Avoiding Additive Powers in Words

by

Jonathan Andrade

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF SCIENCE (HONS.)

in the

Department of Mathematics & Statistics

THOMPSON RIVERS UNIVERSITY

We accept this thesis as conforming to the required standards:

Lucas Mol
Dept. of Mathematics & Statistics
Thesis Supervisor

Sean McGuinness
Dept. of Mathematics & Statistics

Kyle Schlitt
Dept. of Mathematics & Statistics

Dated May 15th, 2024, Kamloops, British Columbia, Canada

THOMPSON RIVERS UNIVERSITY
DEPARTMENT OF MATHEMATICS & STATISTICS

Permission is herewith granted to Thompson Rivers University to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon request of individuals or institutions.

Signature of Author

Abstract

This thesis discusses an implementation of an algorithm for deciding if certain words are additive k-power-free. We use the algorithm to prove new results and to reprove several old results. In particular we construct an infinite additive 5-power-free binary rich word. We reprove that certain infinite binary words found by others are additive 4-power-free.

Acknowledgements

I would like to thank Lucas Mol for his invaluable support over the last year. I would particularly like to thank him for his vast amount of time spent trying to dissect my code, his insightful guidance, and his meticulous editing assistance throughout the thesis writing process.

I would like to say thank you to Saeed Rahmati for helping me register for this honours course, organizing my defence and answering any questions I had. I would like to say thank you to Sean McGuinness and Kyle Schlitt for generously agreeing to be examiners and providing feedback for my thesis. Their feedback was invaluable, and I am grateful for their insightful comments, constructive criticism, and dedication throughout the evaluation process.

I am especially grateful to my parents, who supported me emotionally and financially over the last year. Their support and belief in me were a huge driving force behind my completion of this thesis. For that I am forever grateful.

Contents

Chapter 1

Introduction

Long sequences over fixed alphabets appear in many different areas. Examples include: DNA strings, lattice paths, and binary strings in computing. In a sequence, objects are arranged one after another in successive order. As such, sequences are easy to create. Simply stacking varying coloured blocks on top of one another creates a particular sequence. A natural question then arises about what properties this sequence has; does a certain pattern ever occur? If we added another coloured block, what would change? These are the types of questions that *combinatorics on words* aims to answer.

Combinatorics on words is the study of words and their various properties. A *word* over an alphabet A is a finite or infinite sequence of letters from A. In this thesis, we are interested in long words over small alphabets. We place a primary interest in *pattern avoidance* in infinite words, i.e, over some fixed alphabet, which patterns can be avoided by some infinite word, and which patterns must inevitably occur?

Combinatorics on words as a field of research dates back to the start of the 20th century with the work of Axel Thue [30], [31]. Some simple examples of patterns are *squares* and *cubes*. A *square* (resp. *cube*) is a word of the form xx (resp. xxx), where x is a nonempty word. Words that do not contain squares (resp. cubes) are called *square-free* (resp. *cube-free*). While it is straightforward to check that there are only finitely many square-free words over the binary alphabet, Thue showed that there exists an infinite square-free word over the ternary alphabet and an infinite cube-free word over the binary alphabet [31] (see the translation of Thue's work by Berstel [2]). Both of Thue's constructions can be obtained by starting with a letter and repeatedly applying a special type of map called a *morphism*. This type of construction has continued to be important in combinatorics on words, and is fundamental to this thesis.

A large portion of combinatorics on words is about proving results that are analogous to the results of Thue described above. Here, we focus on a variation of the squares and cubes studied by Thue. Over an alphabet where it makes sense to sum the letters, an *additive square* is a nonempty word $w = w_0w_1$ where w_0 and w_1 have the same length and the same sum of letters. It is unknown whether or not there is an infinite additive square-free word over a finite subset of $\mathbb{Z}$. An *additive cube* is a nonempty word $w = w_0w_1w_2$ where all of w_0, w_1 and w_2 have the same length and the sum of the letters of w_0, w_1, and w_2 are all equal. A word w is

called *additive cube-free* if it contains no additive cubes. Cassaigne et al. [5] showed that there exists an infinite word over the alphabet $\{0, 1, 3, 4\}$ that is additive cube-free. This was done with a similar idea to Thue through the use of an iterated morphism. In general, an *additive k-power* is a nonempty word $w = w_0 w_1 \dots w_{k-1}$ such that all of the w_i's have the same length and the same sum of letters.

In this thesis, we implement an algorithm first described by Currie et al. [7] which decides, under certain conditions, whether or not a word produced by iterating a morphism is additive k-power free. This algorithm follows what is known as the *template method*. The early version of the template method was first developed by Currie and Rampersad [10]. A much more powerful version was developed by Rao and Rosenfeld when attempting to find a construction of an infinite additive square-free words over a finite subset of $\mathbb{Z}$ [25].

Using the algorithm described in the previous paragraph along with the automatic theorem-proving software `Walnut`, we reprove several old results and prove some new results.

- We reprove a result of Currie, Mol, Rampersad and Shallit [7], which says that a certain infinite binary word is additive 4-power-free.
- We reprove a result of Dekking [12], which says that a certain infinite binary word is additive 4-power-free.
- We construct an infinite binary word that is additive 5-power-free and rich.
- We construct an infinite ternary word that is additive 4-power-free and conjectured to be rich.

In Chapter 2, we cover the necessary background on combinatorics on words. Chapter 3 begins with a discussion of the theory behind the algorithm. Next, a detailed description of how the algorithm functions is given. In Chapter 4, we use the algorithm to prove the results described above. In Chapter 5, we discuss some related open problems and conjectures in combinatorics on words.

Chapter 2

Background

2.1 Words

An *alphabet* is a nonempty finite set of symbols, which we refer to as *letters*. Let X be an alphabet. A *word* over X is a finite or infinite sequence of letters from X. We let X^* denote the set of all finite words over the alphabet X. The *length* of a word w is the number of letters that make up w, denoted by $|w|$. We let ε denote the unique word of length 0. We call ε the *empty word*.

For words x and y, the *concatenation* of x and y, denoted by xy, is the word consisting of all of the letters of x followed by all of the letters of y. For example, if $x = \texttt{book}$ and $y = \texttt{shelf}$, then $xy = \texttt{bookshelf}$. Let $u, w \in X^*$. We say that u is a *factor* of w if there exist words $x, z \in X^*$ such that $w = xuz$. If w can be written in this form with $x = \varepsilon$ (resp. $z = \varepsilon$), then u is called a *prefix* (resp. *suffix*) of w. If w can be written in this form with $x \neq \varepsilon$ and $z \neq \varepsilon$, then u is called an *internal factor* of w. For example, the English word `interrelated` has `late` as a factor, `inter` as a prefix, `related` as a suffix and `relate` as an internal factor.

2.2 Morphisms

For alphabets X and Y, a *morphism* from X^* to Y^* is a function $h : X^* \to Y^*$ that satisfies $h(uv) = h(u)h(v)$ for all words $u, v \in X^*$. In particular, it follows that $h(\varepsilon) = \varepsilon$ for every morphism h. In fact, a morphism $h : X^* \to Y^*$ can be described completely by its images on the letters of X. For example, we can define the morphisms $\lambda : \{\texttt{0}, \texttt{1}, \texttt{2}\}^* \to \{\texttt{0}, \texttt{1}, \texttt{2}\}^*$ and $\mu : \{\texttt{0}, \texttt{1}\}^* \to \{\texttt{0}, \texttt{1}\}^*$ by

$$\begin{aligned} \lambda(\texttt{0}) &= \texttt{012}, & \mu(\texttt{0}) &= \texttt{01}, \\ \lambda(\texttt{1}) &= \texttt{02}, & \mu(\texttt{1}) &= \texttt{10}. \\ \lambda(\texttt{2}) &= \texttt{1}, \end{aligned}$$

Then for example, we have

$$\begin{aligned}
\lambda(\mathtt{012}) &= \lambda(\mathtt{01})\lambda(\mathtt{2}) && \text{definition of morphism}\\
&= \lambda(\mathtt{0})\lambda(\mathtt{1})\lambda(\mathtt{2}) && \text{definition of morphism}\\
&= \mathtt{012021} && \text{definition of } \lambda \text{ on letters.}
\end{aligned}$$

Note that the morphism λ, and the famous *Thue-Morse* morphism μ, were studied by Thue [30]. We will be using λ and μ to illustrate the definitions in the remainder of this chapter.

Let $h : X^* \to Y^*$ be a morphism and $a \in X$. For all words $x \in X^*$, we define $h^0(x) = x$, and $h^n(x) = h(h^{n-1}(x))$ for all integers $n \geq 1$. We say that the morphism h is *prolongable* on a if $h(a) = ax$ for some nonempty $x \in X^*$. For example, the morphism λ is prolongable on $\mathtt{0}$ as $\lambda(\mathtt{0}) = \mathtt{012}$, but it is not prolongable on $\mathtt{1}$ as $\lambda(\mathtt{1}) = \mathtt{02}$. If h is prolongable on a, then repeatedly applying the morphism h to a, we obtain the *infinite fixed point* of h around a:

$$h^\omega(a) = \lim_{n\to\infty} h^n(a) = axh(x)h^2(x)h^3(x)\cdots .$$

For example, we can compute the infinite fixed point of λ around $\mathtt{0}$ as follows:

$$\begin{aligned}
\lambda(\mathtt{0}) &= \mathtt{012}\\
\lambda^2(\mathtt{0}) &= \mathtt{012021}\\
\lambda^3(\mathtt{0}) &= \mathtt{012021012102}\\
&\vdots\\
\lambda^\omega(\mathtt{0}) &= \mathtt{012021012102012021020121}\cdots .
\end{aligned}$$

In fact, the infinite word $\lambda^\omega(\mathtt{0})$ is square-free. This was first proven by Thue [31]. Similarly, we can compute the fixed point of μ around $\mathtt{0}$ as follows:

$$\begin{aligned}
\mu(\mathtt{0}) &= \mathtt{01}\\
\mu^2(\mathtt{0}) &= \mathtt{0110}\\
\mu^3(\mathtt{0}) &= \mathtt{01101001}\\
&\vdots\\
\mu^\omega(\mathtt{0}) &= \mathtt{011010011001}\cdots .
\end{aligned}$$

The word $\mu^\omega(\mathtt{0})$ is known as the *Thue-Morse word*. A morphism $h : X^* \to Y^*$ is called *strictly growing* if $|h(a)| \geq 2$ for all $a \in X$. For instance, the morphism μ is strictly growing as we have $|\mu(a)| = 2$ for all $a \in \{\mathtt{0}, \mathtt{1}\}$. The morphism λ is not strictly growing because $|\lambda(2)| = 1 \ngeq 2$. For an integer $k \geq 1$, we say that a morphism $h : X^* \to Y^*$ is *k-uniform* if $|h(a)| = k$ for all $a \in X$. Note that the morphism μ is a 2-uniform morphism as $|\mu(\mathtt{0})| = |\mu(\mathtt{1})| = 2$. Meanwhile, the morphism λ is *non-uniform*, i.e., it is not k-uniform for any integer k.

2.3 Ordinary Powers

The work done in this thesis revolves mainly around the idea of what we call powers in words. A *square* is a word of the form xx, where x is a nonempty word. An example of a square in the English language is `murmur` with $x =$ `mur`. We call a word w *square-free* if w contains no squares as factors. For example, the word `1021` is square-free, while the words 00123012301 and 01210121012 are not square-free. A backtracking algorithm shows that the longest square-free word over 2 letters is of length 3. Over alphabets of size at least 3, it is possible to construct arbitrarily long square-free words [30].

A *cube* is a nonempty word of the form xxx. An example of a cube in the English Language is the word `shshsh` (an admonition to be quiet) [29]. More generally, for every integer $k \geq 2$, a *k-power* is a nonempty word of the form $x^k = xx\cdots x$ where x is a word concatenated with itself k times. Note that a 2-power is a square, and a 3-power is a cube.

There is no reason to restrict ourselves to integer powers of words. Dejean [11] was the first to generalize to fractional powers. Let $p \in \mathbb{Z}$ such that $1 \leq p \leq n$, and let $w = w_1 w_2 \cdots w_n$, where the w_i's are letters. We say that p is a *period* of w if $w_i = w_{i+p}$ for $i = 1, ..., n-p$. For example, the word `alfalfa` has periods 3, 6, and 7. The *exponent* of w is its length divided by its minimum period. For example, `alfalfa` has length 7 and minimum period 3, so it has exponent 7/3. A word of exponent α is also called an *α-power*. Words that do not contain a factor with exponent greater than or equal to α are known as *α-power-free*. For example, `alfalfa` is 3-power-free as it contains no cube, but it is not 2-power-free, because it contains the square `alfalf`. For a finite or infinite word $\mathbf{w}$, we define the *critical exponent* of $\mathbf{w}$ as

$$\sup\{\alpha \in \mathbb{Q} \mid \text{there is a factor of } \mathbf{w} \text{ with exponent } \alpha\}.$$

For example, the Thue-Morse sequence has critical exponent 2, because it contains factors of exponent 2, but no factors of exponent greater than 2 [31]. Let $\alpha \geq 1$. A word that has exponent greater than α is known as an *α^+-power*. A word w is *α^+-power-free* if it contains no α^+-power as a factor. For example, the Thue-Morse word is 2^+-power-free, but not 2-power-free. The word `sausage` contains a $\frac{5}{3}$-power, but is $\frac{5}{3}^+$-power-free. The *repetition threshold* for k letters, denoted $\mathrm{RT}(k)$, is defined as:

$$\inf\{\alpha \in \mathbb{R} \mid \text{there exists an infinite } \alpha\text{-power free word over } k \text{ letters}\}.$$

Dejean's theorem describes the repetition threshold for k letters for all $k \geq 2$:

$$\mathrm{RT}(k) = \begin{cases} 2, & \text{if } k = 2; \\ \frac{7}{4}, & \text{if } k = 3; \\ \frac{7}{5}, & \text{if } k = 4; \\ \frac{k}{k-1}, & \text{if } k \geq 5. \end{cases}$$

Note that we have $\mathrm{RT}(2) = 2$ since every binary word of length greater than 3 contains a square, but the Thue-Morse word has critical exponent 2, i,.e., it is 2^+-power-free. Similarly,

we have $\mathrm{RT}(3) = \frac{7}{4}$ since every sufficiently long ternary word contains a $\frac{7}{4}$-power, but the word $\delta^\omega(\mathtt{0})$ is $\frac{7}{4}^+$-power-free, where $\delta : \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \to \{\mathtt{0},\mathtt{1},\mathtt{2}\}^*$ is defined by

$$\begin{aligned}\delta(\mathtt{0}) &= \mathtt{01202120121021202 10},\\ \delta(\mathtt{1}) &= \mathtt{12010201202102010 21},\\ \delta(\mathtt{2}) &= \mathtt{20121012010210121 02}.\end{aligned}$$

The morphism δ was found by Dejean [11], who proved that $\delta^\omega(\mathtt{0})$ has critical exponent $7/4$. The proof of Dejean's theorem for larger alphabets relies on more complicated constructions, and took the work of many different authors [8], [9], [20], [23]. Carpi [4] made a major contribution in proving Dejean's theorem by providing a proof for $k \geq 33$.

2.4 Abelian Powers

We will now explore another type of power known as an abelian power. Two words x and y are called *abelian equivalent* if they are anagrams of each other. For example, the English words `dog` and `god` are abelian equivalent, as are the words `add` and `dad`. An *abelian square* is a nonempty word $w = xy$ such that x and y are abelian equivalent. Some examples of abelian squares in the English language include `mesosome`, `reappear`, and `intestines` [29]. An *abelian cube* is a nonempty word $w = xyz$ such that x, y, and z are abelian equivalent. An example in the English language is `deeded` [29]. In general, for an integer $k \geq 2$, an *abelian k-power* is a nonempty word of the form $w = w_0 w_1 \dots w_{k-1}$ where w_i and w_j are abelian equivalent for all $i, j \in \{0, 1, \dots, k-1\}$. A word w is *abelian k-power-free* if it contains no abelian k-powers as factors. For example, the word $w = \mathtt{010}$ is abelian square-free and the word $w = \mathtt{0012102}$ is abelian cube-free. Meanwhile, the words $w = \mathtt{011200}$ and $w = \mathtt{21302121}$ are not abelian 2-power-free.

Dekking [12] showed the existence of an infinite binary word that is abelian 4-power-free. In particular, Dekking showed that the infinite fixed point around $\mathtt{0}$ of the morphism $h : \{\mathtt{0},\mathtt{1}\}^* \to \{\mathtt{0},\mathtt{1}\}^*$ defined by

$$\begin{aligned}h(\mathtt{0}) &= \mathtt{011},\\ h(\mathtt{1}) &= \mathtt{0001},\end{aligned}$$

is abelian 4-power free. This result is best possible in the sense that there are only finitely many abelian cube-free binary words; a backtracking algorithm shows that the longest abelian cube-free binary word has length 9. Dekking also constructed an infinite abelian cube-free ternary word. He showed that the infinite fixed point around $\mathtt{0}$ of the morphism $h : \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \to \{\mathtt{0},\mathtt{1},\mathtt{2}\}^*$ defined by

$$\begin{aligned}h(\mathtt{0}) &= \mathtt{0012},\\ h(\mathtt{1}) &= \mathtt{112},\\ h(\mathtt{2}) &= \mathtt{022},\end{aligned}$$

is abelian cube-free. Note that this result is best possible in the sense that there are only finitely many abelian square-free ternary words; a backtracking search shows that the longest abelian square-free ternary word has length 537.

Erdős [14] asked if there exists an infinite abelian square-free word over some finite alphabet. Evdokimov [15] answered this question in the positive by constructing an infinite abelian square-free word over an alphabet of size 25. The size of the alphabet was improved upon by Pleasants [22], who was able to reduce it down to size 5. This was even further reduced down to an alphabet of size 4 by Keränen [17]. The result by Keränen is the best possible since backtracking shows that over three letters, there is no abelian square-free word of length greater than 7. Keränen used the 85-uniform morphism $g : \{0,1,2,3\}^* \to \{0,1,2,3\}^*$ defined by:

$g(0) = 0120232123203231301020103101213121021232021013010203212320231210212320232132303132120,$
$g(1) = 1231303230310302012131210212320232132303132120121310323031302321323031303203010203231,$
$g(2) = 2302010301021013123202321323031303203010203231232021030102013032030102010310121310302,$
$g(3) = 3013121012132120230313032030102010310121310302303132101213120103101213121021232021013.$

2.5 Additive Powers

We now introduce one more variation on powers in words, namely *additive powers*. Note that we restrict to words over alphabets where there is some notion of sum, so that we can sum all of the letters in a word. The *sum* of w, denoted by $S(w)$, is the sum of all the letters that make up w. An *additive square* is a nonempty word $w = w_0w_1$, where both w_0 and w_1 have the same length and the same sum, i.e., $|w_0| = |w_1|$ and $S(w_0) = S(w_1)$. A word w is said to be *additive square-free* if it contains no additive squares as factors. For example, the word $w = 0120$ is additive square-free, and the word $w = 013121$ contains an additive square.

Additive square-free words were studied by Rao and Rosenfeld [25], who were able to show that there exists an infinite additive square-free word over a finite subset of $\mathbb{Z}^2$. It is still unknown whether or not there exists such a word over some finite subset of $\mathbb{Z}$.

An *additive cube* is a nonempty word $w = w_0w_1w_2$ where all of w_0, w_1, and w_2, have the same length and the same sum. A word w is called *additive cube-free* if it contains no additive cubes as factors. For example, the word $w = 020312$ is additive cube-free, but it contains the additive square $u = 0312$. Cassaigne et al. [5] showed that there exists an infinite word over the alphabet $\{0,1,3,4\}$ that is additive cube-free. Lietard and Rosenfeld [18] showed that there exists an infinite additive cube-free word over all subsets of $\mathbb{Z}$ of size 4 except possibly subsets equivalent to $\{0,1,2,3\}$, i.e., any arithmetic progression of length 4. Rao [24] was able to show that there exist infinite additive cube-free words over several integer alphabets of size 3 including $\{0,1,5\}$, $\{0,3,7\}$, and $\{0,2,9\}$. Note that it is still unknown if such words exist over the alphabets $\{0,1,2\}$, $\{0,1,3\}$, $\{0,1,4\}$, and $\{0,2,5\}$.

In general, an *additive k-power* is a nonempty word $w = w_0w_1 \dots w_{k-1}$ such that all of the w_i's have the same length and the same sum. Recall from Section 2.4 that Dekking

produced an infinite abelian 4-power-free binary word. Over the binary alphabet $\{0, 1\}$, it is straightforward to show that a word w is an abelian k-power if and only if it is an additive k-power. Thus, we can extend Dekking's earlier conclusion and see that there exists an infinite additive 4-power-free word over the binary alphabet. A backtracking search shows that this result is best possible as over the binary alphabet, there is no additive 3-power-free word of length greater than 9.

2.6 Rich Words

A *palindrome* is a word w that reads the same backward as forward. Some examples in the English language include `racecar`, `level`, and `kayak`. It is known that every word of length n contains at most $n + 1$ palindromic factors, including the empty word [13]. This fact led to the study of words that are "rich" in palindromes. We say that a finite word w is *rich* if it contains $|w| + 1$ distinct palindromic factors. In other words, a word w is rich if it contains the maximum number of palindromic factors among all words of length $|w|$. An example of a rich word is `1001` since it has 5 palindromic factors, namely ε, `0`, `1`, `00` and `1001`. Note that `012120` is not rich as it has only 6 palindromic factors: ε, `0`, `1`, `2`, `121`, `212`. An infinite word $\mathbf{w}$ is *rich* if all of its finite factors are rich. For example, the *Fibonacci word* $\mathbf{F} = \texttt{01001010}\ldots$ where $\mathbf{F}$ is the infinite fixed point of the morphism $\phi : \{0, 1\}^* \to \{0, 1\}^*$ defined by $\phi(0) = \texttt{01}$ and $\phi(1) = \texttt{0}$ is known to be an infinite rich word [16]. The Thue-Morse word $\mu^\omega(0) = \texttt{0110100110010110}\cdots$ is an example of an infinite non-rich word. For example, the factor `11010011` of $\mu^\omega(0)$ is non-rich as it has length 8 and 8 palindromic factors: ε, `0`, `1`, `00`, `11`, `101`, `010`, and `1001`.

A factor u of a word w is said to be *unioccurrent* in w if u occurs exactly once in w. For example, in the word `01`, the factors `0`, `1` and `01` are all unioccurrent. In the word `011001`, the factor `01` is not unioccurrent, but `001` is. The following theorem gives a useful characterization of finite rich words in terms of unioccurrent factors.

Theorem 2.6.1 (Glen et al. [16])**.** *A finite word w is rich if and only if every prefix of w has a unioccurrent palindromic suffix.*

The result of Theorem 2.6.1 can be extended to a well-known result about infinite rich words. We provide a proof below.

Theorem 2.6.2. *An infinite word $\mathbf{w}$ is rich if and only if every finite prefix of $\mathbf{w}$ has a unioccurrent palindromic suffix.*

Proof.

- $\Rightarrow$ Suppose the infinite word $\mathbf{w}$ is rich. Then by definition, all of its finite factors are rich. This implies by Theorem 2.6.1 that every one of the finite factors of $\mathbf{w}$ has a unioccurrent palindromic suffix. Thus, every finite prefix of $\mathbf{w}$ has a unioccurrent palindromic suffix.
- $\Leftarrow$ Suppose that every finite prefix of the infinite word $\mathbf{w}$ has a unioccurrent palindromic suffix. Let x be a finite factor of $\mathbf{w}$. Then there is some finite prefix p of $\mathbf{w}$ that contains

x. Since every finite prefix of $\mathbf{w}$ has a unioccurrent palindromic suffix, we see by Theorem 2.6.1 that p must be rich. By [16, Corollary 2.5], since p is rich, every one of its factors must also be rich. Since x is contained in p, this implies that x is rich. Therefore, we conclude that $\mathbf{w}$ is rich.

□

After being introduced implicitly by Droubay, Justin, and Pirillo [13], rich words have been well-studied, sometimes under the name *full words* (see [3]). We will now briefly summarize the known results about repetitions in rich words. Pelantová and Starosta [21] proved that every infinite rich word, over *any* finite alphabet, contains a square. Shallit and Baranwal [1] proved that the infinite binary word $g(f^\omega(\mathtt{0}))$ is rich and has critical exponent $2+\sqrt{2}/2$, where $f : \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \to \{\mathtt{0},\mathtt{1},\mathtt{2}\}^*$ and $g : \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \to \{\mathtt{0},\mathtt{1}\}^*$ are defined by

$$\begin{aligned} f(\mathtt{0}) &= \mathtt{01}, & g(\mathtt{0}) &= \mathtt{0}, \\ f(\mathtt{1}) &= \mathtt{02}, & g(\mathtt{1}) &= \mathtt{01}, \\ f(\mathtt{2}) &= \mathtt{022}, & g(\mathtt{2}) &= \mathtt{011}. \end{aligned}$$

Further, they conjectured that this word has the least critical exponent among all infinite binary rich words, i.e., that the repetition threshold for binary rich words is $2+\sqrt{2}/2$. Currie, Mol and Rampersad [6] proved this conjecture of Shallit and Baranawal by giving a structure theorem for infinite binary rich words that are $\frac{14}{5}$-power-free. In Section 4, we will show that there is an infinite additive 5-power-free binary rich word, and that this is best possible in the sense that there are only finitely many additive 4-power-free binary rich words. One can think of this result as an additive-power analogue of the result of Currie, Mol, and Rampersad.

2.7 Walnut

Walnut is a free Java software that was originally developed by Hamoon Mousavi [19] and has been added to by Aseem Raj Baranwal, Laindon C. Burnett, Kai Hsiang Yang, and Anatoly Zavyalov. To download Walnut for free, visit

https://cs.uwaterloo.ca/~shallit/walnut.html

Walnut is an implementation of the algorithm described in chapter 6 of the Walnut book [28, Theorem 6.4.1]. Roughly speaking, this theorem says that there is an algorithm which can determine the truth value of all statements about automatic sequences that can be written in a certain first-order logic. The internal representation of natural numbers in Walnut by default is base-2 representation with the most significant digit coming first. Walnut is quite flexible and supports *k-automatic* sequences for integers $k \geq 2$, *Fibonacci* and *Tribonacci-automatic* sequences, *Pell-automatic* sequences, and *Ostrowski-automatic* sequences based on arbitrary quadratic irrationals.

2.7.1 Walnut Syntax

Walnut provides a user interface that accepts input in the form of first-order logical formulas. An *automaton* is created to represent these various inputs. Automata are one of the simplest forms of a computer. An automaton starts in an initial state and after reading the input provided, transitions to one of a finite number of states. An automaton takes as input strings and based on the symbols in the strings, moves from state to state. Each Walnut automaton accepts tuples of integers in a user specified numeration system. These automata are stored in .txt files with easy to read and understand syntax. All automata created by Walnut are deterministic and minimal.

One can express first-order logical formulas in Walnut with a relatively simple syntax. First of all, words made up of capital letters (apart from the reserved letters A and E) are used to refer to sequences stored in the directory Word Automata Library. For example, some hard-coded sequences, namely the Thue-Morse and Fibonnaci sequences, have respective symbols T and F.

Indexing sequences in Walnut is fairly easy. Note that all variables in Walnut have domain $\mathbb{N} = \{0, 1, 2, \ldots\}$ and thus do not accept negative values. For $i \in \mathbb{N}$, we write W[i] for the symbol at position i in W. For example, since

$$\mathbf{T} = 01101001\cdots,$$

the input T[0] would return the symbol 0, and T[2] would give the symbol 1. Algebraic expressions may also be used to index sequences. For example, to access the symbol at position $x+y$ in W we write W[x+y]. Note that Walnut supports addition, subtraction, multiplication by a natural number constant, and integer division, where x/y represents the floor of the quotient of x and y.

The letter E is the Walnut symbol for the *existential quantifier* $\exists$. The letter A is the Walnut symbol for the *universal quantifier* $\forall$. The symbol => represents logical implication, & represents AND, | represents OR, and <=> represents the biconditional (if and only if).

There is an extra command that is used to designate the numeration system for the automata involved in evaluating the command. The command ?msd_n preceding a first-order logical formula specifies that it should be evaluated in base-n, where $n \geq 2$ is some integer.

Example 2.7.1. We will use Walnut to show that the Thue-Morse word $\mathbf{T} = 0110100110\cdots$ is cube-free and 2^+-power-free. In order to do this, we write a first-order logical formula that asserts the existence of a cube and 2^+-power in Thue-Morse. We give the construction for the cube. Recall that a cube is a word w of the form xxx for some nonempty word x (see Section 2.1).

We define $T[a \ldots b]$ to represent the word starting at position a and ending at position b inclusive. The idea is that if the word $\mathbf{T}$ contains xxx starting at position i, we must have $|x| = n$ for some $n \geq 1$ (see Figure 2.1). So there is a cube in $\mathbf{T}$ if and only if there are natural numbers i and n with $n \geq 1$ such that

$$\mathbf{T}[i \ldots i+n-1] = \mathbf{T}[i+n \ldots i+2n-1] = \mathbf{T}[i+2n \ldots i+3n-1]$$

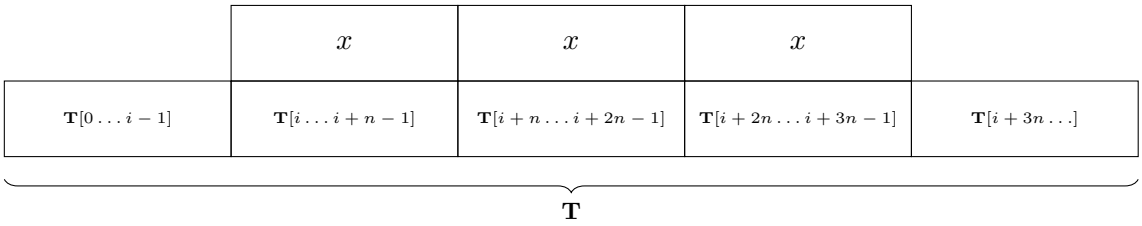

Figure 2.1: An instance of a cube occurring in $\mathbf{T}$.

or equivalently,

$$\exists i, n, (n \geq 1) \wedge \forall s(0 \leq s < 2n) \implies \mathbf{T}[i+s] = \mathbf{T}[i+n+s].$$

We can translate this into `Walnut` as:

```
Ei, n(n>=1) & As (s<2*n) => T[i+s] = T[i+n+s]:
```

Now that we have the `Walnut` syntax for checking T for cubes, all we need to do is simply evaluate it using `Walnut`. To do so we use the `eval` command followed by the statement name, and we include `?msd_2` immediately before the formula to indicate that we are using the base-2 numeration system with most significant digit first:

```
eval containscubes "?msd_2 Ei, n(n>=1) & As (s<2*n) => T[i+s] = T[i+n+s]":
```

`Walnut` returns `FALSE`, thus proving that the Thue-Morse word is cube-free. We can prove that the Thue-Morse word is 2^+-power-free using a similar command:

```
eval containstwoplus "?msd_2 Ei, n(n>=1) & As (s<=n) => T[i+s] = T[i+n+s]":
```

`Walnut` returns `FALSE`, thus proving that the Thue-Morse word is 2^+-power-free.

It is possible to add your own automatic sequences to `Walnut`. For example, consider the morphism $h : \{0,1\}^* \to \{0,1\}^*$ defined by $h(0) = 001$ and $h(1) = 100$. To define h in `Walnut`, we use the following command:

```
h "0->001 1->100":
```

If the morphism is uniform, it can be "promoted" to an automaton through the use of the `promote` command. For example, using our morphism h as defined above, we can write

```
promote H h:
```

which creates an automaton H that generates $h^\omega(0)$. For words that are of the form $g(h^\omega(0))$, we use the `image` command. For example, if g and h are uniform morphisms, then we can write:

```
promote H h:
# promotes h to an automaton H that generates the fixed point of h around 0

image GH g H:
# creates the automaton GH which generates the word g(h^{w}(0))
```

The resulting automaton `GH` is defined over the same numeration system as the original automaton H.

Chapter 3

Decision Algorithm

In this chapter we discuss a decision algorithm that checks if certain types of morphic sequences are additive k-power-free. Currie et al. [7] devised this method, and described it for additive k-powers. They demonstrated that the words $f^\omega(\mathtt{0})$ and $g(f^\omega(\mathtt{0}))$ are additive 4-power-free, where $f : \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \to \{\mathtt{0},\mathtt{1},\mathtt{2}\}^*$ is the morphism defined by

$$\begin{aligned} f(\mathtt{0}) &= \mathtt{001}, \\ f(\mathtt{1}) &= \mathtt{012}, \\ f(\mathtt{2}) &= \mathtt{212}, \end{aligned}$$

and $g : \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \to \{\mathtt{0},\mathtt{1}\}^*$ is the morphism defined by

$$\begin{aligned} g(\mathtt{0}) &= \mathtt{00010011100100011100011}, \\ g(\mathtt{1}) &= \mathtt{00100111001110110011}, \\ g(\mathtt{2}) &= \mathtt{01110011100111011001 1}. \end{aligned}$$

The algorithm is based on the *template method*. An early version of this method was described by Currie and Rampersad [10]. A more powerful version was described by Rao and Rosenfeld [25]. This chapter is an overview of the decision algorithm used in this thesis. In Section 3.1, we describe the theory behind the algorithm. In Section 3.2, we describe the algorithm in detail, and explain why it works.

3.1 Theory of the Algorithm

The results and proofs given in this section were given by Currie et al. [7] for $k = 4$, and have been generalized to all $k \geq 2$ below.

We first introduce additive k-templates, which describe structures that are not "too far" from an additive k-power. Given an integer alphabet X, an *additive k-template* (for additive k-powers) is a $2k$-tuple

$$t = [a_0, a_1, \ldots, a_k, d_0, d_1, \ldots, d_{k-2}] \tag{3.1}$$

where $a_i \in X^*$, $|a_i| \leq 1$, and each of the d_i are in $\mathbb{Z}^2$. For example, an additive 3-template is a 6-tuple $t = [a_0, a_1, a_2, a_3, d_0, d_1]$. Recall that for a finite word w over an integer alphabet, the length of w is denoted $|w|$, and the sum of the letters of w is denoted $S(w)$. We define $\sigma(w)$ to be the vector $[|w|, S(w)]^T$. For example, if $w = \texttt{01210}$, then $\sigma(w) = [5, 4]^T$. A word x is an *instance* of t if there exist $x_0, x_1, \ldots, x_{k-1} \in X^*$ such that $x = a_0 x_0 a_1 x_1 \cdots x_{k-1} a_k$ and for each i, we have $d_i = \sigma(x_{i+1}) - \sigma(x_i)$. Observe that an additive k-power is an instance of the additive k-template

$$t_0 = [\varepsilon, \varepsilon, \ldots, \varepsilon, [0,0]^T, [0,0]^T, \ldots, [0,0]^T].$$

The word $v = \texttt{102122}$ is an instance of the template

$$[1, \varepsilon, 2, [0,1]^T]$$

where $x_0 = \texttt{02}$, and $x_1 = \texttt{12}$. The word $w = \texttt{1110202011}$ is an instance of the template

$$[1, \varepsilon, \varepsilon, \varepsilon, 1, [0,0]^T, [0,0]^T, [0,-1]^T]$$

where $x_0 = \texttt{11}$, $x_1 = \texttt{02}$, $x_2 = \texttt{02}$ and $x_3 = \texttt{01}$. Note that the word v is not "too far" from an additive 2-power and w is not "too far" from an additive 4-power.

Let $h : X^* \to Y^*$ be a morphism. For $A \in X^*$ with $|A| \leq 1$, an *h-split* of A is a 3-tuple $[p, a, s]$, where $h(A) = pas$, $a \in Y^*$, and $|a| \leq 1$. One can think of an h-split of A as a way in which the word $h(A)$ can be factored into three pieces, with the empty word or a single letter in the center. For example, let $\beta : \{\texttt{0}, \texttt{1}, \texttt{2}\}^* \to \{\texttt{0}, \texttt{1}, \texttt{2}\}^*$ be the morphism defined by

$$\begin{aligned} \beta(\texttt{0}) &= \texttt{001}, \\ \beta(\texttt{1}) &= \texttt{012}, \\ \beta(\texttt{2}) &= \texttt{212}, \end{aligned}$$

and $\gamma : \{\texttt{0}, \texttt{1}, \texttt{2}\}^* \to \{\texttt{0}, \texttt{1}\}^*$ be the morphism defined by

$$\begin{aligned} \gamma(\texttt{0}) &= \texttt{00001}, \\ \gamma(\texttt{1}) &= \texttt{000111}, \\ \gamma(\texttt{2}) &= \texttt{0111011}. \end{aligned}$$

Then the β-splits of 0, 1, 2, and ε respectively are:

- $[\varepsilon, \texttt{0}, \texttt{01}]$, $[\texttt{0}, \texttt{0}, \texttt{1}]$, $[\texttt{00}, \texttt{1}, \varepsilon]$, $[\varepsilon, \varepsilon, \texttt{001}]$, $[\texttt{0}, \varepsilon, \texttt{01}]$, $[\texttt{00}, \varepsilon, \texttt{1}]$, $[\texttt{001}, \varepsilon, \varepsilon]$,
- $[\varepsilon, \texttt{0}, \texttt{12}]$, $[\texttt{0}, \texttt{1}, \texttt{2}]$, $[\texttt{01}, \texttt{2}, \varepsilon]$, $[\varepsilon, \varepsilon, \texttt{012}]$, $[\texttt{0}, \varepsilon, \texttt{12}]$, $[\texttt{01}, \varepsilon, \texttt{2}]$, $[\texttt{012}, \varepsilon, \varepsilon]$,
- $[\varepsilon, \texttt{2}, \texttt{12}]$, $[\texttt{2}, \texttt{1}, \texttt{2}]$, $[\texttt{21}, \texttt{2}, \varepsilon]$, $[\varepsilon, \varepsilon, \texttt{212}]$, $[\texttt{2}, \varepsilon, \texttt{12}]$, $[\texttt{21}, \varepsilon, \texttt{2}]$, $[\texttt{212}, \varepsilon, \varepsilon]$,
- $[\varepsilon, \varepsilon, \varepsilon]$,

. The γ-splits of 0, 1, 2, and ε respectively are:

- $[\varepsilon, \texttt{0}, \texttt{0001}]$, $[\texttt{0}, \texttt{0}, \texttt{001}]$, $[\texttt{00}, \texttt{0}, \texttt{01}]$, $[\texttt{000}, \texttt{0}, \texttt{1}]$, $[\texttt{0000}, \texttt{1}, \varepsilon]$, $[\varepsilon, \varepsilon, \texttt{00001}]$, $[\texttt{0}, \varepsilon, \texttt{0001}]$, $[\texttt{00}, \varepsilon, \texttt{001}]$, $[\texttt{000}, \varepsilon, \texttt{01}]$, $[\texttt{0000}, \varepsilon, \texttt{1}]$, $[\texttt{00001}, \varepsilon, \varepsilon]$,

- $[\varepsilon, \mathtt{0}, \mathtt{00111}], [\mathtt{0}, \mathtt{0}, \mathtt{0111}], [\mathtt{00}, \mathtt{0}, \mathtt{111}], [\mathtt{000}, \mathtt{1}, \mathtt{11}], [\mathtt{0001}, \mathtt{1}, \mathtt{1}], [\mathtt{00011}, \mathtt{1}, \varepsilon]$,
  $[\varepsilon, \varepsilon, \mathtt{000111}], [\mathtt{0}, \varepsilon, \mathtt{00111}], [\mathtt{00}, \varepsilon, \mathtt{0111}], [\mathtt{000}, \varepsilon, \mathtt{111}], [\mathtt{0001}, \varepsilon, \mathtt{11}], [\mathtt{00011}, \varepsilon, \mathtt{1}], [\mathtt{000111}, \varepsilon, \varepsilon]$,
- $[\varepsilon, \mathtt{0}, \mathtt{111011}], [\mathtt{0}, \mathtt{1}, \mathtt{11011}], [\mathtt{01}, \mathtt{1}, \mathtt{1011}], [\mathtt{011}, \mathtt{1}, \mathtt{011}], [\mathtt{0111}, \mathtt{0}, \mathtt{11}][\mathtt{01110}, \mathtt{1}, \mathtt{1}], [\mathtt{011101}, \mathtt{1}, \varepsilon], [\varepsilon, \varepsilon, \mathtt{0111011}]$,
  $[\mathtt{0}, \varepsilon, \mathtt{111011}], [\mathtt{01}, \varepsilon, \mathtt{11011}], [\mathtt{011}, \varepsilon, \mathtt{1011}], [\mathtt{0111}, \varepsilon, \mathtt{011}], [\mathtt{01110}, \varepsilon, \mathtt{11}], [\mathtt{011101}, \varepsilon, \mathtt{1}], [\mathtt{0111011}, \varepsilon, \varepsilon]$,
- $[\varepsilon, \varepsilon, \varepsilon]$,

A morphism $h : X^* \to Y^*$ where $X, Y \subseteq \mathbb{Z}$ is called an *affine morphism* if there exist $a, b, c, d \in \mathbb{Z}$ such that for all $x \in X$, we have $|h(x)| = a + bx$ and $S(h(x)) = c + dx$. For an affine morphism h, we define the matrix

$$M_h = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

For example, let β be the morphism defined previously. Then for all $x \in \{\mathtt{0}, \mathtt{1}, \mathtt{2}\}$, we have $|\beta(x)| = 3 + 0x$ and $S(\beta(x)) = 1 + 2x$, so β is affine, and we have

$$M_\beta = \begin{bmatrix} 3 & 0 \\ 1 & 2 \end{bmatrix}.$$

Similarly for the morphism γ, we have $|\gamma(x)| = 5 + x$ and $S(\gamma(x)) = 1 + 2x$ for all $x \in \{\mathtt{0}, \mathtt{1}\}$. Therefore, γ is affine, and we have

$$M_\gamma = \begin{bmatrix} 5 & 1 \\ 1 & 2 \end{bmatrix}.$$

Lemma 3.1.1. *Let X and Y be finite subsets of $\mathbb{Z}$ and $h : X^* \to Y^*$ be an affine morphism with*

$$M_h = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

Then for any word $w \in X^*$ we have $\sigma(h(w)) = M_h\sigma(w)$. Thus, if M_h^{-1} exists, then $\sigma(w) = M_h^{-1}\sigma(h(w))$.

Proof. Let w be a word in X^*. We proceed by induction on $|w|$. If $|w| = 0$, then w is empty, so we have $\sigma(h(w)) = \sigma(h(\varepsilon)) = \sigma(\varepsilon) = [0, 0]^T$ and $M_h\sigma(w) = M_h[0, 0]^T = [0, 0]^T$. Therefore the result holds when $|w| = 0$. Now suppose that $|w| > 0$, and that the statement holds for all words of length $|w| - 1$. Then we write $w = w'x$, where x is the final letter in w, and we have

$$\begin{aligned}
M_h\sigma(w) &= M_h(\sigma(w') + \sigma(x)) && \text{property of } \sigma \\
&= M_h\sigma(w') + M_h\sigma(x) && \text{distributive property} \\
&= \sigma(h(w')) + M_h\sigma(x) && \text{inductive hypothesis} \\
&= \sigma(h(w')) + \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} && \text{definition of } M_h \text{ and } \sigma \\
&= \sigma(h(w')) + \begin{bmatrix} a + bx \\ c + dx \end{bmatrix} && \text{matrix multiplication} \\
&= \sigma(h(w')) + \sigma(h(x)) && \text{definition of affine} \\
&= \sigma(h(w)) && \text{property of } \sigma
\end{aligned}$$

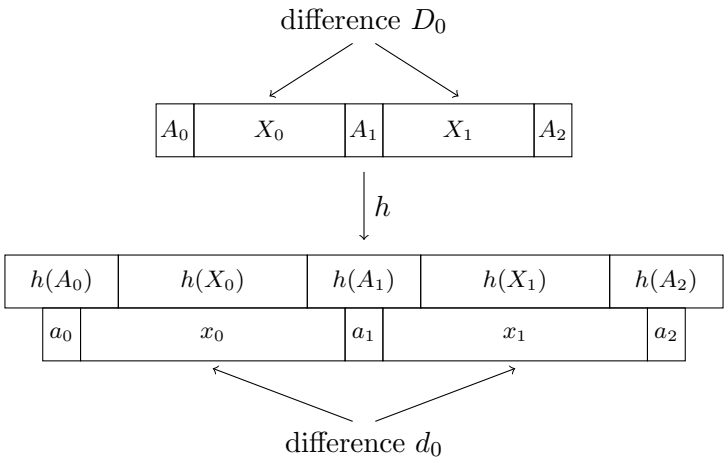

Figure 3.1: An instance of a template $t = [a_0, a_1, a_2, d_0]$ arising from an instance of its h-parent $T = [A_0, A_1, A_2, D_0]$.

Therefore, the result follows by induction. □

Let

$$t = [a_0, a_1, \ldots, a_k, d_0, d_1, \ldots, d_{k-2}],$$

and

$$T = [A_0, A_1, \ldots, A_k, D_0, D_1, \ldots, D_{k-2}],$$

be templates. We call T an *h-parent* of t if the A_i have h-splits $[p_i, a_i, s_i]$ such that $d_i = M_h D_i + b_i$, where $b_i = \sigma(s_{i+1}p_{i+2}) - \sigma(s_i p_{i+1})$. Roughly speaking, the next lemma says that an instance of t arises when we apply h to an instance of T (see Figure 3.1).

Lemma 3.1.2. *Let $h : X^* \to Y^*$ be an affine morphism and $W \in X^*$. Let*

$$t = [a_0, a_1, \ldots, a_k, d_0, d_1, \ldots, d_{k-2}] \quad \textit{and} \quad T = [A_0, A_1, \ldots, A_k, D_0, D_1, \ldots, D_{k-2}]$$

be additive k-templates such that T is an h-parent of t. If W contains an instance of T, then $h(W)$ contains an instance of t.

Proof. Suppose that W contains an instance $V = A_0X_0A_1X_1 \cdots X_{k-1}A_k$ of T. Since T is an h-parent of t, there are h-splits $[p_0, a_0, s_0], [p_1, a_1, s_1], \ldots, [p_k, a_k, s_k]$ of $A_0, A_1, \ldots, A_k$ respectively such that for all $i \in \{0, 1, \ldots, k-2\}$, we have $d_i = M_h D_i + b_i$, where $b_i = \sigma(s_{i+1}p_{i+2}) - \sigma(s_i p_{i+1})$.

Thus $w = h(W)$ contains the factor $v = a_0x_0a_1x_1 \cdots x_{k-1}a_k$, where $x_i = s_ih(X_i)p_{i+1}$. We have

$$\begin{aligned}
\sigma(x_{i+1}) - \sigma(x_i) &= \sigma(s_{i+1}h(X_{i+1})p_{i+2}) - \sigma(s_ih(X_i)p_{i+1}) && \text{substituting in for } x_i \text{ and } x_{i+1}\\
&= \sigma(h(X_{i+1})) + \sigma(s_{i+1}p_{i+2}) - \sigma(h(X_i)) - \sigma(s_ip_{i+1}) && \text{property of } \sigma\\
&= M_h\sigma(X_{i+1}) + \sigma(s_{i+1}p_{i+2}) - M_h\sigma(X_i) - \sigma(s_ip_{i+1}) && h \text{ is affine so } \sigma(h(X_i)) = M_h\sigma(X_i)\\
&= M_h(\sigma(X_{i+1}) - \sigma(X_i)) + \sigma(s_{i+1}p_{i+2}) - \sigma(s_ip_{i+1}) && \text{factor out } M_h\\
&= M_hD_i + b_i && V \text{ is an instance of } T\text{, definition of } b_i\\
&= d_i && \text{definition of parent}
\end{aligned}$$

Hence, the word v is an instance of t. □

For a morphism $h : X^* \to Y^*$, we define

$$W_h = \max_{a \in X} |h(a)|.$$

So for example if γ and β are the morphisms defined previously, then we have $W_\gamma = 7$ and $W_\beta = 3$. Let $h : X^* \to Y^*$ be a morphism and

$$t = [a_0, a_1, \ldots, a_k, d_0, d_1, \ldots, d_{k-2}]$$

be an additive k-template. Then we write $d_i = \left[d_i^{(0)}, d_i^{(1)}\right]^T$. Furthermore, we define the values

$$\Delta(t) = \max\{|d_i^{(0)}| : i = 0, \ldots, k-2\}$$

and

$$B_h(t) = k + 2 + k(W_h - 2) + \frac{(k-1)k}{2}\Delta(t).$$

For example, if we look at the morphism γ and the additive 4-template

$$t = [\varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon, [1,0]^T, [2,1]^T, [0,1]^T],$$

then

$$d_0 = [1,0]^T, d_1 = [2,1]^T, \text{ and } d_2 = [0,1]^T.$$

Hence

$$\Delta(t) = \max\{1, 2, 0\} = 2.$$

Therefore, we have

$$B_\gamma(t) = 4 + 2 + 4 \cdot (7-2) + \frac{(4-1)\cdot 4}{2} \cdot 2 = 38.$$

In an additive k-power, the k "blocks" must have the same length and sum. Here $\Delta(t)$ is a bound on the difference in length between successive "blocks" of an instance of t, and the quantity $B_h(t)$ bounds the length of instances of t that will be relevant to our algorithm. Roughly speaking, the next lemma says that if h satisfies certain conditions, then every sufficiently long instance of a template in a word of the form $h(W)$ must have come from a shorter instance of a parent template in W.

Lemma 3.1.3. *Let $h : X^* \to Y^*$ be a strictly growing affine morphism such that M_h is invertible. Let $W \in X^*$ and $t = [a_0, a_1, \ldots, a_k, d_0, d_1, \ldots, d_{k-2}]$ be an additive k-template. If $h(W)$ contains an instance v of t where $|v| \geq B_h(t)$, then there is an h-parent T of t such that W contains an instance V of T and $|V| < |v|$.*

Proof. Let $v = a_0x_0a_1x_1 \ldots a_{k-1}x_{k-1}a_k$ be a factor of the word $w = h(W)$ and be an instance of

$$t = [a_0, a_1, \ldots, a_k, d_0, d_1, \ldots, d_{k-2}].$$

Suppose that $|x_i| \geq W_h - 1$ for all $0 \leq i \leq k-1$. Then x_i is not an internal factor of the h-image of any letter in X. Therefore, we can write each x_i in the form $x_i = s_i h(X_i) p_{i+1}$ for some $X_i \in X^*$, where s_i is a suffix of $h(A_i)$, and p_{i+1} is a prefix of $h(A_{i+1})$ for some words $A_0, A_1, \ldots, A_k \in X^*$ with $|A_i| \leq 1$. Therefore, the analysis in the proof of Lemma 3.1.2 may be reversed. This means that W contains an instance $V = A_0X_0A_1X_1 \cdots A_{k-1}X_kA_k$ of the template

$$T = [A_0, A_1, \ldots, A_k, D_0, D_1, \ldots, D_{k-2}]$$

where the A_i have h-splits $[p_i, a_i, s_i]$ for all $0 \leq i \leq k$ and

$$D_i = M_h^{-1}(d_i - b_i)$$

for all $0 \leq i \leq k-2$. Note that since h is strictly growing, the instance V of T satisfies $|V| < |v|$.

It now suffices to show that if $|v| \geq B_h(t)$, then for every i we have $|x_i| \geq W_h - 1$. We proceed by contradiction. Suppose that for some i we have $|x_i| \leq W_h - 2$. Let $\zeta_0, \zeta_1, \ldots, \zeta_{k-1}$ be the lengths of the x_i's arranged in non-decreasing order, and so that $\zeta_0 \leq \zeta_1 \leq \cdots \leq \zeta_{k-1}$. Since $|x_i| \leq W_h - 2$, it follows that $\zeta_0 \leq W_h - 2$. Using the definition of $\Delta(t)$, we see that $\zeta_i \leq \zeta_0 + i\Delta(t)$ for $0 \leq i \leq k-1$. Hence,

$$\begin{aligned}
\sum_{i=0}^{k-1} |x_i| = \sum_{i=0}^{k-1} \zeta_i \leq \sum_{i=0}^{k-1} (\zeta_0 + i\Delta(t)) &= \sum_{i=0}^{k-1} \zeta_0 + \sum_{i=0}^{k-1} i\Delta(t) \\
&= k\zeta_0 + \Delta(t) \sum_{i=0}^{k-1} i \\
&= k\zeta_0 + \frac{k(k-1)}{2}\Delta(t) \\
&\leq k(W_h - 2) + \frac{k(k-1)}{2}\Delta(t).
\end{aligned}$$

Therefore, this implies that

$$|v| \leq (k+1) + \sum_{i=0}^{k-1} |x_i| \leq (k+1) + k(W_h - 2) + \frac{k(k-1)}{2}\Delta(t) < B_h(t)$$

which contradicts our assumption that $|v| \geq B_h(t)$. □

Theorem 3.1.4. *Let $h : X^* \to X^*$ be an affine morphism such that every eigenvalue λ of M_h has modulus $|\lambda| > 1$ and let t be an additive k-template. Then the set of h-ancestors of t is finite.*

Proof. Since the eigenvalues of M_h have modulus greater than 1, it follows that M_h is invertible (otherwise $\lambda = 0$ is an eigenvalue), and the eigenvalues of M_h^{-1} are smaller than 1 in absolute value. Let λ_1 and λ_2 be the eigenvalues of M_h^{-1} and let $M_h^{-1} = P^{-1}JP$, where J is the Jordan form of M_h^{-1}. Then we have

$$J = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \quad \text{when } \lambda_1 \neq \lambda_2, \quad \text{or} \quad J = \begin{bmatrix} \lambda_1 & 1 \\ 0 & \lambda_1 \end{bmatrix} \quad \text{when } \lambda_1 = \lambda_2.$$

Hence,

$$M_h^{-n} = P^{-1} \begin{bmatrix} \lambda_1^n & 0 \\ 0 & \lambda_2^n \end{bmatrix} P \quad \text{or} \quad M_h^{-n} = P^{-1} \begin{bmatrix} \lambda_1^n & n\lambda_1^{n-1} \\ 0 & \lambda_1^n \end{bmatrix} P,$$

and so $\sum_{i \geq 0} M_h^{-i}$ converges. Let

$$t = [a_0, a_1, \ldots, a_k, d_0, \ldots, d_{k-2}]$$

be an additive k-template and

$$T = [A_0, A_1, \ldots, A_k, D_0, \ldots, D_{k-2}]$$

be an h-ancestor of t obtained by applying the h-parent relation ℓ times. When $\ell = 1$, then we have $D_i = M_h^{-1}(d_i - b_i)$, where b_i can be a finite number of values as there are only finitely many choices for the h-splits. For $\ell \geq 1$, we find by iteration that D_i has the form

$$D_i = c_\ell M_h^{-\ell} + c_{\ell-1} M_h^{-(\ell-1)} + \cdots + c_1 M_h^{-1} + c_0,$$

where all of the c_j are chosen from a finite set of vectors. We know that $\sum_{i \geq 0} M_h^{-i}$ converges and that $D_i \in \mathbb{Z}^2$, so there are only finitely many possible vectors D_i. Thus, we conclude that there are only finitely many h-ancestors T of t. □

3.2 Description of the Algorithm

In this section, we use the theory from the previous section to prove the following theorem, which describes an algorithm to decide if the words $h(g^\omega(a))$ and $g^\omega(a)$ are additive k-power-free. This algorithm was described briefly by Currie et al. [7], but we provide a more detailed description in this section.

Theorem 3.2.1. *Let $g : X^* \to X^*$ and $h : X^* \to Y^*$ be affine morphisms over the integer alphabets X and Y such that g is prolongable on a. If*

- *h and g are strictly growing,*
- *M_g and M_h are invertible, and*
- *the eigenvalues λ_g of M_g satisfy $|\lambda_g| > 1$.*

then it is possible to decide whether or not $g^\omega(a)$ (and $h(g^\omega(a))$) are additive k-power-free.

The algorithm of Theorem 3.2.1 relies on two main sub-algorithms, which are described separately in the lemmas below.

Lemma 3.2.2 (Finding parents)**.** *Let $f : X^* \to Y^*$ be an affine morphism such that M_f is invertible, and let t be an additive k-template. Then the set of all f-parents of t is effectively computable.*

Proof. Suppose that

$$t = [a_0, a_1, \ldots, a_k, d_0, d_1, \ldots, d_{k-2}],$$

and

$$T = [A_0, A_1, \ldots, A_k, D_0, D_1, \ldots, D_{k-2}]$$

is an f-parent of t. By the definition of f-parent, the A_i have f-splits $[p_i, a_i, s_i]$ such that $d_i = M_h D_i + b_i$, where $b_i = \sigma(s_{i+1}p_{i+2}) - \sigma(s_i p_{i+1})$.

To enumerate all f-parents of t, we first note that there are finitely many choices for the A_i's since X is finite. Once the A_i's are chosen, there are only finitely many f-splits of A_i (with a_i at the centre). We find all such f-splits. Once the f-splits are chosen, the D_i's are completely determined, since we can calculate them as follows:

$$D_i = M_h^{-1}(d_i - b_i).$$

If all D_i have integer entries, then we have a valid h-parent template

$$T = [A_0, A_1, \ldots, A_k, D_0, D_1, \ldots, D_{k-2}]. \qquad \square$$

Lemma 3.2.3 (Enumerating Factors)**.** *Let g be a strictly growing morphism prolongable on a, and n be a non-negative integer. Then the set of all factors of $g^\omega(a)$ of length n is effectively computable.*

Proof. Let $\mathbf{w} = g^\omega(a)$ and F denote the set of factors of length n of $\mathbf{w}$. We know that every finite factor of $\mathbf{w}$ arises by applying g to a a finite number of times. Further, since g is strictly growing, every factor u of length n in $\mathbf{w}$ must be contained in $g(v)$ for some other factor v of $g^\omega(a)$ of length n.

We start with a and apply g repeatedly until we obtain a prefix $g^k(a)$ of $g^\omega(a)$ of length at least n. We add all factors of length n of $g^k(a)$ to F. We go through each element u in F and again apply the morphism g to u. Then, we look at these words $g(u)$ and add any new factors of length n in $g(u)$ to F. We repeat this process, each time checking for any new factors

of length n and adding them to F. When no new factors are available to add to F (i.e., they are already present in F), the process terminates. Note that there are only a finite number of words of length n over a finite alphabet, therefore the process will terminate.

We claim that F contains all factors of $g^\omega(a)$ of length n. Suppose otherwise that there is some factor of $g^\omega(a)$ of length n that is not contained in F and let w be such a factor that appears earliest in $g^\omega(a)$, i.e., starting at the smallest index. Let ℓ be the smallest integer such that w appears in $g^\ell(a)$. Note that $\ell > k$ since all factors of length n of $g^k(a)$ were already added to F in the first step. Then w appears in the g-image of some factor v of length n which appears in $g^{\ell-1}(a)$. Note that if v were in F, then w would also be in F. Therefore, v is not in F. This however, contradicts the assumption that w is a factor of length n of $g^\omega(a)$ that is not in F and appears earliest in $g^\omega(a)$. Therefore, the list F now contains all factors of the word $g^\omega(a)$ of length n. □

Proof of Theorem 3.2.1. We begin by proving that it is possible to decide whether $g^\omega(a)$ is additive k-power-free. We will do this by showing that if $g^\omega(a)$ contains an additive k-power, then we can find it after a finite amount of computation. So suppose that $g^\omega(a)$ contains an additive k-power v. We know by Definition 3.1 that v is an instance of the template

$$t_0 = [\varepsilon, \varepsilon, \ldots, \varepsilon, [0,0]^T, [0,0]^T, \ldots, [0,0]^T].$$

Initial Check: We begin by applying Lemma 3.1.3 with g and $t = t_0$. We have $\Delta(t_0) = 0$ and

$$B_g(t_0) = k + 2 + k \cdot (W_g - 2) + \frac{k \cdot (k-1)\Delta(t_0)}{2} = k + 2 + k \cdot (W_g - 2).$$

Hence if $|v| \geq B_g(t_0)$, then there is a g-parent T of t_0 such that $g^\omega(a)$ contains an instance V of T and $|V| < |v|$. We deal with this case in the **Calculating Ancestors** step. On the other hand, if $|v| < B_g(t_0)$, then v can be found by enumerating all factors of $g^\omega(a)$ of length less than $B_g(t_0)$, as described in the proof of Lemma 3.2.3. If $g^\omega(a)$ contains no additive k-power of length less than $B_g(t_0)$, then we proceed onto calculating ancestors. Otherwise, the algorithm terminates with $g^\omega(a)$ containing an additive k-power.

Calculating Ancestors: We begin by building a list $\mathcal{A}$ of all possible g-parents of t_0 as described in the proof of Lemma 3.2.2. This is the end of the first generation. Now we run through all templates in $\mathcal{A}$, and find all of their g-parents. Any new templates that we find are added to $\mathcal{A}$. This is the end of the second generation. Now we repeat the process. By Theorem 3.1.4, the set of g-ancestors of t_0 is finite and thus the process will terminate with $\mathcal{A}$ containing all of the g-ancestors of t_0. Let j be the number of generations required to find all of the templates in $\mathcal{A}$. It follows by Lemma 3.1.2 that v is a factor of $g^i(V)$ for some instance V of a template T in $\mathcal{A}$ and some $i \in \{0, 1, 2, \ldots, j\}$.

Final Check: Let $M = \max\{\Delta(t) \mid t \in \mathcal{A}\}$. We note that for every template t in $\mathcal{A}$, we have $\Delta(t) \leq M$. Let

$$B_M = k + 2 + k \cdot (W_g - 2) + \frac{k \cdot (k-1)M}{2}.$$

We now repeatedly apply Lemma 3.1.3 with g and $B_g(t) \leq B_M$ and notice that if $g^\omega(a)$ contains an instance of some template in $\mathcal{A}$, then it contains an instance V of some template in $\mathcal{A}$ where $|V| < B_M$. Lemma 3.1.2 assures us that $g^j(V)$ contains an additive k-power. We now enumerate all factors V of $g^\omega(a)$ of length less than B_M, as described in the proof of Lemma 3.2.3. Lastly, we examine the words $g^j(V)$ for each V and check to see if any of them contains an additive k-power.

We now consider words of the form $h(g^\omega(a))$. We want to decide if the word $h(g^\omega(a))$ is additive k-power-free. We show that if $h(g^\omega(a))$ contains an additive k-power, then we can find it with a finite amount of computation. Suppose that $h(g^\omega(a))$ contains an additive k-power v. We know by definition that v is an instance of the template

$$t_0 = [\varepsilon, \varepsilon, \ldots, \varepsilon, [0,0]^T, [0,0]^T, \ldots, [0,0]^T].$$

Initial Check: We begin by applying Lemma 3.1.3 with h and $t = t_0$. We have $\Delta(t_0) = 0$ and

$$B_h(t_0) = k + 2 + k \cdot (W_h - 2) + \frac{k \cdot (k-1)\Delta(t_0)}{2} = k + 2 + k \cdot (W_h - 2).$$

Hence if $|v| \geq B_h(t_0)$, then there is an h-parent T of t_0 such that $g^\omega(a)$ contains an instance V of T and $|V| < |v|$. We deal with this case in the **Calculating Ancestors** step. On the other hand, if $|v| < B_h(t_0)$, then v can be found by enumerating all factors of $h(g^\omega(a))$ of length less than $B_g(t_0)$, as described in the proof of Lemma 3.2.3. If $h(g^\omega(a))$ contains no additive k-power of length less than $B_h(t_0)$, then we proceed onto calculating ancestors. Otherwise, the algorithm terminates with $h(g^\omega(a))$ containing an additive k-power.

Calculating Ancestors: We begin by building a list $\mathcal{A}$ of all possible h-parents of t_0 as described in the proof of Lemma 3.2.2. This is the end of the zeroth generation. Now we run through all templates in $\mathcal{A}$, and find all of their g-parents. Any new templates that we find are added to $\mathcal{A}$. This is the end of the first generation. Now we repeat the process. By Theorem 3.1.4, the set of g-ancestors of any template is finite, and thus the process will terminate with $\mathcal{A}$ containing all of the g-ancestors of all h-parents of t_0. Let j be the number of generations required to find all of the templates in $\mathcal{A}$. It follows by Lemma 3.1.2 that v is a factor of $h(g^i(V))$ for some instance V of a template T in $\mathcal{A}$ and some $i \in \{0, 1, 2, \ldots, j\}$.

Final Check: Let $M = \max\{\Delta(t) \mid t \in \mathcal{A}\}$. We note that for every template t in $\mathcal{A}$, we have $\Delta(t) \leq M$. Let

$$B_M = k + 2 + k \cdot (W_g - 2) + \frac{k \cdot (k-1)M}{2}.$$

Applying Lemma 3.1.3 repeatedly, we see that if $g^\omega(a)$ contains an instance V of some template in $\mathcal{A}$, then it contains one where $|V| < B_M$. Lemma 3.1.2 assures us that $h(g^j(V))$ contains an additive k-power v. We now calculate every factor V of $g^\omega(a)$ of length less than B_M. Lastly, we examine the words $h(g^j(V))$ for each V and check to see if any of them contains an additive k-power. □

Chapter 4

Applications

In this section, the algorithm described in Chapter 3 is used to prove some new results, and to reprove some previously known results. The new constructions were found using a backtracking algorithm, i.e., a depth-first search through the tree of all words satisfying a given property.

Theorem 4.0.1 (Currie, Mol, Rampersad, Shallit [7]). *Let $f : \{0,1,2\}^* \to \{0,1,2\}^*$ be the morphism defined by $f(0) = 001$, $f(1) = 012$, $f(2) = 212$. The word $f^\omega(0)$ is additive 4-power-free.*

Proof. We verify that $f^\omega(0)$ is additive 4-power-free, by running the algorithm described in Section 3.2. By definition, an additive 4-power v in $f^\omega(0)$ is an instance of the template

$$t_0 = [\varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon, [0,0]^T, [0,0]^T, [0,0]^T].$$

For the initial check, we enumerate all factors of $f^\omega(0)$ of length $B_f(t_0) = 10$, and verify that none of them contains an additive 4-power. Next, we find the set $\mathcal{A}$ of all f-ancestors of t_0. We find a total of 17056 ancestors after 3 generations. Proceeding onto the final check, we find that for every template t in $\mathcal{A}$, we have $\Delta(t) \leq 2$. Therefore, we have

$$B_M = k + 2 + k \cdot (W_f - 2) + \frac{k(k-1)}{2} \cdot M = 4 + 2 + 4 \cdot 1 + \frac{4 \cdot 3}{2} \cdot 2 = 22.$$

We now enumerate all factors V of $f^\omega(0)$ of length 21. We examine the word $f^3(V)$ for each such factor V and check that none of them contain an additive 4-power. Therefore, we conclude that $f^\omega(0)$ is additive 4-power-free. □

Theorem 4.0.2 (Currie, Mol, Rampersad, Shallit [7]). *Let $f : \{0,1,2\}^* \to \{0,1,2\}^*$ be the morphism defined by $f(0) = 001$, $f(1) = 012$, $f(2) = 212$ and $g : \{0,1,2\}^* \to \{0,1\}^*$ be the*

morphism defined by

$$g(0) = 0001001110010001100011,$$
$$g(1) = 000100111001101100011,$$
$$g(2) = 011100111001101100011.$$

The word $g(f^\omega(0))$ is additive 4-power-free and 3^+-power-free. This is best possible in the sense that there are only finitely many additive 4-power-free and ordinary 3-power-free binary words with the longest being length 39.

Proof. First, we show that $g(f^\omega(0))$ is additive 4-power-free. By definition, an additive 4-power v in $g(f^\omega(0))$ is an instance of the template

$$t_0 = [\varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon, [0,0]^T, [0,0]^T, [0,0]^T].$$

For the initial check, we enumerate all factors of $g(f^\omega(0))$ of length $B_g(t_0) = 85$, and verify that none of them contains an additive 4-power. Next, we find the set $\mathcal{A}$ of all f-ancestors of all g-parents of t_0. We find a total of 17104 ancestors after 1 generation. Proceeding onto the final check, we find $M = \max\{\Delta(t) \mid t \in \mathcal{A}\} = 2$. Therefore, we have

$$B_M = k + 2 + k \cdot (W_f - 2) + \frac{k(k-1)}{2} \cdot M = 4 + 2 + 4 \cdot 1 + \frac{4 \cdot 3}{2} \cdot 2 = 22.$$

We now enumerate all factors V of $f^\omega(0)$ of length 21. We examine the words $g(f^1(V))$ for each factor V and check that none of them contain an additive 4-power. Therefore, we conclude that $g(f^\omega(0))$ is additive 4-power-free.

To check that the word $g(f^\omega(0))$ is 3^+-power-free, we enter the following commands into `Walnut`:

```
morphism g "0->0001001110010001100011 1->000100111001101100011 2->011100111001101100011":
morphism f "0->001 1-> 012 2->212":
promote aut f:
image SA g aut:
eval containsthreeplus "?msd_3 Ei, n (n>=1) & At (t<=2*n) => SA[i+t]=SA[i+n+t]":
```

The predicate `containsthreeplus` checks to see if the word contains a 3^+-power , and `Walnut` returns `FALSE`. □

Theorem 4.0.3. *The word $h^\omega(0)$ produced by the morphism $h : \{0,1\}^* \to \{0,1\}^*$ where*

$$h(0) = 00001,$$
$$h(1) = 01101,$$

is additive 5-power-free and rich. This is best possible in the sense that there are only finitely many additive 4-power-free binary rich words.

Proof. First we verify that $h^\omega(\mathtt{0})$ is additive 5-power-free, by running the algorithm described in Section 3.2. By definition, an additive 5-power v in $h^\omega(\mathtt{0})$ is an instance of the template

$$t_0 = [\varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon, [0,0]^T, [0,0]^T, [0,0]^T, [0,0]^T].$$

For the initial check, we enumerate all factors of $h^\omega(\mathtt{0})$ of length $B_h(t_0) = 22$, and verify that none of them contains an additive 5-power. Next, we find the set $\mathcal{A}$ of all h-ancestors of t_0. We find a total of 15625 ancestors after 1 generation. Proceeding onto the final check, we find that for every template t in $\mathcal{A}$, we have $\Delta(t) \leq 2$. Therefore, we have

$$B_M = k + 2 + k \cdot (W_h - 2) + \frac{k(k-1)}{2} \cdot M = 5 + 2 + 5 \cdot 3 + \frac{5 \cdot 4}{2} \cdot 2 = 42.$$

We now enumerate all factors V of $h^\omega(\mathtt{0})$ of length 41. We examine the word $h^1(V)$ for each such factor V and check that none of them contain an additive 5-power. Therefore, we conclude that $h^\omega(\mathtt{0})$ is additive 5-power-free.

We will now prove that this word is rich. By Theorem 2.6.2, it suffices to show that every finite prefix of $h^\omega(\mathtt{0})$ has a unioccurrent palindromic suffix. We show this using `Walnut`. We adapt the `Walnut` commands used by Baranwal and Shallit [1] and Schaeffer and Shallit [27] in proving that similar infinite words are rich.

```
morphism h "0->00001 1->01101":
# creates the desired morphism h.

promote H h:
# creates an automaton based on the morphism h.

def FactorEq "?msd_5 Ak (k<n)=>(H[i+k]=H[j+k])":
def Occurs "?msd_5 (m<= n) & ( Ek (k+m<=n) & $FactorEq(i,j+k,m))":
def Palindrome "?msd_5 Aj,k ((k<n) & (j+k+1=n)) => (H[i+k]=H[i+j])":

def HisRich  "?msd_5 An Ej $Palindrome(j,n-j) & ~$Occurs(j,0,n-j,n-1)":
```

The predicate `FactorEq` takes 3 parameters i, j, n and returns true if the length-n factors of $h^\omega(\mathtt{0})$ starting at indices i and j are equal. The predicate `Occurs` takes 4 parameters i, j, m, n and returns true if the length-m factor of the word $h^\omega(\mathtt{0})$ starting at index i occurs in the length-n factor starting at index j. The predicate `Palindrome` takes 2 parameters i, n and returns true if the length-n factor of $h^\omega(\mathtt{0})$ starting at index i is a palindrome. The predicate `HisRich` returns true if all the finite prefixes of $h^\omega(\mathtt{0})$ have a unioccurrent palindromic suffix. When we run all of these commands, `Walnut` returns `TRUE`.

Finally, to see that this is best possible, we use a backtracking algorithm to find that the longest additive 4-power-free binary rich word has length 2411. This was a long backtracking search, made possible by the use of the data structure `eerTree` of Rubinchik and Shur [26]. This data structure greatly increased the efficiency of searching through the tree of binary rich words. □

Theorem 4.0.4 (Dekking [12])**.** *The word $h^\omega(\mathtt{0})$ produced by the morphism $h : \{\mathtt{0},\mathtt{1}\}^* \to \{\mathtt{0},\mathtt{1}\}^*$ where*

$$h(\mathtt{0}) = \mathtt{0001},$$
$$h(\mathtt{1}) = \mathtt{011},$$

is additive 4-power-free.

Proof. By definition, an additive 4-power v in $h^\omega(\mathtt{0})$ is an instance of the template

$$t_0 = [\varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon, [0,0]^T, [0,0]^T, [0,0]^T].$$

For the initial check, we enumerate all factors of $h^\omega(\mathtt{0})$ of length $B_h(t_0) = 14$, and verify that none of them contains an additive 4-power. Next, we find the set $\mathcal{A}$ of all h-ancestors of t_0. We find a total of 3123 ancestors after 1 generation. Proceeding onto the final check, we find that for every template t in $\mathcal{A}$, we have $\Delta(t) \leq 2$. Therefore, we have

$$B_M = k + 2 + k \cdot (W_h - 2) + \frac{k(k-1)}{2} \cdot M = 4 + 2 + 4 \cdot 2 + \frac{4 \cdot 3}{2} \cdot 2 = 26.$$

We now enumerate all factors V of $h^\omega(\mathtt{0})$ of length 25. We examine the word $h^1(V)$ for each such factor V and check that none of them contain an additive 4-power. Therefore, we conclude that $h^\omega(\mathtt{0})$ is additive 4-power-free. □

Theorem 4.0.5. *The word $g^\omega(\mathtt{0})$ produced by the morphism $g : \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \to \{\mathtt{0},\mathtt{1},\mathtt{2}\}^*$ defined by*

$$g(\mathtt{0}) = \mathtt{01100},$$
$$g(\mathtt{1}) = \mathtt{0110121},$$
$$g(\mathtt{2}) = \mathtt{011012221},$$

is additive 4-power-free.

Proof. By definition, an additive 4-power v in $g^\omega(\mathtt{0})$ is an instance of the template

$$t_0 = [\varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon, [0,0]^T, [0,0]^T, [0,0]^T].$$

For the initial check, we enumerate all factors of $g^\omega(\mathtt{0})$ of length $B_g(t_0) = 34$, and verify that none of them contains an additive 4-power. Next, we find the set $\mathcal{A}$ of all g-ancestors of t_0. We find a total of 17404 ancestors after 1 generation. Proceeding onto the final check, we find that for every template t in $\mathcal{A}$, we have $\Delta(t) \leq 2$. Therefore, we have

$$B_M = k + 2 + k \cdot (W_h - 2) + \frac{k(k-1)}{2} \cdot M = 4 + 2 + 4 \cdot 2 + \frac{4 \cdot 3}{2} \cdot 2 = 46.$$

We now enumerate all factors V of $g^\omega(\mathtt{0})$ of length 45. We examine the word $g^1(V)$ for each such factor V and check that none of them contain an additive 4-power. Therefore, we conclude that $g^\omega(\mathtt{0})$ is additive 4-power-free. □

Chapter 5

Open Problems

This thesis was focused on an algorithm for deciding if a word is additive k-power-free. We now discuss several open problems related to additive k-power-free words.

Problem 1: In Theorem 4.0.4, we reprove that the infinite binary word discovered by Dekking is additive 4-power-free. The frequency of zeros in this word converges to $\approx 55\%$. We attempted to find a construction of an infinite additive 4-power-free binary word with a higher frequency of 0's, but were unsuccessful. The open problem is stated as follows: What is the maximum possible frequency of `0`'s in an infinite additive 4-power-free binary word?

Problem 2: It is known that over many alphabets of size 3 such as $\{0,3,8\}$, $\{0,2,7\}$, and $\{0,1,8\}$ there exist infinite additive cube-free words [24]. There are various alphabets of size 3 where it is unknown if such words exists. It is conjectured that for every alphabet $A = \{0, i, j\}$ such that i and j are co-prime and $j \geq 6$, there exists an infinite additive-cube-free word on the alphabet A [24]. The cases $\{0,1,2\}$, $\{0,1,3\}$, $\{0,1,4\}$, and $\{0,2,5\}$ remain open. It is also unknown if an infinite additive cube-free word exists over the 4 letter alphabet $\{0,1,2,3\}$. Rao constructed a word over $\{0,1,2,3\}$ of length 140000 that is additive cube-free.

Problem 3: In Theorem 4.0.3, we proved that the infinite binary word $g^\omega(0)$ is additive 5-power-free. Using `Walnut`, we also proved it was rich. Through backtracking, we found a morphism $g : \{0,1,2\}^* \to \{0,1,2\}^*$ defined by

$$\begin{aligned} g(0) &= 01100, \\ g(1) &= 0110121, \\ g(2) &= 011012221. \end{aligned}$$

We proved the word $g^\omega(0)$ is additive 4-power-free in Theorem 4.0.5. We conjecture that this word is also rich. If our conjecture is correct, then this would be considered best possible in the sense that there are only finitely many additive cube-free ternary rich words.

Problem 4: Over the course of writing this thesis, the word $h^\omega(\mathtt{0})$ produced by the morphism $h : \{\mathtt{0},\mathtt{1},\mathtt{2},\mathtt{3}\}^* \to \{\mathtt{0},\mathtt{1},\mathtt{2},\mathtt{3}\}^*$ where

$$\begin{aligned} h(\mathtt{0}) &= \mathtt{01210}, \\ h(\mathtt{1}) &= \mathtt{20103}, \\ h(\mathtt{2}) &= \mathtt{20123}, \\ h(\mathtt{3}) &= \mathtt{13231}, \end{aligned}$$

was discovered through backtracking, and we conjecture that it is additive 5-power-free and square-free. We attempted to run our algorithm to prove the additive 5-power-freeness, but at the calculating ancestors step, we found 957318 ancestors of t_0, and it took 36 generations. This large number of generations required meant that completing the final check was not feasible. We were able to prove that $h^\omega(\mathtt{0})$ is square-free using `Walnut`. Here is the `Walnut` code for proving square-freeness:

```
morphism h "0->01210 1->20103 2->20123 3->13231":
promote H h:

eval HContainsSquares "?msd_5 Ei,n n>=1 & As s<n => H[i+s]=H[i+s+n]":
# returns FALSE.
```

Bibliography

[1] A. R. Baranwal and J. Shallit, "Repetitions in infinite palindrome-rich words," in *Combinatorics on Words*. Springer International Publishing, 2019, pp. 93–105.

[2] J. Berstel, "Axel Thue's Papers on Repetitions in Words: A Translation," in *Monographies du LaCIM*, vol. 11, LaCIM, 1992, pp. 65–80.

[3] S. Brlek, S. Hamel, M. Nivat, and C. Reutenauer, "On the palindromic complexity of infinite words," *International Journal of Foundations of Computer Science*, vol. 15, pp. 293–306, 2004.

[4] A. Carpi, "On dejean's conjecture over large alphabets," *Theoretical Computer Science*, vol. 385, pp. 137–151, 2007.

[5] J. Cassaigne, J. D. Currie, L. Schaeffer, and J. Shallit, "Avoiding three consecutive blocks of the same size and same sum," *Journal of the ACM*, vol. 61, pp. 1–17, 2011.

[6] J. D. Currie, L. Mol, and N. Rampersad, "The repetition threshold for binary rich words," *Discrete Mathematics & Theoretical Computer Science*, vol. 22, 2020.

[7] J. D. Currie, L. Mol, N. Rampersad, and J. Shallit, "Extending dekking's construction of an infinite binary word avoiding abelian 4-powers," pp. 1–8, 2021. arXiv: `2111.07857`.

[8] J. D. Currie and N. Rampersad, "A proof of dejean's conjecture," *Mathematics of Computation*, vol. 80, 2009.

[9] J. D. Currie and N. Rampersad, "Dejean's conjecture holds for $N \geq 27$," *RAIRO - Theoretical Informatics and Applications*, vol. 43, pp. 775–778, 2009.

[10] J. D. Currie and N. Rampersad, "Fixed points avoiding abelian k-powers," *Journal of Combinatorial Theory, Series A*, vol. 119, pp. 942–948, 2012.

[11] F. Dejean, "Surunth´eor'emedethue," *Journal of Combinatorial Theory, Series A*, pp. 90–99, 1972.

[12] F. Dekking, "Strongly non-repetitive sequences and progression-free sets," *Journal of Combinatorial Theory, Series A*, vol. 27, pp. 181–185, 1979.

[13] X. Droubay, J. Justin, and G. Pirillo, "Episturmian words and some constructions of De Luca and Rauzy," *Theoretical Computer Science*, vol. 255, pp. 539–553, 2001.

[14] P. Erdős, "Some unsolved problems," *Michigan Mathematical Journal*, vol. 4, pp. 291–300, 1957.

[15] A. A. Evdokimov, "Strongly asymmetric sequences generated by a finite number of symbols," *Dokl. Akad. Nauk SSSR*, vol. 179, pp. 1268–1271, 1968.

[16] A. Glen, J. Justin, S. Widmer, and L. Q. Zamboni, "Palindromic richness," *European Journal of Combinatorics*, vol. 30, pp. 510–531, 2009.

[17] V. Keränen, "Abelian squares are avoidable on 4 letters," in *Automata, Languages and Programming*, W. Kuich, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 41–52.

[18] F. Lietard and M. Rosenfeld, "Avoidability of additive cubes over alphabets of four numbers," in 2020, pp. 192–206.

[19] H. Mousavi, "Automatic theorem proving in walnut," *preprint*, vol. 2, pp. 1–34, 2016. [Online]. Available: `http://arxiv.org/abs/1603.06017`.

[20] J. M. Ollagnier, "Proof of dejean's conjecture for alphabets with 5, 6, 7, 8, 9, 10 and 11 letters," *Theoretical Computer Science*, vol. 95, pp. 187–205, 1992.

[21] E. Pelantová and Š. Starosta, "Languages invariant under more symmetries: Overlapping factors versus palindromic richness," *Discrete Mathematics*, vol. 313, pp. 2432–2445, 2013.

[22] P. A. B. Pleasants, "Non-repetitive sequences," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 68, pp. 267–274, 1970.

[23] M. Rao, "Last cases of dejean's conjecture," *Theoretical Computer Science*, vol. 412, pp. 3010–3018, 2011.

[24] M. Rao, "On some generalizations of abelian power avoidability," *Theoretical Computer Science*, vol. 601, pp. 39–46, 2015.

[25] M. Rao and M. Rosenfeld, "Avoiding two consecutive blocks of same size and same sum over $\mathbb{Z}^2$," *SIAM Journal on Discrete Mathematics*, vol. 32, pp. 2381–2397, 2018.

[26] M. Rubinchik and A. M. Shur, "Eertree: An efficient data structure for processing palindromes in strings," *European Journal of Combinatorics*, vol. 2, 2015.

[27] L. Schaeffer and J. Shallit, "Closed, palindromic, rich, privileged, trapezoidal, and balanced words in automatic sequences," *Electronic Journal of Combinatorics*, pp. 1–19, 2016.

[28] J. Shallit, *The Logical Approach To Automatic Sequences: Exploring Combinatorics on Words with Walnut*, 1st ed. Cambridge: Cambridge University Press, 2022.

[29] J. Shallit. "Palindromes, squares, and other repitions in natural languages." (), [Online]. Available: `https://cs.uwaterloo.ca/~shallit/repetitions.html`.

[30] A. Thue, "Über unendliche zeichenreihen," *Norske Vid. Selsk. Skr. I. Mat. Nat. Kl.*, vol. No. 7, pp. 1–22, 1906.

[31] A. Thue, "Über die gegenseitige lage gleicher teile gewisser zeichenreihen," *Videnskaps-selskapet: Skrifter; 1912,1*, pp. 1–67, 1912.