Temple University
Doctoral Dissertation
Submitted to the Graduate Board

Title of Dissertation: (Please type)	Extensions And Applications of the Goulden-Jackson Method To Self-Avoiding Walks, Square and Cube Free Words, Probability, Entropy, Cyclic Words And Related Sequences
Author: (Please type)	Ms. Anne E. Edlin
Date of Defense: (Please type)	Monday, July 17, 2000

Dissertation Examining Committee: (please type)	Read and Approved By: (Signatures)
Professor Doron Zeilberger Dissertation Advisory Committee Chairperson	[illegible]
Professor Shiferaw Berhanu	S. Berhanu
Professor Boris Datskovsky	[illegible]
Professor Dominique Foata	[illegible]
Professor Doron Zeilberger Examining Committee Chairperson	[illegible] If Member of the Dissertation Examining Committee

Date Submitted to Graduate Board: 8-4-00

Accepted by the Graduate Board of Temple University in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Date 9/11/00 [illegible]
(Dean of the Graduate School)

EXTENSIONS AND APPLICATIONS OF THE GOULDEN-JACKSON METHOD TO SELF-AVOIDING WALKS, SQUARE AND CUBE FREE WORDS, PROBABILITY, ENTROPY, CYCLIC WORDS AND RELATED SEQUENCES

A Dissertation
Submitted to
the Temple University Graduate Board

in Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

by
Anne E. Edlin
August, 2000

ABSTRACT

EXTENSIONS AND APPLICATIONS OF THE GOULDEN-JACKSON METHOD TO SELF-AVOIDING WALKS, SQUARE AND CUBE FREE WORDS, PROBABILITY, ENTROPY, CYCLIC WORDS AND RELATED SEQUENCES

Anne E. Edlin

DOCTOR OF PHILOSOPHY

Temple University, August, 2000

Dr. Doron Zeilberger, Chair

This dissertation will explore the combinatorial and statistical properties of various classes of words. The words studied will include both English words and more abstract mathematical objects including square-free words, cube-free words and self-avoiding walks. In addition to these linear words, we will also explore cyclic words. The study of words has applications to genetic theory and Crystallography.

The linear words will be analyzed from the perspectives of limiting behavior, entropy and generating functions. Noonan and Zeilberger's implementations of the Goulden-Jackson Method will be expanded to analyze both the enumeration of linear words, and the probability of their occurrence. The limiting behavior of these objects is of great interest at this time. As recently as August 1998 Ekhad and Zeilberger improved Brinkhuis and Brandenburg's lower bounds for the 'connective constant' for ternary square-free words. This dissertation will adapt their work to the situation of binary cube-free words.

Self-avoiding walks will be explored on the rectangular and honeycomb lattice. The inter relation between these walks and other avoidance patterns will be explored and bounds will be obtained for these cases.

In the case of cyclic words we will adapt the Goulden-Jackson method to this situation and expand on the results of Burstein and Wilf regarding cyclic words that avoid long constant blocks.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Doron Zeilberger who inspired the interest to start this research and who gave me all the motivation and nudging I needed to finish it.

I would like to thank Robert Hallowell for his support.

Thanks too to Drs. Alu Srinivasan, Jack Schiller, Bashar Hanna, Eric Grinberg, Boris Datskovsky and Shiferaw Berhanu for their support and encouragement throughout my time at Temple.

Finally I would like to thank Dominique Foata for his help.

To Sister Cecelia,

FCJ Convent, Jersey,

her belief in a ten-year-old girl helped to give me the strength to

survive the bruises my confidence got in later years.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

SEQUENCES OF WORDS

1.1 Words

Words surround us, not just in the literal sense of the words on billboards, road signs, cereal packets, in books and magazines, but also in a more abstract sense. Our DNA is defined by a word over the language of nucleotides. The bar codes on our groceries are words in the computer language of zeroes and ones. Further, in mathematics there are words that avoid certain patterns, such as repeating blocks, and some that have applications in such areas as the study of linear polymer molecules in chemical physics.

In order to explore the behavior of such a wide range of words, we must first introduce a format by which words are defined and some basic terminology that will be used throughout this work. My choice of notation is based on my frequent reliance on Maple to perform calculations.

Notation 1.1 *Let **V** be the alphabet over which our language is defined.*

E.g. in English $V = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$. In computing $V = \{0, 1\}$.

Definition 1.1 *A* **word**, *w, over the alphabet V is an ordered sequence of letters from V, $w = [w_1, w_2, ..., w_n]$ where $w_i \in V$ for $1 \leq i \leq n$.*

E.g. the English word "alphabet" becomes $[a, l, p, h, a, b, e, t]$.

Notation 1.2 ***V^**** *is the set of all possible words over the alphabet V.*

Definition 1.2 *A* **factor** *of w is any of the $\binom{n+1}{2}$ possible sub-sequences $[w_i, w_{i+1}, \dots, w_j]$ where $1 \leq i \leq j \leq n$.*

Thus $[a, l, p], [h, a]$ and $[b, e, t]$ are all factors of $[a, l, p, h, a, b, e, t]$.

Notation 1.3 *The* **empty word** *is considered to be a factor of all words and belongs to V^* for every V. It will be denoted* [ ].

Definition 1.3 *The* **length** *of a word $l(w)$ is the number of letters in the word, counting multiplicity.*

E.g. $l([a, l, p, h, a, b, e, t]) = 8$. Note $l([\,]) = 0$.

Notation 1.4 *wu represents the juxtaposition of the two words w and u.*

Notation 1.5 *If W and U are sets of words and u a word then $Wu = \{wu : w \in W\}$ and $WU = \{wu : w \in W.u \in U\}$.*

One of the main areas of research into words is their limiting behavior. That is if a_n is the number of words in our language of length n we want to find $\mu := \lim_{n\to\infty} {a_n}^{1/n}$, if it exists.

Clearly if no constraint is put upon our choice of words and if k is the number of letters in our alphabet V then $a_n = k^n$ and hence $\mu = k$. This leads us to believe our quest for limits will not prove fruitless.

Often it is useful to use the model $a_n = n^\theta \mu^n$. Zinn's method can be used to obtain good approximations of this type.

1.2 Avoiding the Bad

Most of the sequences $\{a_n\}$ considered in this text are ones whose words avoid specific factors. We consider the factors we wish to avoid as the bad words (or mistakes), and the set of all such words will be denoted B. The set of all bad words up to length m will be denoted B_m. When the set of bad words is infinite we often consider the **memory m** case by using B_m as our set of mistakes.

As an example consider the case of binary square-free words, that is words over a two letter alphabet that avoid any non-trivial factor being repeated directly after itself. In this case $B_4 = \{[0,0],[1,1],[0,1,0,1],[1,0,1,0]\}$.

It should be noted that B and B_m are always minimal in the sense that no member of B (or B_m) is a factor of any other member of B(or B_m). In the above example note $[1, 1, 1, 1]$ is omitted from B_4 because it contains $[1, 1]$ as a factor.

In fact in this case $B = B_4$ and $a_n = [1, 2, 2, 2, 0, 0, 0, \ldots]$, which is not a very interesting sequence. The more interesting case of ternary square-free words is discussed by Noonan and Zeilberger [NZ99].

1.3 Subadditive Sequences

Many of the sequences we will be discussing are sub-multiplicative. That is that $a_{n+m} \leq a_n a_m$. In sequences where $a_n \neq 0$ we have that $log(a_{n+m}) \leq log(a_n) + log(a_m)$ which shows that the sequence $\{log(a_n)\}$ is subadditive $(c_{n+m} \leq c_n + c_m)$. This fact can be used to show that the μ exists and is in fact the $\inf a_n^{1/n}$

Lemma 1.1 *Let $\{c_n\}$ be a subadditive sequence of real numbers. Then the* $\lim_{n\to\infty} \frac{c_n}{n}$ *exists and equals* $\inf_{n\geq 1} \frac{c_n}{n}$.

Proof of Lemma: Let $C_k = max_{1\leq r\leq k} c_r$. Then for any given n we can find j such that $n = jk + r$ with $1 \leq r \leq k$.

Using the subadditivity of $\{c_n\}$ we obtain

$$c_n \leq jc_k + c_r \leq \frac{n}{k} c_k + C_k$$

Then we divide both sides by n and take the $\limsup_{n\to\infty}$ to obtain

$$\limsup_{n\to\infty} \frac{c_n}{n} \le \limsup_{n\to\infty}(\frac{c_k}{k} + \frac{C_k}{n}) \le \frac{c_k}{k}$$

Finally we take the $\liminf_{k\to\infty}$ and obtain that the $\limsup \le \liminf$ thus proving the limit exists.

As the limit exists it equals the lim sup and so as this is less than $\frac{c_k}{k}$ for all k we obtain

$$\lim_{n\to\infty} \frac{c_n}{n} = \inf_{n\geq 1} \frac{c_n}{n}$$

$\square$

Theorem 1.1 *If $\{a_n\}$ is a sequence of positive terms for which $a_{n+m} \le a_n a_m$ then $\mu = \lim_{n\to\infty} a_n^{\frac{1}{n}}$ exists. Further $\mu \le a_n^{\frac{1}{n}}$.*

Proof: As discussed above $a_{n+m} \le a_n a_m$ implies that the sequence $\{\log a_n\}$ is subadditive. This means $\log \mu = \lim_{n\to\infty} \frac{\log a_n}{n} = \lim_{n\to\infty} \log a_n^{\frac{1}{n}}$ exists and further $\log \mu = \inf_{n\geq 1} \frac{\log a_n}{n} = \inf_{n\geq 1} \log a_n^{\frac{1}{n}} \le \log a_n^{\frac{1}{n}}$ for all n. This gives the required results.

$\square$

1.4 The Naive Approach

At this point we are only considering linear sequences. Later we will investigate the case of cyclic sequences.

For any given word we define its $m-$weight at follows:

$$W_m(w) = s^n \prod_{v \in V} x[v],$$

where v extends over all the factors of w for which $l(v) \leq m$.

For example the 3-weight of the word apple would be

$s^5 x[a]x[p]^2 x[l]x[e]x[a,p]x[p,p]x[p,l]x[l,e]x[a,p,p]x[p,p,l]x[p,l,e]$

Then if $x[v] = 1$ the coefficient of s^n will give us the number of words of length n and if $x[v] =$ Probability(v) then we obtain the probability of finding a word of length n. The first case is considered in most of the following chapters. The second case is discussed in Chapter 11.

This means are goal becomes to find the generating function that has all words of length n (or often just the number of them) that meet our criteria as the coefficient of s^n. When we consider probability we will be using the 2-weight of the words, but in general we will only consider the number of words of length n and not how they are made up.

One method for doing this is to use a matrix, A, to analyze the interaction between all possible blocks of length m then by taking $(1 - A)^{-1}$ and adding all the resulting entries we obtain a generating function for all words over the chosen alphabet. We then set any blocks that are disallowed equal to zero and obtain the generating function for the desired set of words.

We call this method the Naive Approach because it produces all possible words without taking into account the bad words until the very end. For example if we were to take the English alphabet and look for all words that did not contain any bad "4-letter" words we would need a matrix that was 26^4 by 26^4, and worse yet need to find the inverse of such a matrix, a very slow task, even for a computer. Thus this approach is only useful in very small cases and as a check for our clever techniques, like the Goulden-Jackson Method.

1.5 The Goulden-Jackson Method

One method used throughout this dissertation is the Goulden-Jackson Cluster Method [GJ79]. This method can be used to find the generating function $f(s) = \sum_{n=0}^{\infty} a_n s^n$ for words that avoid certain mistakes. In many cases we can not find $f(s)$ explicitly as we are looking at infinite sets of mistakes, but we can obtain $f_m(s)$ (the memory m scenario) which gives correct values for a_n when $n \leq m$ and good over estimates for $n > m$, as it only considers mistakes up to length m.

We will discuss briefly this method, for a more in depth explanation and some applications see [GJ79].

If $L(B)$ represents the set of words that avoid any of B as factors, $Bad(w)$ denotes the set of factors of w that belong to B and $N(w) = |Bad(w)|$ then

$$\begin{aligned}
f(s) &= \sum_{w\in L(B)} weight(w) \\
&= \sum_{w\in V^*} weight(w) 0^{N(w)} \\
&= \sum_{w\in V^*} weight(w)(1+(-1))^{N(w)} \\
&= \sum_{w\in V^*} weight(w) \sum_{t=0}^{N(w)} \binom{N(w)}{t} (-1)^t \\
&= \sum_{w\in V^*} weight(w) \sum_{t=0}^{M(w)} (\text{number of subsets of } Bad(w) \text{ of size } t)(-1)^t \\
&= \sum_{w\in V^*} weight(w) \sum_{S\subseteq Bad(w)} (-1)^{|S|} \\
&= \sum_{w\in V^*} s^{l(w)} \sum_{S\subseteq Bad(w)} (-1)^{|S|}, \qquad (1.1)
\end{aligned}$$

where $0^0 = 1$ and we are only considering the number of words, not their m-weight.

We now mark the words by the overlapping mistakes in them and call the set of all marked words $M = \sum_{w\in V^*} \sum_{S\subseteq Bad(w)} (w, S)$.

For example if $B = \{[a,p],[p,p,l],[p,l,e]\}$ and we take the word $[a,p,p,l,e]$ we obtain the following marked words: $([a,p,p,l,e];), ([a,p,p,l,e];[1,2])$, $([a,p,p,l,e];[2,4]), ([a,p,p,l,e];[3,5]), ([a,p,p,l,e];[1,2],[2,4])$, $([a,p,p,l,e];[2,4],[3,5]), ([a,p,p,l,e];[1,2],[2,4],[3,5])$. Where the indices refer to the position of the mistakes and are ordered so that the end of each marked block exceeds the end of the previous block. The minimality of the set of bad words assures us that the start of each block also exceeds the start of the previous block. These are called **clusters**.

Here is one of the clusters:

```
a p p l e
a p
  p p l
    p l e
```

Then equation 1.1 becomes

$$f(s) = \sum_{(w,S)\in M} s^{l(w)}(-1)^{|S|}$$

We now note that every marked word is either the empty word, ends in a cluster or ends in a letter that is not part of a cluster so letting C represent the clusters, and recalling M are our marked words and V our alphabet of size k we have

$$M = \{\text{empty word}\} \cup MV \cup MC.$$

Taking weights in this equation we obtain

$$f(s) = 1 + f(s)ks + f(s)weight(Clusters)$$

Which yields

$$f(s) = \frac{1}{1 - ks - weight(Clusters)} \tag{1.2}$$

So now we need only find the weight of the clusters.

Firstly note that we can divide up the clusters by the last mistake in them so that if we let $W(C[v])$ represent the weight of the clusters ending in v then

$$weight(Clusters) = \sum_{v\in B} W(C[v])$$

Now when two mistakes overlap the additional contribution to the cluster is $(u : v) = s^{(\text{number of new letters in } v \text{ not in } u\ )}$. For example in the cluster above

$([a,p] : [p,p,l]) = s^2$ and $([p,p,l] : [p,l,e]) = s$. We must in fact consider every possible overlap i.e. $([1,1,1] : [1,1,1]) = s^2 + s$, and if two words do not overlap $(u : v) = 0$. Then we have

$$W(C[v]) = -weight(v) - \sum_{u \in B} (u : v) W(C[u]) \tag{1.3}$$

Theorem 1.2 *(The Goulden Jackson Method) Given a set of bad words B the number of words of length n that avoid these words as factors is given by the coefficient of s^n in*

$$f(s) = \frac{1}{1 - ks - weight(Clusters)}$$

where

$$weight(Clusters) = \sum_{v \in B} W(C[v])$$

and

$$W(C[v]) = -weight(v) - \sum_{u \in B} (u : v) W(C[u])$$

.

An Example of Applying the Goulden-Jackson Method

Let our alphabet be $\{a, b\}$ and our bad words be $\{[a,a],[b,b]\}$.

Then

$$W(C[a,a]) = -s^2 - sW(C[a,a])$$

$$W(C[b,b]) = -s^2 - sW(C[b,b]),$$

and solving we obtain

$$W(C[a,a]) = \frac{-s^2}{1+s}$$
$$W(C[b,b]) = \frac{-s^2}{1+s}$$

So that

$$\begin{aligned} f(s) &= \frac{1}{1-2s+\frac{2s^2}{1+s}} \\ &= \frac{1+s}{1-s} \\ &= 1+2s+2s^2+2s^3+\ldots \end{aligned}$$

as would be expected.

CHAPTER 2

BINARY CUBE-FREE WORDS

2.1 Introduction

Let us now look closely at a specific non-trivial example.

Definition 2.1 *A word is* **cube-free** *if it contains no factors of the form xxx, where x is any non-empty word.*

E.g. The cube-free words of length 3 over the alphabet $\{a, b\}$ are

$\{[a, a, b], [a, b, a], [a, b, b], [b, a, a], [b, a, b], [b, b, a]\}$

My Maple package Cubefree (available from **http://www.math.temple.edu/~anne/cubefree.html**) can be used to derive cube-free words over any given alphabet up to the required length. The number of binary cube-free words of length at most n for $0 \leq n \leq 47$ are given below.

These results were obtained by applying the Goulden-Jackson Method with all cubes of length at most 45 as the input mistakes.

2.1.1 The Sequence of Binary Cube-Free Words of length up to 47

1, 2, 4, 6, 10, 16, 24, 36, 56, 80, 118, 174, 254, 378, 554, 802, 1168, 1716, 2502, 3650, 5324, 7754, 11320, 16502, 24054, 35058, 51144, 74540, 108664, 158372, 230800, 336480, 490458, 714856, 1041910, 1518840, 2213868, 3226896, 4703372, 6855388, 9992596, 14565048, 21229606, 30943516, 45102942, 65741224, 95822908, 139669094.

2.1.2 The 'Connective Constant'

Let a_n be the number of cube-free words of length n. Brandenburg [BRA83] proved that for $n > 18$

$$2 \times 1.080^n < 2 \times 2^{\frac{n}{9}} \leq a_n \leq 2 \times 1251^{\frac{n-1}{17}} < 1.315 \times 1.522^n$$

Thus $1.080 \leq \mu \leq 1.522$

Lemma 2.1 *$\{a_n\}$ is sub-multiplicative.*

Proof: Given a cube-free word of length $n + m$ if we split it into the first n letters and the last m letters both of these words must be cube-free or the original word was not. Hence $a_{n+m} \leq a_n a_m$.

It is also worth noting that when we adjoin two cube-free words we do not necessarily obtain a cube-free word so this is not a multiplicative sequence.

By Theorem 1.1 we know that $\mu = \lim_{n\to\infty} a_n^{1/n}$ exists and $\mu = \liminf_{n\to\infty} a_n$.

$\square$

Using the 'memory-45' analog (i.e. the corresponding sequence that enumerates words that avoid cubes x^3, with $length(x) \leq 15$), that was generated using the Maple package, up to word-length 300, we find the rigorous upper bound $\mu < 1.457579200596766$, which improves on Brandenburg's result.

Using Zeilberger's implementation of Zinn's method obtained from his Maple package GJsqfree (available from http://www.math.temple.edu/~zeilberg/), we also found that, assuming that $a_n \sim n^\theta \mu^n$, then $\mu \approx 1.457$, and $\theta \approx 0$. Hence it is reasonable to conjecture that $a_n \sim \mu^n$, where $\mu := \lim_{n\to\infty} a_n^{1/n} \approx 1.457$.

2.2 Lower-Bounds and the Brinkhuis Method

2.2.1 Lower Bounds for Square-free Ternary Words

Jan Brinkhuis [BRI83] obtained a lower bound for the number of square-free ternary words in the following way. He found a pair of words, $U0, V0$, on $\{0, 1, 2\}$ and from these formed $U1, V1$ and $U2, V2$ all with the following property. If W is a square-free word over $\{0, 1, 2\}$, and $S(W)$ is obtained by replacing all the 0's in W

with $U0$ or $V0$, the 1's with $U1$ or $V1$ and the 2's with $U2$ or $V2$ then $S(W)$ is also square-free.

Lemma 2.2 *(Brinkhuis) If we can find $U0, V0, U1, V1, U2, V2$ that satisfy the above condition and are of length k then $\mu \geq 2^{\frac{1}{k-1}}$.*

Proof: As we have two choices of what to substitute for each of the letters of W

$$a_{kn} \geq 2^n a_n$$

Thus

$$a_{kn}^{\frac{1}{kn}} \geq 2^{\frac{1}{k}}(a_n^{\frac{1}{n}})^{\frac{1}{k}}$$

and taking the limit with respect to n we obtain

$$\mu \geq 2^{\frac{1}{k}}\mu^{\frac{1}{k}}$$

which simplifies to

$$\mu \geq 2^{\frac{1}{k-1}}$$

□

Brinkhuis chose words that were palindromes and obtained $U1$ from $U0$ by adding 1 mod 3 to each letter of $U0$, $U2$ is obtained from $U0$ by adding 2 mod 3 to each letter of $U0$, likewise for $V1$ and $V2$. He found (by hand) such a Brinkhuis pair ($U0$ and $V0$) of length 24. Giving lower bound of $\mu \geq 2^{\frac{1}{23}} = 1.030595545$

Zeilberger and Ekhad [ZE98] removed the palindromic requirement and computerized the search for good pairs. They thus found a Brinkhuis pair of length 18, and so improved the lower bound to $\mu \geq 2^{\frac{1}{17}} = 1.04162$.

In their paper Zeilberger and Ekhad note that the relationship between $U0, U1$ and $U2$ and $V0, V1$ and $V2$ is not necessary, and it is with this comment in mind that we begin our adaptation of the Brinkhuis method to cube-free words.

2.2.2 Lower Bounds for Cube-Free Binary Words

Theorem 2.1 *The number of n-letter binary cube-free words is greater than* $2^{n/8}$.

This result can be obtained as a corollary of Brandenburg's result, but as our method is different from his we will give the full details.

The goal is to find binary words $U0, U1, V0, V1$ of minimal length such that if we take a cube-free word W over the alphabet $\{0, 1\}$ and substitute $U0$ or $V0$ for the zeros and $U1$ or $V1$ for the ones the resulting word $S(W)$ will also be cube-free.

Lemma 2.3 *If* $U0, V0, U1$, *and* $V1$ *satisfy the following conditions and if* W *is cube-free then* $S(W)$ *is cube-free.*

1) All legitimate triples of $U0, V0, U1, V1$ *are cube-free*

2) None of $U0, V0, U1, V1$ *are non-trivial factors of all the possible pairs of* $U0, V0, U1, V1$

Proof:

Clearly as $U0, V0, U1$, and $V1$ meet condition 1 then if W is cube-free and of length at most 3 then $S(W)$ is cube-free.

So if $S(W)$ contains a cube it has length greater than 3. For such a word to contain a cube the pattern of at least one of $U0, V0, U1$, and $V1$ must be repeated elsewhere in $S(W)$. If every time such a repetition occurs it is as $U0, V0, U1$, and $V1$ respectively then the original word W cannot have been cube-free (contrary to assumptions). So, the only way the repeat can occur is as a factor of a pair of concatenated words, but condition 2 eliminates this possibility. Therefore $S(W)$ is cube-free whenever W is.

□

Lemma 2.4 *If we can find $U0, V0, U1, V1$ that satisfy the above condition and are of length k then $\mu = \lim_{n\to\infty} a_n^{1/n} \geq 2^{\frac{1}{k-1}}$, where a_n is the number of cube-free words of length n.*

Proof: As for the lemma 2.2 in the square-free case.

Proof of Theorem: It is easily verified (by hand , or more quickly by computer) that $U0 = [0, 1, 1, 0, 0, 1, 1, 0, 1]$, $V0 = [0, 1, 1, 0, 1, 0, 0, 1, 0]$, $U1 = [1, 0, 0, 1, 1, 0, 0, 1, 0]$, and $V1 = [1, 0, 0, 1, 0, 1, 1, 0, 1]$ satisfy the conditions of the lemma. Hence $a(n) \geq 2^{1/8} \approx 1.09$.

□

It should be noted that our words are not palindromes, but $U1$ and $V1$ can be obtained by switching 1's and 0's and vice-versa in $U0$ and $V0$. Removing this condition does not produce any shorter choices for $U0, V0, U1$ and $V1$

CHAPTER 3

PATTERN FREE WORDS

In this chapter we will look at several specific cases of words avoiding certain patterns. The cube-free words in the previous chapter are an example of this type. The sequences were obtained using our Maple package patfr. In the table below we summarize the results when various patterns are avoided over various alphabet sizes. Each sequence is given a number so that it can be referenced in the discussion that follows. It should also be noted that all symmetries of the representative pattern are avoided and that k is the size of the alphabet. Note in these examples, $A = B$ is allowed.

Pattern 1 represents square-free ternary words, which as we have mentioned were studied by Noonan and Zeilberger in [NZ99], and pattern 8 represents cube-free binary words, which were discussed in the previous chapter. The next case in this sequence occurs when we avoid blocks of the form $xxxx$, and is pattern 10.

Table 3.1: Enumeration of pattern free words.

No.	Pattern	k	n=1	2	3	4	5	6	7	8	9	10
1	A A	3	3	6	12	18	30	42	60	78	108	144
2	A B	2	2	0	0	0	0	0	0	0	0	0
3	A B	k	k	0	0	0	0	0	0	0	0	0
4	A B A	2	2	4	4	2	0	0	0	0	0	0
5	A B A	3	3	9	18	24	18	6	0	0	0	0
6	A A B	2	2	4	4	4	0	0	0	0	0	0
7	A B B	2	2	4	4	4	0	0	0	0	0	0
8	A A A	2	2	4	6	10	16	24	36	56	80	118
9	A B C	2	2	4	0	0	0	0	0	0	0	0
10	A A A A	2	2	4	8	14	26	48	88	160	292	532
11	A A A B	2	2	4	8	12	20	32	48	72		
12	A A B A	2	2	4	8	12	16	18	16	10	4	0
13	A B A B	2	2	4	8	12	20	26	38	42	52	56
14	A B A B	3	3	9	27	72	198					
15	A B B A	2	2	4	8	12	18	18	14	8	6	2
16	A B B A	3	3	9	27	72	192					
17	A A B B	2	2	4	8	12	18	22	28	28	22	18
18	A B A C	2	2	4	8	8	4	0	0	0	0	0
19	A B C A	2	2	4	8	8	4	2	0	0	0	0
20	A A B C	2	2	4	8	8	8	0	0	0	0	0
21	A B B C	2	2	4	8	8	8	0	0	0	0	0
22	A B C A B	2	2	4	8	16	24					
23	A B A B A	2	2	4	8	16	28	52	90			
24	A B A B A, A A A	2	2	4	6	10	14	20	24	30	36	44

Further examples like the generalization of pattern 2 to the case of general dimension size in pattern 3 will be discussed in the next chapter.

Patterns 13 and 14 are examples of square-free words where we require the length of each block x in the square xx to be of length at least 2. Unlike the regular square-free situation, here we do make it past $n = 4$.

Overlap-free words [FI99] are represented by pattern 24. The other patterns shown have not been studied in any detail. Some sequences quickly converge to zero, but others do show signs of continued growth. The number of terms given for each sequence depends on the amount of memory required to calculate each case, and can usually be improved on by writing sequences specific programs.

CHAPTER 4

DIMENSION TWO AND BEYOND

In this chapter we explore the use of the Goulden-Jackson method in the situation when the alphabet size is unknown, or it can be thought of as symbolic. We first explore several examples, then look into how we can implement the process on computer.

4.1 Linear Burstein-Wilf

In a later chapter we will look at the result of Burstein and Wilf [BW97] for cyclic words. Here we are looking at the linear analog. On an alphabet of size k, how many words are there of length n that avoid any long constant blocks of length m or greater?

For example in a two letter alphabet if we avoid all blocks of length 2 or more then the only possible words of length 5 are [1, 2, 1, 2, 1] and [2, 1, 2, 1, 2].

In the general situation, it is fairly easy to apply the Goulden-Jackson Method. Referring to the notation of Chapter 1 and letting α^m represent a block of length m made up of some letter α we have:

$$W(C[\alpha^m]) = -s^m - \sum_{t=1}^{m-1} s^t W(C[\alpha^m])$$

Thus for each $\alpha \in L$ we have

$$W(C[\alpha^m]) = \frac{-s^m(s-1)}{-1+s^m}$$

Assuming the alphabet is of size k there are k such clusters so the total cluster weight is k-times this and the resulting generating function is:

$$f(s) = \frac{-1+s^m}{-1+s^m+ks-ks^m}$$

4.2 Linear $\alpha\beta$

In this example we consider words over a k-letter alphabet that avoid any blocks of the form $[\alpha, \beta]$ where $\alpha \neq \beta$.

For example over a three letter alphabet we avoid all blocks in the set $\{[1,2],[1,3],[2,1],[2,3],[3,1],[3,2]\}$.

So let us go right ahead and apply the Goulden-Jackson Method.

$$W(C[\alpha,\beta]) = -s^2 - \sum_{\gamma\neq\beta} sW(C[\beta,\gamma])$$

Thus, summing over all $\alpha, \beta \in L$ we have

$$\begin{aligned} W = \sum_{\beta \in L} \sum_{\alpha \neq \beta} W([C[\alpha, \beta]) &= -\sum_{\beta \in L} \sum_{\alpha \neq \beta} s^2 - \sum_{\beta \in L} \sum_{\alpha \neq \beta} \sum_{\gamma \neq \beta} sW(C[\beta, \gamma]) \\ &= -k(k-1)s^2 - (k-1)s \sum_{\beta \in L} \sum_{\gamma \neq \beta} W(C[\beta, \gamma]) \end{aligned}$$

Noting that the double sum on the far right is equivalent to that on the left hand side of the equation and that this is our desired cluster weight we obtain.

$$W = \frac{-k(k-1)s^2}{1 + (k-1)s}$$

And hence

$$f(s) = \frac{1 + (k-1)s}{1-s} = 1 + ks + ks^2 + ks^3 + ks^4 + \ldots$$

In fact this example is fairly easy to deduce without the Goulden-Jackson Method, as the only words allowed are those containing only one letter repeated.

Let us now look at a more general example of this type.

4.3 Linear α _ ω

This is the general analog of the above example. Here we wish to avoid words of the form $[\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_p]$ with $\alpha_i \neq \alpha_j$ for $i \neq j$. As before we simply apply the Goulden-Jackson Method.

$$W(C[\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_p]) = -s^p - \sum_{t=1}^{p-1} \sum_{t+1 \leq i,j \leq t+p} s^t W(C[\alpha_{t+1}, \alpha_{t+2}, \alpha_{t+3}, \ldots, \alpha_{t+p}])$$

By summing over the α_i's we obtain:

$$W = -\frac{k!s^p}{(k-p)!} - \sum_{t=1}^{p-1} \frac{(k-p+t)!}{(k-p)!} s^t W$$

where the factorials are due to the number of letters we have choice over. We can now solve for W and find our generating function.

$$f(s) = \frac{\sum_{t=0}^{p-1}(k-p+t)!s^t}{k!s^p + (1-ks)\sum_{t=0}^{t=p-1}(k-p+t)!s^t}$$

which agrees with our previous result for $p = 2$.

4.4 Linear $\alpha^m\beta^n$

Here we look at the linear case of words over some alphabet L with k letters that avoid any blocks of the form $[\alpha^m\beta^n]$ where $\alpha \neq \beta$. In a later chapter we will look at the cyclic version of this case.

Theorem 4.1 *The number of words of length r over a k letter alphabet that avoid $[\alpha^m\beta^n]$ where $\alpha \neq \beta$ is the coefficient of s^r in the generating function*

$$f(s) = \frac{s - 1 + (k-1)(s^{m+n} - s^{\max(m,n)})}{(s - 1 + (k-1)(s^{m+n} - s^{\max(m,n)}))(1-ks) + k(k-1)s^{m+n}(s-1)}$$

Proof: As in the above examples the proof is by the Goulden-Jackson Method. Firstly we find the weight of a general cluster.

$$W(C[\alpha^m\beta^n]) = -s^{m+n} - \sum_{\gamma\in L,\gamma\neq\beta} W(C[\beta^m\gamma^n]) \sum_{t=1}^{\min(m,n)} s^{m+n-t}.$$

Now we sum over α and β to obtain the total weight of the clusters

$$\begin{aligned}
W &= \sum_{\alpha\in L,\alpha\neq\beta}\sum_{\beta\in L} W(C[\alpha^m\beta^n]) \\
&= \sum_{\alpha\in L,\alpha\neq\beta}\sum_{\beta\in L} -s^{m+n} - \sum_{\alpha\in L,\alpha\neq\beta}\sum_{\beta\in L}\sum_{\gamma\in L,\gamma\neq\beta} W(C[\beta^m\gamma^n]) \sum_{t=1}^{\min(m,n)} s^{m+n-t} \\
&= -k(k-1)s^{m+n} - (k-1)(\sum_{\beta\in L}\sum_{\gamma\in L,\gamma\neq\beta} W(C[\beta^m\gamma^n]))(\sum_{t=1}^{min(m,n)} s^{m+n-t}).
\end{aligned}$$

By evaluating the geometric sum on the far right and noting that the double sum in the first line and that in the last are equivalent we can solve for them to obtain:

$$\begin{aligned}
W &= \sum_{\alpha\in L,\alpha\neq\beta}\sum_{\beta\in L} W(C[\alpha^m\beta^n]) \\
&= \frac{-k(k-1)s^{m+n}(s-1)}{s-1+(k-1)(s^{m+n}-s^{\max(m,n)})}.
\end{aligned}$$

And we obtain our required result by substituting this into

$$f(s) = \frac{1}{1-ks-W}$$

□

To enable calculation of more generating functions of this type with k, the alphabet size, symbolic the Goulden-Jackson Method was extended to produce generating functions for an arbitrary size alphabet. The package developed, called GJdim, uses the fact that the number of ways one pattern can overlap with the symmetries of another is related to the number of letters that we are free to choose after the letters to match the overlap have been defined.

We shall use the memory m situation to apply GJdim to square-free and cube-free words. That is that we shall obtain formulas for the generating function for all words that avoid squares (or cubes) up to length m.

4.5 Square-Free Words

Memory 2

Here the fundamental mistake is $[1, 1]$, to find our generating function we note

$$W([1,1]) = -s^2 - sW([1,1]).$$

Solving for $W([1, 1])$ and noting that there are k mistakes of this type we obtain

$$W(clusters) = \frac{-ks^2}{1+s}$$

and hence

$$f_2(s) = -\frac{1+s}{-1-s+ks}$$

Memory 4

Here the fundamental mistakes are $\{[1,1],[1,2,1,2]\}$ and we obtain (by hand or using our Maple package GJdim)

$$f_4(s) = -\frac{1+2s+2s^2+ks^3}{-1-2s-2s^2+ks+ks^2}.$$

Memory 6

Now the mistakes are $\{[1,1],[1,2,1,2],[1,2,3,1,2,3]\}$, and it is still fairly easy to do the calculation by hand, but much quicker by computer. The generating function is

$$f_6(s) = -\frac{3s^2+1+2s^3+2s-ks^5+ks^3+ks^4+s^5k^2}{ks^3-2s^3-1-3s^2-2s+ks+ks^2}.$$

Memory 8

In this case we have a much larger set of mistakes, and doing the calculation by hand becomes much harder. The mistakes are $\{[1,1], [1,2,1,2], [1,2,3,1,2,3], [1,2,3,4,1,2,3,4], [1,2,1,3,1,2,1,3], [1,2,3,2,1,2,3,2]\}$ and our generating function is

$$\begin{aligned} f_8(s) = -(&-1 - 4s - 10s^2 - 32s^6 + s^{14}k^4 - ks^3 - 18s^3 - 26s^4 - 30s^5 + 2s^{12}k^2 \\ &+s^{17}k^4 + 14s^{15}k^2 - 3s^6k^2 + 6s^{11}k^2 - 4ks^{17} - 8ks^{13} - 2s^7k^2 \\ &-4ks^6 - 3ks^7 - 2ks^{16} - 9ks^{15} + 4ks^{10} + ks^{14} + 3ks^{12} \\ &+2ks^{11} + 4ks^9 - 4s^{13}k^3 + 9s^{13}k^4 + s^{16}k^4 - 2s^{14}k^3 \\ &-3ks^4 + 5s^{16}k^2 - 5s^{17}k^3 + 8s^{17}k^2 - 4s^{16}k^3 - 33s^{11} - 27s^{12} \\ &-37s^9 - 38s^{10} - 12s^{13} - 6s^{14} - 2s^{11}k^3 - s^7k^3 - 9s^{15}k^3 - 2s^8k^3 \\ &-k^2s^5 + 2s^{15}k^4 - 3s^9k^3 - 5ks^5 - 33s^8 - 32s^7 + 3s^9k^2 - 3s^{10}k^3 \\ &+6s^{10}k^2)/(1 - 3ks^2 + 4s + 10s^2 + 32s^6 - ks - 6ks^3 + 18s^3 \\ &+26s^4 + 30s^5 + 4s^{12}k^2 + 5s^{11}k^2 - 10ks^{13} + s^7k^2 - 12ks^6 \\ &-15ks^7 + 2s^8k^2 - 28ks^{10} - 5ks^{14} - 21ks^{12} - 26ks^{11} - 18ks^8 \\ &-25ks^9 + 2s^{13}k^2 + s^{14}k^2 - 9ks^4 + 33s^{11} + 27s^{12} + 37s^9 + 38s^{10} \\ &+12s^{13} + 6s^{14} - 11ks^5 + 33s^8 + 32s^7 + 4s^9k^2 + 5s^{10}k^2). \end{aligned}$$

The generating functions get fairly complex quite quickly and we are limited by the memory of the computer and the complexity of the system of equations that must be solved.

4.6 Cube-Free Words

Memory 3

In this case our fundamental mistake is $[1, 1, 1]$ and it is fairly easy to apply the Goulden-Jackson method by hand to obtain

$$g_3 = -\frac{1+s+s^2}{-1-s-s^3+ks+ks^2}.$$

Memory 6

Now our mistakes are $\{[1, 1, 1], [1, 2, 1, 2, 1, 2]\}$ and the generating function obtained using GJdim is

$$g_6 = -\frac{1+2s+3s^2+3s^3+3s^4+ks^5+s^5+ks^6}{-s^5-3s^4-3s^2-2s+ks+ks^5+2ks^4+2ks^3+2ks^2-3s^3-1}.$$

Again we are limited by computer capacity and currently can go no further.

CHAPTER 5

SELF-AVOIDING WALKS THAT AVOID OTHER FACTORS ON THE 2-D CUBIC LATTICE

5.1 Self-Avoiding Walks on the 2-d Cubic Lattice

Anyone who likes a little variety will try to take self-avoiding walks. For now we consider walks much like those through a city whose streets form a grid. A walk is self-avoiding if we never visit the same intersection twice. This is modeled by a walk on the integer lattice in 2-dimensions, where we never return to a lattice point after we have left it.

Definition 5.1 *A* **self-avoiding walk** *is a path on any lattice that does not visit the*

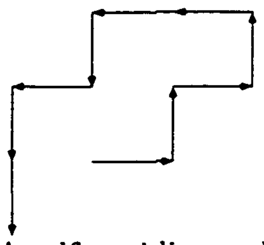
Figure 5.1: A self-avoiding walk.

same site twice [MS96].

In 2-dimensions we can use the alphabet $\{1, -1, 2, -2\}$ as our set of possible steps. Here 1 represents a step to the right, -1 a step to the left, 2 a step up and -2 a step down.

Using our notation this is equivalent to a word is self-avoiding if it contains no factors for which the number of 1s and -1s are equal and the number of 2s and -2s are equal. The Maple package walk (available from **http://www.math.temple.edu/~anne/sqfrwalk.html**) can be used to derive or count the number of self-avoiding walks on a cubic lattice that avoid an input set of mistakes in any given dimension.

We will investigate walks that are not only self-avoiding, but also avoid a prescribed set of additional mistakes.

5.2 Self-Avoiding Walks that Avoid Double Steps.

These are walks for people who get bored of the view ahead of them and so at every cross roads turn right or left, never going straight on.

Definition 5.2 *A word avoids* **double steps** *if it contains no factors of the form ww, where w is any single step.*

In our notation this means it excludes

$\{[1,1],[2,2],[-1,-1],[-2,-2]\}$ as factors.

Theorem 5.1 *The number of self-avoiding walks that avoid double steps for n from 0 to 20 are:*

[1, 4, 8, 16, 24, 40, 64, 104, 168, 272, 440, 712, 1128, 1808, 2896, 4640, 7368, 11744, 18752, 29920, 47376].

Or equivalently:

$$\begin{aligned} a(0) &= 1 \\ a(n) &= 2^{n+1} \textit{ if } 1 \leq n \leq 3 \\ a(n) &= a(n-1) + a(n-2) \textit{ if } 4 \leq n \leq 11 \\ a(n) &\leq a(n-1) + a(n-2) \textit{ if } n \geq 12 \end{aligned}$$

Proof: Firstly we note that all double steps have been eliminated and the walk must contain no immediate reversals if it is to be self-avoiding. Thus every 1 or -1 must be followed by a 2 or -2 and vice versa. This means that the only other way a walk of length less than 12 can fail to be self-avoiding is if it contains a unit square. This is due to the fact the next self-avoiding polygon that avoids double steps is of length 12 (it looks like a plus sign).

The case $n = 0$ is a convention. There is precisely one empty word.

For $1 \leq n \leq 3$ we note that there are 4 initial steps and we have 2 choices for our second step and again for our third step as explained above.

Finally we consider the interesting case $n \geq 4$. As explained above our only danger for $4 \leq n \leq 11$ are unit squares, and for $n \geq 12$ we will only consider this danger. This means it suffices to only look at the three previous steps to decide what our next step may be. Without loss of generality we may assume the first two steps of this block of three steps are 1 and 2. Then regardless of whether the third step is 1 or -1 we may chose 2 for our fourth step. See Figure 5.2. This generates $a(n-1)$ walks.

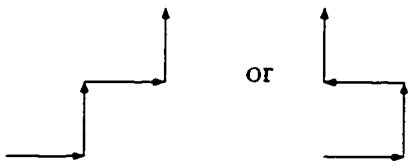

Figure 5.2: Fourth step equals second step.

Now we investigate when we may allow -2 to be our fourth step. We may only do this if step one and three are the same, else we will form a square. See Figure 5.3. Thus for every step one and two there is only one way we can have step four as minus step two. This generates $a(n-2)$ walks.

Thus for $4 \leq n \leq 11$ we have $a(n) = a(n-1) + a(n-2)$, our Fibonacci style sequence, and when $n \geq 12$ we have that $a(n) \leq a(n-1) + a(n-2)$, in fact this inequality is strict as we are now avoiding plus sign style shapes.

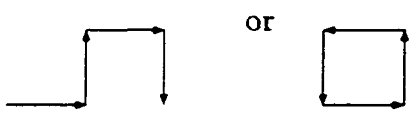

Figure 5.3: Fourth step is minus second step.

□

5.3 Square-Free Self-Avoiding Walks

Now we consider a much stricter case. Here our walks are not only self-avoiding, but at no time do we repeat the same sequence of steps twice in a row. For example we cannot go left, straight, right, left, straight, right.

Theorem 5.2 *The number of square-free self-avoiding walks of length n is given by the sequence* $1, 4, 8, 16, 16, 16, 16, 16, 0, 0, 0, 0, 0, 0, 0, 0$ *for* $0 \leq n \leq 15$.

Proof:

By making our walks self-avoiding we know we must eliminate all immediate back steps and all polygons, at the very least. This means that none of the following set of words may appear as a factor of any of our words:

$$\{ \ [1,-1],[2,-2],[-1,1],[-2,2],[1,2,-1,-2],[2,-1,-2,1],[-1,-2,1,2],$$
$$[-2,1,2,-1],[-1,2,1,-2],[-2,-1,2,1],[1,-2,-1,2],[2,1,-2,-1] \quad \}.$$

The fact the walks are also square free eliminates double steps and double 'corners', that is paths like (right, up, right, up). So we must also eliminate all of the following as factors:

$$\{ \quad [1,1],[-1,-1],[2,2],[-2,-2],[2,1,2,1],[-2,1,-2,1],[2,-1,2,-1],$$
$$[-2,-1,-2,-1],[1,2,1,2],[-1,2,-1,2],[1,-2,1,-2],[-1,-2,-1,-2] \quad \}.$$

So let us now try to form a square-free self-avoiding walk. By symmetry it does not matter in which direction we start, so let our first step be a 1.

Now our second step may not be -1 as the walk is self-avoiding, and it can not be 1, because our walk is square-free. So, our next step must be 2 or -2. Again by symmetry it does not matter which we chose, so we will pick 2.

Our walk so far is $[1,2]$. Now as before, we may not pick -2 or 2, because our previous step was 2, so we must pick 1 or -1. Both cases are very similar so we will only look at the case that the next step is 1. The case when the next step is -1 is left to the reader.

We now have $[1,2,1]$. For our next step we may not pick -1 or 1, because the last step was a 1, and we may not pick 2, because $[1,2,1,2]$ is a square (of $[1,2]$), this means we are forced to pick a -2.

Now we have $[1,2,1,-2]$. From here we may not pick -2 or 2 as usual, and we may not pick -1, or the last four steps will form a polygon $[2,1,-2,-1]$, and so our walk will not be self-avoiding. Thus we are forced to pick 1.

We are forced into our next step up until the eighth step. Here is the position after 7 steps:

$[1,2,1,-2,1,2,1]$. See Figure 5.4.

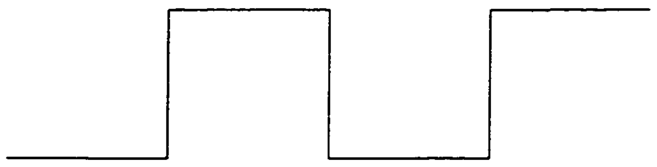

Figure 5.4: A seven step square free self-avoiding walk

Now based on the previous analysis we must chose -2 as our next step, but if we do this we will have the $[1, 2, 1, -2]$ twice in succession. So as we want our walk to be square-free we are stuck, and can take no further steps. Thus for $n \geq 8$ the number of square-free self-avoiding walks is zero.

$\square$

5.4 Cube-Free Self-Avoiding Walks

In this section we consider walks that are both self-avoiding and cube-free as in the sense in Chapter 2.

Theorem 5.3 *The number of cube-free self-avoiding walks for* $0 \leq n \leq 10$ *is* $1, 4, 12, 32, 80, 200, 472, 1136, 2656, 6256, 14584.$

Proof: Obtained by using the cubes of length ≤ 9 and the mistakes for self-avoiding walks up to length 10 in a Maple application of the Goulden-Jackson method.

5.5 Rate of Growth

The idea behind investigating Self-Avoiding Walks that also avoid other steps is that we will be able to produce sequences that grow slowly enough to be analysable (unlike Self-Avoiding Walks), but quickly enough to give us information about Self-Avoiding Walks in general. The three examples given above are all of interest in their own right, but do not help us to learn more about Self-Avoiding Walks. Clearly the second example's sequence becomes zero too quickly to be of use and the first and third examples are a little better but provide no new information. There are many other examples that can be explored and there is still hope that this approach will help us learn more about Self-Avoiding Walks.

CHAPTER 6

SELF-AVOIDING WALKS IN k DIMENSIONS

So far we have only looked at Self-Avoiding Walks in two dimensions, but the idea is not dimension dependent. In fact we shall see later it is not even lattice dependent. As in Chapter 5 where we looked at words that avoided certain patterns in undefined dimensions, here we look at self-avoiding walks in k dimensions. As this is an example where there are infinitely many mistakes we can not obtain an exact generating function using the Goulden-Jackson Method, but we can use the finite memory approach to see the general pattern. Using a Maple package we developed called SPGJdim, which takes into account the symmetry of the mistakes both with regard to the symmetric group and sign changes, we obtained the following generating functions.

Memory 2

Here we use the fundamental mistake [1, −1] and the package SPGJdim to obtain

$$f_2(s) = -\frac{1+s}{-1-s+2ks}.$$

Memory 4

The mistakes are now of the form $\{[1,-1],[1,2,-1,-2]\}$ and the generating function is

$$f_4(s) = -\frac{1+2s+2s^2-s^3+2s^3k}{-1-2s-2s^2+s^3+2ks+2ks^2}$$

Memory 6

Now we have $\{[1,-1],[1,2,-1,-2],[1,1,2,-1,-1,-2],[1,2,2,-1,-2,-2]\}$ as our set of fundamental mistakes and our generating function becomes

$$\begin{aligned}
f_6(s) = -(&1 - 24ks^6 + 7s^7 - 9s^8 + 4s - 4s^5k - 4ks^{19} + 9s^2 + 18ks^4 \\
&+16k^3s^{17} - 8k^3s^{16} - 48k^2s^{17} + 8s^{18}k^3 - 16s^{18}k^2 + 24s^{16}k^2 \\
&+10s^{18}k - 26s^{16}k - 32k^3s^{15} - 24k^3s^{14} + 92k^2s^{15} + 48s^{14}k^2 \\
&-2s^{14}k - 9s^4 + 10s^3 + 16s^5k^2 - 19s^5 - 14s^{13} + 32k^3s^9 + 34s^{17}k \\
&-64s^{15}k - 8s^{13}k^3 + 17s^{12} + 8s^{11}k^3 - 12s^{11}k^2 - 36s^{11}k + s^{17} \\
&+10s^{16} - 3s^{15} - 28s^{14} - 60ks^7 + 36k^2s^7 - s^{18} - 26s^6 + 16s^{10}k^3 \\
&+14s^{10}k - 44k^2s^{10} + 16s^{10} + 4s^3k + 102s^9k - 108k^2s^9 - 22s^9 \\
&+28k^2s^6 + 50s^8k - 64s^8k^2 + 24k^3s^8 - 8k^2s^{12} + 20s^{13}k \\
&+4k^2s^{13} + 48s^{11} + 4k^2s^{19})/(-1 - 32ks^6 - 7s^7 + 9s^8 - 4s \\
&-18s^5k + 6ks^2 - 9s^2 - 6ks^4 + 2sk + 6s^{16}k + 4k^2s^{15} + 8s^{14}k^2 \\
&-32s^{14}k + 9s^4 + 4k^2s^4 - 10s^3 + 8s^5k^2 + 19s^5 + 14s^{13} + 2s^{17}k \\
&-10s^{15}k - 17s^{12} - 20s^{11}k^2 + 64s^{11}k - s^{17} - 10s^{16} + 3s^{15} + 28s^{14} \\
&-12ks^7 + 8k^2s^7 + s^{18} + 26s^6 + 22s^{10}k - 8k^2s^{10} - 16s^{10} + 8s^3k \\
&-22s^9k + 4k^2s^9 + 22s^9 + 12k^2s^6 - 24s^8k + 8s^8k^2 - 12k^2s^{12} \\
&-10s^{13}k + 32s^{12}k - 48s^{11})
\end{aligned}$$

As is clearly seen by the case memory 6 the generating function gets very complicated very quickly, and in fact the current package cannot handle the memory 8 case.

CHAPTER 7

SELF-AVOIDING WALK TYPE PROBLEMS

So far we have looked at walks that are based on unit steps in the available directions. In this section we consider the case of a less traditional walker. Specifically we will look at the Self-Avoiding Knight. In this example we are allowing steps of the form $\{[1,2],[2,1]\}$ and all their symmetries on a two dimensional rectangular lattice. You will probably recognize these as the legitimate moves for a knight in chess. With the aid of Maple it is fairly easy to count the number of different walks of n knight steps long that never visit the same lattice point twice. In fact you can simply think of this as counting the number of walks of n moves that a knight can make on a infinite chess board without ever visiting the same square twice. We obtain the following sequence:

$[1, 8, 56, 392, 2696, 18584]$

We are unable to go further due to computer memory, but what is interesting is for $1 \leq n \leq 5$ this exactly agrees with the number of regular Self-Avoiding Walks on a 4-d cubic lattice.

With improved memory, it would be possible to see whether an isomorphism between the two situations is likely to exist or if this is purely a coincidence.

CHAPTER 8

SUPER k SELF-AVOIDING WALKS

Super k Self-Avoiding Walks are like self-avoiding walks only more so. In addition to never visiting the same sight twice, they also never get within j steps of a previously visited sight once they have gone j steps away from it for $1 \leq j \leq k$.

The conditions we need for this to be true for a given walk in two dimensions are

$$|\#(1) - \#(-1)| + |\#(2) - \#(-2)| > k$$

and

$$|\#(1) - \#(-1)| + |\#(2) - \#(-2)| > j, 0 \leq j \leq k-1 \text{ for previous } j+1 \text{ steps}$$

For example on the 2-d cubic lattice with $k = 2$ the allowable 3 step walks are represented by $\{[1,2,2],[1,2,1],[1,1,2],[1,1,1],[1,1,-2]\}$.

The normal Self-Avoiding Walk can be considered the case $k = 0$, that is to say that we can get as close as we want to any previously visited sight as long as we do not visit it. The following table summarizes the number of Super k Self-Avoiding Walks for $0 \leq k \leq 10$ and $0 \leq n \leq 10$, and the corresponding values of μ,obtained by Zinn's method.

Table 8.1: Super k self-avoiding walks.

k	0	1	2	3	4	5	6	7	8	9
$n = 0$	1	1	1	1	1	1	1	1	1	1
1	4	4	4	4	4	4	4	4	4	4
2	12	12	12	12	12	12	12	12	12	12
3	36	28	28	28	28	28	28	28	28	28
4	100	68	60	60	60	60	60	60	60	60
5	284	164	132	124	124	124	124	124	124	124
6	780	396	292	260	252	252	252	252	252	252
7	2172	940	644	548	516	508	508	508	508	508
8	5916	2244	1420	1156	1060	1028	1020	1020	1020	1020
9	16268	5324	3132	2436	2180	2084	2052	2044	2044	2044
10	44100	12668	6884	5132	4484	4228	4132	4100	4092	4092
μ	2.738	2.378	2.208	2.106	2.027	2.032	2.018	2.006	1.997	1.997

It is clear from the table that as k increases the number of possible walks decrease and the rate of this decrease also decreases, that is that the effect of increasing k by 1 is more noticeable when k is small than when k is large. It should also be noted that the effect of k does not appear until the $k+2$nd step.

CHAPTER 9

SELF-AVOIDING WALKS ON A HONEYCOMB LATTICE

Previously we have been considering only the traditional rectangular lattice. In this chapter we consider the case of a Self-Avoiding Walk on a Honeycomb Lattice as in Figure 9.1 below.

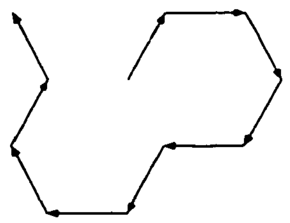

Figure 9.1: Self-avoiding walk on a honeycomb lattice.

In order to make use of the computer in exploring the growth of this type of walk we must find a notation for describing it. We let the basic steps be $\{a, b, c, -a, -b, -c\}$ with the added conditions that a, b and c may only be followed by $-a, -b$ or $-c$, and that $-a$, $-b$ and $-c$ may only be followed by a, b or c. Under this notation the example in Figure 9.1 would be $[-c, a, -b, c, -a, c, -a, b, -c, b]$

We will let $N(t)$ =the number of times t appears in the sequence for $t \in \{a, b, c, -a, -b, -c\}$. Then as in the previous examples of self-avoiding walks it is easy to see that a walk on the honeycomb lattice is self-avoiding if for no sub-sequences

$$N(a) = N(-a), N(b) = N(-b) \text{ and } N(c) = N(-c). \tag{9.1}$$

By producing sequences for which 9.1 is true we can apply the Goulden-Jackson Method using these for our mistakes. Due to memory limitations we are only able to obtain the first 13 terms of the sequence $a(n)$. Then applying Zinn's method we obtain the estimate $\mu = 1.899963712$ for the connective constant. This does not improve on the current best upper bound which was obtained by Alm [ALM93] as $\mu < 1.87603$.

CHAPTER 10

ENTROPY OF A LANGUAGE

In probability entropy is a measure of the randomness of a random variable. For a distribution with N outcomes the entropy will be greatest when the probability of each outcome is $\frac{1}{n}$. Thus, the greater our entropy the more evenly distributed our events are.

Definition 10.1 *The entropy of a discrete random variable X whose ith outcome has probability p_i, is given by*

$$H(X) = -\sum_{i=0}^{N} p_i \log_m p_i \tag{10.1}$$

where the choice of the base of the logarithm is one of convenience.

When considering the entropy of a language we divide the words into blocks of length k. For example if $k = 3$ then the word $[a, b, c, d, e, f, g, h, i, j, k, l]$ would be broken down to $[a, b, c]$, $[d, e, f]$, $[g, h, i]$, and $[j, k, l]$. We will only consider words whose length is a multiple of k. We then count the frequency with which these blocks occur and thus find their probability. We will look at two examples in depth square-free ternary words and then cube-free binary words.

10.1 Entropy of Square-Free Ternary Words

Recall that square-free words avoid any factors of the form xx where x is any non-empty word. In our choice of base for the logarithm we first considered all possible ternary words of length k giving us a base of 3^k, see Table 10.1.

Table 10.1: Entropy of square-free ternary words with base 3^k

	k	2	3	4	5	6	7	8	9	10
n										
2		.8155								
3			.7540							
4		.8155		.6577						
5					.6192					
6		.8155	.7508			.5670				
7							.5324			
8		.8155		.6564				.4957		
9			.7487						.4735	
10		.8155			.6092					.4524
11										
12		.8155	.7476	.6567		.5658				
13										
14		.8155					.5183			
15			.7455		.6083					
16		.8155		.6571				.4939		
17										
18		.8155	.7450			.5644			.4721	
19										
20		.8155		.6575	.6064					.4507

Then as each of ourblocks must automatically be square-free we used a base of the number of square-free ternary words of length k see Table 10.2.

Table 10.2: Entropy of square-free with base the number of valid k length words.

	k	2	3	4	5	6	7	8	9	10
n										
2		1.000								
3			1.000							
4		1.000		1.000						
5					1.000					
6		1.000	.9959			1.000				
7							1.000			
8		1.000		.9980				1.000		
9			.9931						1.000	
10		1.000			.9838					1.000
11										
12		1.000	.9915	.9984		.9979				
13										
14		1.000					.9734			
15			.9888		.9824					
16		1.000		.9991				.9964		
17										
18		1.000	.9882			.9954			.9970	
19										
20		1.000		.9996	.9794					.9962

Comparing these two tables we can see that once we allow for the fact that only certain blocks of length k can possibly occur the entropy of square-free ternary words is close to 1. This shows that the square-free blocks of length k are fairly evenly distributed throughout the words.

10.2 Entropy of Cube-Free Binary Words

A cube-free word avoids any factors of the form xxx where x is any non-empty word. In our choice of base for the logarithm we first considered all possible binary words of length k giving us a base of 2^k, see Table 10.3, then as each of our blocks must automatically be cube-free we used a base of the number of cube-free binary words of length k, see Table 10.4. The results are summarized in the following tables, as usual n represents the length of the words.

Table 10.3: Entropy of cube-free binary words with base 2^k

k	2	3	4	5	6	7	8	9	10
n									
2	1.000								
3		.8617							
4	.9855		.8305						
5				.8000					
6	.9820	.8617			.7642				
7						.7386			
8	.9772		.8275				.7259		
9		.8617						.7024	
10	.9738			.7932					.6883
11									
12	.9712	.8617	..8233		.7594				
13									
14	.9696					.7346			
15		8617		.7886					
16	.9686		.8207				.7156		
17									
18	.9679	.8617			.7563			.6993	
19									
20	.9672		.8192	.7858					.6852

Comparing the two tables for cube-free words we again can see that once we allow

Table 10.4: Entropy of cube-free with base the number of valid words of length k

	k	2	3	4	5	6	7	8	9	10
n										
2		1.000								
3			1.0000000000							
4		.9855		1.000						
5					1.000					
6		.9820	1.0000000000			1.000				
7							1.000			
8		.9772		.9964				1.000		
9			1.0000000000						1.000	
10		.9738			.9916					1.000
11										
12		.9712	.9999978390	.9913		.9938				
13										
14		.9696					.9955			
15			.9999965264		.9857					
16		.9686		.9882				.9857		
17										
18		.9679	.9999967883			.9897			.9956	
19										
20		.9672		.9865	.9823					.9955

for the fact that only certain blocks of length k can possibly occur the entropy of cube-free binary words is close to 1. This shows that the cube-free blocks of length k are fairly evenly distributed throughout the words.

In contrast to the square-free case, where for block length 2 the entropy was 1 for all n (see Table 10.2), implying that the square-free blocks $[0,1], [0,2], [1,0], [1,2], [2,0], [2,1]$ all occur with equal frequency in the cube-free case with block length 3 we see a slight deviation from 1 (see Table 10.4).

CHAPTER 11

APPLYING THE GOULDEN-JACKSON METHOD TO A PROBABILISTIC SITUATION

11.1 The Formula

The work of Noonan and Zeilberger to apply the Goulden-Jackson Method to various situations can be extended to a probabilistic situation in the following way.

In Chapter 1 the generating function for words that avoid a certain set of bad words was found to be:

$$f(s) = \frac{1}{1 - ks - weight(C)} \tag{11.1}$$

where k is the number of letters in the alphabet used and $weight(C)$ the weight of the clusters formed by bad words.

We extend this equation to include the following information:

1. The probability for each letter being in the first position, $y[a]$.
2. The conditional probability for each pair of letters, $t[a, b]$.

And the new equation for the generating function is:

Theorem 11.1

$$f(s) = 1 + \sum_{a \in V} y[a] \frac{x[a] + weight'(C_a)}{1 - \sum_{b \in V} x[b] - \sum_{b \in V} weight'(C_b)} \tag{11.2}$$

Proof: Clearly if we let $L(B)$ be the set of words that avoid all bad words then

$$\begin{aligned} f(s) &= weight([]) + \sum_{a \in V} \sum_{w \in L(B), w = au} weight(w) \\ &= 1 + \sum_{a \in V} \sum_{w \in L_a(B)} weight(w) \end{aligned}$$

where $L_a(B)$ represents the set of good words that begin with the letter a.

Now the goal is to find $f_a(s)$, that is the generating function for all good words that start with the letter a for each $a \in V$. As all words in f_a start with a their weight includes the factor $y[a]$ so we define $weight'(w) = \frac{weight(w)}{y[a]}$. We now have

$$f(s) = 1 + \sum_{a \in V} y[a] f'_a(s) \tag{11.3}$$

$f'_a(s)$ is the generating function for the members of $L_a(B)$ using *weight'*.

Now we use the ideas described in Chapter 1 to find $f'_a(s)$.

Let M_a be the set of marked words that start with the letter a, M the set of marked words, C_a the set of clusters beginning with a, and $g(s) = \sum_{w \in M} weight'(w)$. Also note that $f'_a = \sum_{w \in M_a} weight'(w)$.

Now if $w \in M_a$ then one of the following is true

- $w = a$
- w starts with an a that is not part of a cluster
- w is a cluster starting with a
- w starts with a cluster beginning with a

This results in the following:

$$\begin{aligned} M_a &= a \cup aM \cup C_a \cup C_aM, and \\ M = \cup_{a \in V} M_a &= V \cup VM \cup_{a \in V} C_a \cup_{a \in V} C_aM \end{aligned}$$

Hence,

$$\begin{aligned} f'_a &= x[a] + x[a]G + weight'(C_a)G + weight'(C_a) \\ g &= \sum_{b \in V} x[b] + \sum_{b \in V} x[b]G + \sum_{b \in V} weight'(C_b)G + \sum_{b \in V} weight'(C_b) \end{aligned}$$

By solving the second expression and substituting it into the first we obtain

$$f'_a = \frac{x[a] + weight'(C_a)}{1 - \sum_{b \in V} x[b] - \sum_{b \in V} weight'(C_b)}$$

and finally we obtain $f(s)$ by substituting this into equation 11.3.

$\square$

We will now look at an applications of this method that analyzes the relationship between vowels and consonants in a sample of English.

11.2 Vowels and Consonants

To obtain data we took the list of English words from Unix and converted them to Maple format. We substituted A for every vowel and B for every consonant. This gives us a large data set that is easily handled. From this data we were able to find the generating function produced when certain patterns are avoided. The coefficient of s^n in this generating function is the probability that a word of length n avoids the mistakes pattern as a factor.

Clearly if we do not define any bad patterns then our generating function is $-\frac{1}{s-1}$ and if we avoid both $[A]$ and $[B]$ then $f(s) = 1$, as we cannot find any words containing no vowels and no consonants. Interestingly if we avoid just vowels, $f_A(s)$, or just consonants $f_B(s)$, we do get non-trivial generating functions, and for each length we find the likelihood of finding a word with no vowels higher than for finding one with no consonants. This makes sense because there are more possible consonant pairs than vowel pairs and statistically they are more likely.

$$\begin{aligned} f_A(s) &= -\frac{1}{5017}\frac{160824891s + 383780432}{30077s - 76496} \\ f_B(s) &= -\frac{1}{5017}\frac{13756751s + 257020910}{6877s - 51230} \end{aligned}$$

We see a similar situation when we look at words avoiding $[A, A]$ ($f_{AA}(s)$) and those avoiding $[B, B]$ ($f_{BB}(s)$). Again the probability of finding a word with no vowel pairs is higher than the probability of finding one with no consonant pairs. Just looking at the previous sentence we can find very few words with vowel pairs, and almost all the words contain consonant pairs.

$$\begin{aligned} f_{AA}(s) &= -\frac{1}{5017}\frac{1105992775407s^2 + 11930653621290s + 19661071531360}{2058851907s^2 + 1540844710s - 3918890080} \\ f_{BB}(s) &= -\frac{1}{5017}\frac{413761799827s^2 + 17021813500496s + 19661071531360}{2058821907s^2 + 526062992s - 3918890080} \end{aligned}$$

The same scenario occurs when we avoid blocks of three vowels or three consonants. In fact the probability of finding a word of length n that avoids three vowels in a row is close to 1 for all n. This makes sense as it takes most people a couple of moments to think of a word with 3 vowels in a row (*conscious* is an example), but 3 consonants is not a problem (there are 3 examples in this sentence).

The results are summarized in the following table for $0 \leq n \leq 8$.

Table 11.1: Probability of avoiding vowel and consonant patterns.

	0	1	2	3	4	5	6	7
none	1	1	1	1	1	1	1	1
A	1	.812	.319	.126	.0494	.0194	.00763	.00300
B	1	.188	.0252	.00338	.000454	.0000610	$.818 * 10^{-5}$	$.110 * 10^{-5}$
A A	1	1	.975	.909	.869	.819	.779	.737
B B	1	1	.681	.617	.440	.383	.283	.239
A A A	1	1	1	.997	.988	.982	.975	.969
B B B	1	1	1	.874	.849	.780	.732	.683

CHAPTER 12

CYCLIC WORDS

Previously we have only been looking at linear words. Those are words for which there is no interaction between the end of the word and the beginning. For cyclic words on the other hand there is such an interaction. They are like necklaces.

In order to explore such words we will write them as a linear word (like an unclasped necklace), but we will have to analyze the interaction between the start and the end of the word.

12.1 The Naive Approach

In the linear case the Naive Approach required tagging words by their endings (see [NZ99]). For the cyclic words we will tag words by both their beginning terms and their end terms.

For example if we are looking at sub-blocks of length 2 we would say $[a, n, n, e] \in$

{words that begin with$[a, n]$and end with$[n, e]$}.

We let $W[[w_1, \dots, w_k], [v_1, \dots, v_k]]$ represent the linear weight of all words that begin with $[w_1, \dots, w_k]$ and end with $[v_1, \dots, v_k]$. Then it is easy to see that

$$W[[w_1, \dots, w_k], [v_1, \dots, v_k]] = \sum_{\alpha \in V, \beta \in V} W[[w_2, \dots, w_k, \alpha], [\beta, v_1, \dots, v_{k-1}]] * \prod_{i=1}^{k} x[w_1, \dots, w_i] x[v_i, \dots v_k] + \text{initial terms.}$$

Where the initial terms are: $x[w_1, \dots w_k]$ if $[w_1, \dots w_k] = [v_1, \dots v_k]$, and $x[w_1, \dots w_k, v_k]$ if $[w_2, \dots w_k] = [v_1, \dots v_{k-1}]$

We then solve this system of equations and obtain the generating function in the following way.

$$gf := \sum_{w,v \in V^* : l(w) = l(v) = k} W[w, v] * \mathit{overlap}(w, v) + (\text{terms of length} < k)$$

Here the *overlap*(w, v) refers to the weight caused by doing up the necklace.

$\mathit{overlap}(w, v) = \prod_{i=2}^{k} x[v_i, \dots, v_k, w_1, \dots, w_{i-1}]$

As in the linear case we are not taking advantage of the fact that we avoid the bad blocks until the end.

12.2 The Edlin Zeilberger Extension

Here we take advantage of the Goulden-Cluster Method to do most of the work for us. When looking at cyclic words which contain bad clusters there are three possibilities.

Case 1: The cluster does not cross the "invisible" clasp of the necklace. That is that the cluster does not contain a mistake that contains both the first and last letter of the word. This is equivalent to a linear cluster as discussed in Chapter 1, as if we undid the clasp it would not affect the cluster. This gives us the generating function

$$\frac{1}{1-ks-L} \tag{12.1}$$

where $L = weight(Clusters)$.

Case 2: Here the cluster may cross the clasp, but it does not make it all the way around, so that we can break the necklace at some point without breaking the cluster. This is then a translation of the first case and it can easily be seen that in this case the generating function is given by

$$\frac{sL'-L}{1-ks-L} \tag{12.2}$$

Case 3: This is the case that truly extends beyond the Goulden-Jackson Method. We now look at clusters that wrap all the way around the necklace. It is impossible to break the necklace anywhere without breaking the cluster. To count these cluster we set up a matrix A that shows the interaction between each of the bad words. That is $A[i, j]$=the sum of the weights of all possible overlaps from mistake i to mistake j. Now in order to make sure that no cluster is counted twice we must find a way to identify the "first" mistake in the cluster. We do this by labeling the first letter of the imbedded word and it can only be one of the letters that stick out from the last mistake. For example if the end of the cluster was

a n n e
e d l i n then any of the last four letters could be the initial letter of the word. In general this will result in m possible starting points if the word provides additional weight of s^m to the cluster, or more simply $s\frac{d}{ds}(weight)$.

From this we obtain a new matrix B in the following way:

$$\begin{aligned} B &= \sum_{r=1}^{\infty} A^r \frac{d}{ds} A & (12.3) \\ &= \frac{A}{I-A} s \frac{d}{ds} A & (12.4) \end{aligned}$$

.

The last step is to remove those words that are too short. For example if one of the mistakes is $[1,1,1]$ then we will get $[1,1]$ as a bad word, because at this point the procedure is unable to realize that it has counted the same 1 twice. To deal with this we simply remove the lower terms of the power series of all the diagonal terms (these are the ones that start and finish at the same mistake, and so the ones we are interested in). Let us define a function $Chop_r$ that does this. If our mistake i has length l_i then

$$Chop_r(mistake_i) = Chop_r(\sum_{t=0}^{\infty} a_t s^t) = \sum_{t=l_i}^{\infty} a_t s^t \qquad (12.5)$$

This results in the generating function for case 3 of

$$\sum_{i=1}^{n} Chop_{l_i} M_{i,i} \qquad (12.6)$$

Theorem 12.1 *The generating function for cyclic words whose first letter is marked* *over a k letter alphabet and which avoid a set of n mistakes as factors is given by:*

$$\frac{1+sL'-L}{1-ks-L}+\sum_{i=1}^{n} Chop_{l_i} M_{i,i}. \tag{12.7}$$

Proof: Combine Equations 12.1, 12.2 and 12.6.

□

12.3 Cyclic Burstein-Wilf

The motivation for finding general formulas for cyclic words with labeled first letter comes from Burstein and Wilf's wonderful discovery of a general formula for words avoiding blocks of the form α^{w+1} [BW97]. Their formula is:

$$f_w^k(s)=\frac{1-s^w}{1-s}(ks+(k-1)s(\frac{w+1-wks}{1-ks+(k-1)s^{w+1}}-\frac{w+1}{1-s^{w+1}})) \tag{12.8}$$

You will recall in Chapter 5 we looked at the linear analog of this. The package CGJ which implements the above method can verify Equation 12.8 for any specific k. By performing the work by hand it also possible to verify for general k using the above method for Case 1 and 2 and some thought for case 3.

12.4 Cyclic $\alpha^m\beta^n$

Recall from Chapter 5 that we are trying to avoid blocks of m α's followed by n β's in this example. The linear case was discussed in that chapter, here we look at the cyclic case.

The Maple Package CGJ was used to obtain formulas for all triples of $2 \leq m, n, k \leq 5$. First formulas for fixed k were deduced and then from these a formula for general k was conjectured as

$$
\begin{aligned}
f^k_{m,n}(s) \;=\; & -\Big(\Big(\sum_{t=1}^{\min(n,m)-1} (k(k-1)^2 t(m+n) \\
& -(\frac{1}{2}(k-1)^2kt + \frac{1}{2}(k-1^2)(k+2))t)s^{2m+2n-t-1}) \\
& +(\sum_{t=1}^{\min(m,n)-2} (((k-1)^2kt - k(k-1))\max(n,m) \\
& +(\frac{1}{2}(k-1)^2kt - \frac{1}{2}(k-1)^2(k+2))t + k - 1)s^{m+n+t}) \\
& +k(k-1)(m+n-1)s^{m+n} \\
& -(k-1)(\max(n,m)-1) + (k-1)^2)s^{2\max(n,m)} \\
& -(k-2)(\sum_{t=1}^{\min(m,n)} s^{m+n-t}) - 1 \\
& +(\sum_{t=1}^{\min(m,n)-1} s^{2\max(n,m)+t}(((k-1)^2kt - k(k-1))\max(m,n) \\
& +(k-1)(\frac{1}{2}k(k-1)t^2 - (\frac{1}{2}k(k+1) - 1)t + 1))) \\
& -(\sum_{t=1}^{\min(m,n)-2} s^{m+n+t}(((k-1)^2kt - k(k-1))\max(m,n) \\
& +(k-1)(\frac{1}{2}k(k-1)t^2 - (\frac{1}{2}k)k+1) - 1)t + 1)))) \\
& /(((\sum_{t=1}^{\min(m,n)} s^{m+n-t}) - 1) \\
& ((k-1)^2(\sum_{t=1}^{\min(m,n)-1} s^{m+n-t}) - (k-1)s^{\max(m,n)} - 1 + ks))
\end{aligned}
$$

The situation for Case 1 and 2 is fairly easy to prove, in fact most of the work for it is done in the linear case. Case 3 is more difficult and currently is purely conjecture.

CHAPTER 13

THE LAST WORD

In this dissertation we have looked at several classes of words from both a combinatorial perspective and a statistical viewpoint. As there is no general way to count words that avoid infinite sets of mistakes the knowledge of the sequences discussed is limited by computer memory and the efficiency of the algorithm used. This means that there is still much to learn about these topics.

Self-avoiding walks are studied by both mathematicians and physicist and the exact value of μ is regularly being refined. In the case of cyclic sequences the work has only just begun, and there is much more exploration to be done using GJcyc as a starting point. It is hoped that more general equations of the Burstein-Wilf type will be produced by further study.

In this dissertation we have only discussed one application to a normal language (Probability of vowel and consonant runs in English), but there is much more to be studied, and the relations between mathematics and the field of statistical linguistics have not been made. There is also the relationship between formal languages [RE83] and the mathematical objects to be explored.

So many objects can be considered as words from our DNA to the structure of crystals that their study can help us learn much about the world around us, and though some patterns have yet to find real world applications in these days when more words are transferred by zeroes and ones than by letters it is only a matter of time.

REFERENCES

[ALM93] Alm, S.E. 1993. *Upper Bounds for the Connective Constant of Self-Avoiding Walks.* Combinatorics, Probability and Computing **2**, 115-136.

[BRA83] Brandenburg, F.J. 1983. *Uniformly Growing k-th Power-Free Homomorphisms.* Theoretical Computer Science **23**, 69-82.

[BRI83] Brinkhuis, J. 1983. *Non-Repetitive Sequences on Three Symbols.* Quart. J. Math. Oxford **2**, no. 34: 145-149.

[BW97] Burstein, A., and Wilf, H.S. 1997. *On Cyclic Strings Without Long Constant Blocks.* Fibonacci Quart. **35**, no. 3: 240-247.

[FI99] Finch, S. 1999. *Pattern-Free Word Constants.* http://www.mathsoft.com/asolve/constant/words.words.html

[GJ79] Goulden, I.P., and Jackson, D.M. 1979. *An Inversion Theorem for Cluster Decompositions of sequences with Distinguished Subsequences.* J. London Math. Soc. **2**, no. 20: 567-576.

[MS96] Madras, N., and Slade, G. 1996. *The Self-Avoiding Walk.* Birkhauser, Boston.

[NZ99] Noonan, J., and Zeilberger, D. 1999. *The Goulden-Jackson Cluster Method: Extensions, Applications and Implementations.* J. Difference Eq. Appl. **5** : 355-377.

[RE83] Révész,G. E. 1983. *Introduction to Formal Languages.* Dover, New York.

[ZE98] Zeilberger, D., Ekhad, S.B. 1998. *There are more than $2^{n/17}$ n-letter Ternary Square-Free Words.* Journal of Integer Sequences. **1** : Article 98.1.9.