We encode this automaton in the file Walnut/Word Automata Library/POW4.txt as follows.
To check whether \( c_4 \) contains 4-anti-powers we now enter the following command in Walnut:

```plaintext
eval POW4_has_no_4_anti_power "?msd_4 Ai,n ((i>=0) & (n>=1)) => ( 
(At (t<n) => POW4[i+0*n+t] = POW4[i+1*n+t]) | 
(At (t<n) => POW4[i+0*n+t] = POW4[i+2*n+t]) | 
(At (t<n) => POW4[i+0*n+t] = POW4[i+3*n+t]) | 
(At (t<n) => POW4[i+1*n+t] = POW4[i+2*n+t]) | 
(At (t<n) => POW4[i+1*n+t] = POW4[i+3*n+t]) | 
(At (t<n) => POW4[i+2*n+t] = POW4[i+3*n+t]))": 
```

This generates the output string `true` in the following file.

`Walnut/Result/POW4_has_no_4_anti_power.txt`

Therefore \( c_4 \) does not contain 4-anti-powers, as desired.
4.3 The Cantor Word

In this section we show that $s$ avoids 10-anti-powers.

It is well-known that $s$ is generated by the automaton in Figure 4.2 below, which reads the base-3 representation of $n$ from left to right and produces $s[n]$ based on the state reached.

![Automaton Generating s](image)

Figure 4.2: Automaton Generating $s$

We encode this automaton in the file Walnut/Word Automata Library/Cantor.txt as follows.

```
msd_3
0 0
0 -> 0
1 -> 1
2 -> 0
1 1
0 -> 1
1 -> 1
2 -> 1
```

To check whether $s$ contains 10-anti-powers we now enter the following command in Walnut.

```
eval cantor_has_no_10_anti_power "?msd_3 Ai,n ((i>=0) & (n>=1)) => ( 
(At (t<n) => Cantor[i+0*n+t] = Cantor[i+1*n+t]) | 
(At (t<n) => Cantor[i+0*n+t] = Cantor[i+2*n+t]) | 
(At (t<n) => Cantor[i+0*n+t] = Cantor[i+3*n+t]) | 
(At (t<n) => Cantor[i+0*n+t] = Cantor[i+4*n+t]) | 
(At (t<n) => Cantor[i+0*n+t] = Cantor[i+5*n+t]) | 
(At (t<n) => Cantor[i+0*n+t] = Cantor[i+6*n+t]) | 
)"
```

22
(At (t<n) => Cantor[i+0*n+t] = Cantor[i+7*n+t])
(At (t<n) => Cantor[i+0*n+t] = Cantor[i+8*n+t])
(At (t<n) => Cantor[i+0*n+t] = Cantor[i+9*n+t])
(At (t<n) => Cantor[i+1*n+t] = Cantor[i+2*n+t])
(At (t<n) => Cantor[i+1*n+t] = Cantor[i+3*n+t])
(At (t<n) => Cantor[i+1*n+t] = Cantor[i+4*n+t])
(At (t<n) => Cantor[i+1*n+t] = Cantor[i+5*n+t])
(At (t<n) => Cantor[i+1*n+t] = Cantor[i+6*n+t])
(At (t<n) => Cantor[i+1*n+t] = Cantor[i+7*n+t])
(At (t<n) => Cantor[i+1*n+t] = Cantor[i+8*n+t])
(At (t<n) => Cantor[i+1*n+t] = Cantor[i+9*n+t])
(At (t<n) => Cantor[i+2*n+t] = Cantor[i+3*n+t])
(At (t<n) => Cantor[i+2*n+t] = Cantor[i+4*n+t])
(At (t<n) => Cantor[i+2*n+t] = Cantor[i+5*n+t])
(At (t<n) => Cantor[i+2*n+t] = Cantor[i+6*n+t])
(At (t<n) => Cantor[i+2*n+t] = Cantor[i+7*n+t])
(At (t<n) => Cantor[i+2*n+t] = Cantor[i+8*n+t])
(At (t<n) => Cantor[i+2*n+t] = Cantor[i+9*n+t])
(At (t<n) => Cantor[i+3*n+t] = Cantor[i+4*n+t])
(At (t<n) => Cantor[i+3*n+t] = Cantor[i+5*n+t])
(At (t<n) => Cantor[i+3*n+t] = Cantor[i+6*n+t])
(At (t<n) => Cantor[i+3*n+t] = Cantor[i+7*n+t])
(At (t<n) => Cantor[i+3*n+t] = Cantor[i+8*n+t])
(At (t<n) => Cantor[i+3*n+t] = Cantor[i+9*n+t])
(At (t<n) => Cantor[i+4*n+t] = Cantor[i+5*n+t])
(At (t<n) => Cantor[i+4*n+t] = Cantor[i+6*n+t])
(At (t<n) => Cantor[i+4*n+t] = Cantor[i+7*n+t])
(At (t<n) => Cantor[i+4*n+t] = Cantor[i+8*n+t])
(At (t<n) => Cantor[i+4*n+t] = Cantor[i+9*n+t])
(At (t<n) => Cantor[i+5*n+t] = Cantor[i+6*n+t])
(At (t<n) => Cantor[i+5*n+t] = Cantor[i+7*n+t])
(At (t<n) => Cantor[i+5*n+t] = Cantor[i+8*n+t])
(At (t<n) => Cantor[i+5*n+t] = Cantor[i+9*n+t])
(At (t<n) => Cantor[i+6*n+t] = Cantor[i+7*n+t])
(At (t<n) => Cantor[i+6*n+t] = Cantor[i+8*n+t])
(At (t<n) => Cantor[i+6*n+t] = Cantor[i+9*n+t])
(At (t<n) => Cantor[i+7*n+t] = Cantor[i+8*n+t])
(At (t<n) => Cantor[i+7*n+t] = Cantor[i+9*n+t])
\[(\text{At } (t<n) \Rightarrow \text{Cantor}[i+8\times n+t] = \text{Cantor}[i+9\times n+t])\]"

This generates the output string \texttt{true} in the following file.

\texttt{Walnut/Result/cantor\_has\_no\_10\_anti\_power.txt}

Therefore \(s\) does not contain 10-anti-powers, as desired.

### 4.4 Remarks

1. Using a similar \texttt{Walnut} program it can be shown that \(s\) does contain 9-anti-powers. So 10 is optimal.

2. J. Shallit observed that the following \texttt{Walnut} program also produced \texttt{true}.

\begin{verbatim}
 eval cantor_has_no_10_anti_power "?msd_3 Ai,n (((i>=0) & (n>=1)) => ( \\
 (At (t<n) => Cantor[i+2*n+t] = Cantor[i+8*n+t]) | \\
 (At (t<n) => Cantor[i+5*n+t] = Cantor[i+8*n+t]) | \\
 (At (t<n) => Cantor[i+3*n+t] = Cantor[i+4*n+t]) | \\
 (At (t<n) => Cantor[i+4*n+t] = Cantor[i+5*n+t]) | \\
 (At (t<n) => Cantor[i+8*n+t] = Cantor[i+9*n+t]) | \\
 (At (t<n) => Cantor[i+6*n+t] = Cantor[i+7*n+t]) | \\
 (At (t<n) => Cantor[i+3*n+t] = Cantor[i+9*n+t]) | \\
 (At (t<n) => Cantor[i+3*n+t] = Cantor[i+7*n+t]) | \\
 (At (t<n) => Cantor[i+0*n+t] = Cantor[i+5*n+t]) | \\
 (At (t<n) => Cantor[i+7*n+t] = Cantor[i+8*n+t]) | \\
 (At (t<n) => Cantor[i+5*n+t] = Cantor[i+6*n+t]) | \\
 (At (t<n) => Cantor[i+2*n+t] = Cantor[i+9*n+t]))
\)
\end{verbatim}

This means that a slightly stronger result is true: for every subword \(w_0 \cdots w_9\) of \(s\) with \(|w_0| = \cdots = |w_9|\), there is a pair \((i,j)\) given by the above list such that \(w_i = w_j\).

3. Similarly, we observed for \(c_4\) that the following \texttt{Walnut} program also produced \texttt{true}. 

24
eval POW4_has_no_4_anti_power ":?msd_4 Ai,n ((i>=0) & (n>=1)) => (  (At (t<n) => POW4[i+0*n+t] = POW4[i+1*n+t]) |  (At (t<n) => POW4[i+1*n+t] = POW4[i+2*n+t]) |  (At (t<n) => POW4[i+1*n+t] = POW4[i+3*n+t]) |  (At (t<n) => POW4[i+2*n+t] = POW4[i+3*n+t]))":  

In other words, for every subword \(w_0w_1w_2w_3\) of \(c_4\) with \(|w_0| = |w_1| = |w_2| = |w_3|\), either \(w_0 = w_1\) or \(w_1 = w_2\) or \(w_1 = w_3\) or \(w_2 = w_3\).
Chapter 5

Abelian Powers and Abelian Anti-Powers

Recall that an abelian $k$-power (resp. abelian $k$-anti-power) is a word of the form $w_1 \cdots w_k$, where the $w_i$ are words such that $|w_1| = \cdots = |w_k|$ and the Parikh vectors $P(w_1), \ldots, P(w_k)$ are equal (resp. distinct). Our goal in this chapter is to give a proof of Theorem 1.4 by showing that there is a binary word of length $k^2 - 1$ avoiding abelian $k$-powers and abelian 3-anti-powers.

5.1 The Proof of Theorem 1.4

Computation suggests that the word

$$w = (0^{k-1}1)^{k-1}0^{k-1}$$

with $|w| = k^2 - 1$ is a longest binary word avoiding abelian $k$-powers and abelian 3-anti-powers. We show below that $w$ avoids abelian $k$-powers and abelian 3-anti-powers.

If $w$ contains an abelian $k$-power $w_1 \cdots w_k$, then no $w_i$ can contain a 1 since otherwise $k - 1 = |w_i| \geq k|w_1|$, which is impossible. Hence the $w_i$s must consist entirely of 0s. But there is no block of $k$ consecutive 0s in $w$.

To see that $w$ does not contain an abelian 3-anti-power, consider any subword

$$v = w[a..a + d - 1]w[a + d..a + 2d - 1]w[a + 2d..a + 3d - 1]$$
where $1 \leq a < a + 3d < k^2$. Observe that

$$w[i] = 1 \iff i \equiv 0 \pmod{k}$$

for $1 \leq i < k^2$. We claim that two of the sets

$$A_1 = \{a, \ldots, a + d - 1\}, \quad A_2 = \{a + d, \ldots, a + 2d - 1\}, \quad A_3 = \{a + 2d, \ldots, a + 3d - 1\}$$

contain the same number of multiples of $k$. To see this, note that the number of multiples of $k$ in $A_i$ is given by

$$\Delta(i) = \left\lfloor \frac{a + id - 1}{k} \right\rfloor - \left\lfloor \frac{a + (i - 1)d - 1}{k} \right\rfloor$$

$$= \frac{d}{k} - \left\{ \frac{a + id - 1}{k} \right\} + \left\{ \frac{a + (i - 1)d - 1}{k} \right\}$$

$$\in \left( \frac{d}{k} - 1, \frac{d}{k} + 1 \right)$$

for each $i$, where $\lfloor x \rfloor$ and $\{ x \}$ respectively denote the integer and fractional parts of the real number $x$. Hence two of $\Delta(1)$, $\Delta(2)$ and $\Delta(3)$ must be equal by the pigeonhole principle. Thus $v$ cannot be an abelian 3-anti-power, as desired.
Chapter 6

Open Problems

We conclude this thesis with the following list of unresolved problems that could serve as pointers to possible future research in the area.

6.1 Computing $N(k, r)$

We believe that Part 2 of Conjecture 1.1 can be resolved using an approach similar to the one in Section 3.3, but will require a deeper case analysis. We leave it as an open problem in the hope of a more novel approach.

**Conjecture 6.1.** $N(k, 4) = 4k$ for $k \geq 11$.

We also propose the following modified version of Part 3 of Conjecture 1.1.

**Conjecture 6.2.** Let $k \geq 10$. Then

$$N(k, 5) = \begin{cases} 6k + 4, & k \not\equiv 5 \pmod{10} \\ 6k + 5, & k \equiv 5 \pmod{10}. \end{cases}$$

Furthermore, if $k \equiv 5 \pmod{10}$, then $0^4(01)^{k-1}0^2(01)^{k-1}0^2(01)^{k-1}0^2$ is the lexicographically least longest binary word avoiding $k$-powers and 5-anti-powers.

We leave the general version of Conjecture 1.1 as an open problem.

**Conjecture 6.3.** $N(k, r) = (2r - 4)k + O(1)$ for fixed $r > 2$. 

28
Lastly, it would be interesting to compute or estimate \( N(k, r) \) for other values of \( k \) and \( r \). There is a noticeable gap between the upper and lower bounds given by Theorem 1.3.

**Open Problem 6.1.** Compute new classes of values of \( N(k, r) \).

**Open Problem 6.2.** Find better estimates for \( N(k, r) \).

### 6.2 Classifying Words with No Anti-Powers

In Section 4.1 we classified all finite and infinite binary words avoiding 3-anti-powers. Such a classification seems difficult for \( r \geq 4 \). We leave it as an open problem.

**Open Problem 6.3.** Classify all finite and infinite binary words avoiding \( r \)-anti-powers for \( r \geq 4 \).

### 6.3 Computing \( A(k, r) \)

As mentioned in the introduction, it is not known whether \( A(k, r) \) exists or is finite.

**Conjecture 6.4.** \( A(k, r) \) exists and is finite for all \( k, r \geq 1 \).

Assuming existence, in Chapter 5 we showed that \( A(k, 3) \geq k^2 \). We conjecture that this is in fact an equality.

**Conjecture 6.5.** \( A(k, 3) \leq k^2 \) for \( k \geq 1 \).

In general, the following seems to hold.

**Conjecture 6.6.** \( A(k, r) = \Theta(k^{r-1}) \) for fixed \( r \geq 2 \).

It would also be interesting to compute or estimate \( A(k, r) \) for \( r > 3 \).

**Open Problem 6.4.** Compute or estimate \( A(k, r) \) for \( r > 3 \).
References


APPENDICES
Appendix A

Table of Values of $N(k, r)$

<table>
<thead>
<tr>
<th>$k$</th>
<th>$r$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>19</td>
<td>41</td>
<td>58</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>12</td>
<td>24</td>
<td>48</td>
<td>79</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>5</td>
<td>12</td>
<td>26</td>
<td>55</td>
<td>84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>6</td>
<td>12</td>
<td>31</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>7</td>
<td>14</td>
<td>33</td>
<td>58</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>8</td>
<td>16</td>
<td>33</td>
<td>58</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>9</td>
<td>18</td>
<td>36</td>
<td>62</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>10</td>
<td>20</td>
<td>42</td>
<td>64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>11</td>
<td>22</td>
<td>44</td>
<td>70</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>12</td>
<td>24</td>
<td>48</td>
<td>76</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>13</td>
<td>26</td>
<td>52</td>
<td>82</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>14</td>
<td>28</td>
<td>56</td>
<td>88</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>15</td>
<td>30</td>
<td>60</td>
<td>95</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$k$</th>
<th>$r$</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>17</td>
<td>34</td>
<td>68</td>
<td>106</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>18</td>
<td>36</td>
<td>72</td>
<td>112</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>19</td>
<td>38</td>
<td>76</td>
<td>118</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>20</td>
<td>40</td>
<td>80</td>
<td>124</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>21</td>
<td>42</td>
<td>84</td>
<td>130</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>22</td>
<td>44</td>
<td>88</td>
<td>136</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>23</td>
<td>46</td>
<td>92</td>
<td>142</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>24</td>
<td>48</td>
<td>96</td>
<td>148</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>25</td>
<td>50</td>
<td>100</td>
<td>155</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>26</td>
<td>52</td>
<td>104</td>
<td>160</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>27</td>
<td>54</td>
<td>108</td>
<td>166</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>28</td>
<td>56</td>
<td>112</td>
<td>172</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>29</td>
<td>58</td>
<td>116</td>
<td>178</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>30</td>
<td>60</td>
<td>120</td>
<td>184</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix B

C++ Program to Compute $N(k, r)$

Instructions:

- Place the following files in an empty directory.
- Run `make` in that directory.
- Now running `./main k r 0 1` from that directory for any $k$ and $r$ will output the lexicographically least longest binary word avoiding $k$-powers and $r$-anti-powers.

---

**Makefile**

```makefile
CXX = g++
CXXFLAGS = -std=c++11

main: krfree.cc
```

**krfree.h**

```c
#ifndef KRFREE_H
#define KRFREE_H

#include <string>

bool k_free_tail(std::string &w, int k);
bool r_free_tail(std::string &w, int r);

#endif /* KRFREE_H */
```

34
krfree.cc

#include "krfree.h"
#include <set>

bool k_free_tail(std::string &w, int k) {
    for (int block_size = 1; k*block_size <= w.size(); block_size++) {
        std::set<std::string> tails;
        for (int j = 1; j <= k; j++) {
            std::string subtail = w.substr(w.size()-j*block_size, block_size);
            tails.insert(subtail);
        }
        if (tails.size() == 1) { // found a k-power
            return false;
        }
    }
    return true;
}

bool r_free_tail(std::string &w, int r) {
    for (int block_size = 1; r*block_size <= w.size(); block_size++) {
        std::set<std::string> tails;
        for (int j = 1; j <= r; j++) {
            std::string subtail = w.substr(w.size()-j*block_size, block_size);
            tails.insert(subtail);
        }
        if (tails.size() == r) { // found an r-anti-power
            return false;
        }
    }
    return true;
}

main.cc

#include <iostream>
#include <set>
#include "krfree.h"
#include <ctime>
std::string max_word;

static bool generate(std::string &start, int k, int r, std::set<std::string> &alphabet) {
    if (start.size() > max_word.size()) {
        max_word = start;
    }

    for (auto &digit : alphabet) {
        start += digit;
        if (k_free_tail(start, k) && r_free_tail(start, r) && generate(start, k, r, alphabet)) {
            return true;
        } else {
            start.pop_back();
        }
    }

    return false;
}

int main(int argc, char **argv) {
    int begin = clock();
    int k = atoi(argv[1]);
    int r = atoi(argv[2]);

    std::string start;

    std::set<std::string> alphabet(argv+3, argv+argc);

    if (!generate(start, k, r, alphabet)) {
        std::cout << max_word << std::endl;
        std::cout << "length: " << max_word.size() << std::endl;
    }

    std::cout << "time: " << (double)(clock()-begin)/CLOCKS_PER_SEC << 's' << std::endl;
}