Counting, Adding, and Regular Languages

by

Thomas Finn Lidbetter

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

Waterloo, Ontario, Canada, 2018

Statement of Contributions

This thesis contains content from two papers of which I, Thomas Finn Lidbetter, am a co-author. The two papers are listed in the References section as [5, 6] and [11, 12]—the conference proceeding version and unpublished extended version pairs. The content in Sections 2.3, 2.5.7, and 2.4 is taken verbatim from [5] and [6]. The content of Section 3.1, all of Section 3.2, and Section 3.3 can be found verbatim in [11, 12]. The abstract for this thesis uses parts of the abstracts for the two aforementioned papers.

Abstract

In this thesis we consider two mostly disjoint topics in formal language theory that both involve the study and use of regular languages.

The first topic lies in the intersection of automata theory and additive number theory. We introduce a method of producing results in additive number theory, relying on theorem-proving software and an approximation technique. As an example of the method, we prove that every natural number greater than 25 can be written as the sum of at most 3 natural numbers whose canonical base-2 representations have an equal number of 0's and 1's. We prove analogous results about similarly defined sets using the automata theory approach, but also give proofs using more "traditional" approaches.

The second topic is the study languages defined by criteria involving the number of occurrences of a particular pair of words within other words. That is, we consider languages of words z defined with respect to words x, y where z has the same number of occurrences (resp., fewer occurrences), (resp., fewer occurrences or the same number of occurrences) of x as a subword of z and y as a subword of z. We give a necessary and sufficient condition on when such languages are regular, and show how to check this condition efficiently.

We conclude by briefly considering ideas tying the two topics together.

Acknowledgements

I would like to express my thanks to Jeffrey Shallit for supervising my research, especially for suggesting many interesting problems to think about and for providing valuable feedback on my mathematical writing. I would also like to acknowledge my co-authors, Jason Bell, Charles Colbourn, Ryan Dougherty, and Jeffrey Shallit, for their work and ideas that went into the two papers serving as a basis for this thesis. Lastly, I would also like to express my gratitude to Bin Ma and Éric Schost for their role as readers of this thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

This thesis examines two mostly disjoint topics involving the study of regular languages within the larger subject area of formal language and automata theory. In Chapter 2 we introduce and apply a method for proving results in number theory, using regular languages and automata theory. As an example of the method, we show that every natural number greater than 25 can be written as the sum of at most three numbers that have an equal number of 0's and 1's when written in their canonical base-2 representations. This chapter is largely based upon the work in [5, 6], of which the author of this thesis is a coauthor. Note that some of the content of Chapter 2 is copied verbatim from [5, 6]. See the Statement of Contributions in the front matter, or the final paragraph of Section 2.2.2 for a more detailed description of which parts of Chapter 2 appeared first in [5, 6].

In Chapter 3 we examine languages where membership of a word w in one of these languages is defined by the relative number of occurrences of two words x, y within word w. In particular, we prove necessary and sufficient conditions on when such languages are regular. This chapter is largely based upon the work in [11, 12], of which the author of this thesis is a coauthor. Note that most of the content of Chapter 3 is copied verbatim from [11, 12]. See the Statement of Contributions in the front matter, or the final paragraph of Section 3.1.1 for a more detailed description of which parts of Chapter 3 appeared first in [11, 12].

We start with some preliminaries in the present chapter, where we discuss the background in formal language theory common to both topics. Here, some relevant conventions are established for the remainder of the work.

1.2 Words, Languages, and Automata

1.2.1 Words and Languages

A fundamental construct for the topics included in this work is the *word*, which is also known in the literature as a *string*. A word is a sequence of *symbols*, also called *letters*, from some set of symbols referred to as an *alphabet*. For example, where $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$ is an alphabet with symbol elements a,b,c,d, the sequence of symbols $w = \mathtt{bccab}$ is a word over Σ. A word is said to be finite if it is a finite sequence of symbols. The length of a finite word w is the number of symbols in w and is denoted by $|w|$. For a symbol, a, we use the notation $|w|_a$ to be the number of occurrences of symbol a in word w. For example, for the word $w = \mathtt{thesis}$ we have $|w| = 6$, and $|w|_{\mathtt{s}} = 2$, and $|w|_{\mathtt{e}} = 1$, and $|w|_{\mathtt{a}} = 0$. We denote the *empty word*, also called the *empty string*, by ϵ. The empty word is defined such that $|\epsilon| = 0$.

For two words $u = u_1u_1 \cdots u_m$ and $v = v_1v_2 \cdots v_n$ we can write their concatenations as $uv = u_1u_2 \cdots u_mv_1v_2 \cdots v_n$ and $vu = v_1v_2 \cdots v_nu_1u_2 \cdots u_m$. Additionally, for a word w and a nonnegative integer n, we use the notation w^n to mean w concatenated with itself n times. That is, $w^n = ww \cdots w$ (n times). A word y is a *subword*, also known as a *factor*, of w if there exist (possibly empty) words x and z such that $w = xyz$. The word y is a prefix of w if there exists a word z such that $w = yz$, and y is a suffix of w if there exists a word x such that $w = xy$. For a word $w = w_1w_2 \cdots w_n$, where each of $w_1, \ldots, w_n$ is a symbol, we use the notation $w[i]$, where i is an integer satisfying $1 \leq i \leq n$, to refer to the ith symbol of w, i.e., $w[i] = w_i$. Similarly, for $w = w_1w_2 \cdots w_n$ and integers $1 \leq i \leq j \leq n$, we define $w[i..j]$ to be the subword $w_iw_{i+1} \cdots w_j$.

Going forward, we assume a basic background in formal language theory. This background includes an understanding of regular languages, deterministic and non-deterministic finite automata (DFAs and NFAs), regular expressions, context-free languages, and push-down automata. For an introduction or refresher to these ideas see, for example, [40] or [38]. However, before proceeding we establish a few conventions related to these concepts that will be kept throughout this work.

A deterministic finite automaton M is a five-tuple $M = (Q, \Sigma, \delta, q_0, F)$, consisting of a finite nonempty set of states Q, a finite nonempty alphabet Σ, a transition function $\delta : Q \times \Sigma \to Q$, an initial state $q_0 \in Q$, and a set of *final* or *accepting* states $F \subseteq Q$. The definition of the transition function can be extended to take words in the second argument, in place of lone symbols. That is, we can define $\delta^* : Q \times \Sigma^* \to Q$ inductively such that for all $q \in Q, u \in \Sigma^*, a \in \Sigma$ we have $\delta^*(q, \epsilon) = q$ and $\delta^*(q, ua) = \delta(\delta^*(q, u), a)$. Since for all

$q \in Q, a \in \Sigma$ we have $\delta^*(q, a) = \delta(q, a)$, we will write δ in place of δ^*, thus assuming that the definition of δ is extended to a function on a state, word pair.

A non-deterministic finite automaton N is also defined by a five-tuple $M = (Q, \Sigma, \delta, q_0, F)$, however, the transition function is given by $\delta : Q \times \Sigma \to 2^Q$. We again extend the definition of δ to the domain $Q \times \Sigma^*$ in the obvious way.

It should be assumed that any transitions omitted in a state diagram for a finite automaton lead to a non-accepting dead state, from which no final state is reachable.

Chapter 2

Additive Number Theory and Formal Languages

2.1 Automatic Sequences and First-Order Logic

Before defining the motivating problem for this chapter, we first build upon the background terminology and definitions used in formal language theory stated in Chapter 1 by defining concepts relevant to topics in the intersection of additive number theory and formal language theory considered in this thesis.

2.1.1 Automatic Sequences

Intuitively, an automatic sequence is a sequence of symbols given by a function that is computed by a deterministic finite automaton that reads natural numbers expressed in some fixed base. To make this idea formal, we first define a *deterministic finite automaton with output*, or *DFAO*, as a six-tuple $M = (Q, \Sigma, \delta, q_0, \Delta, \tau)$. Here, we have that Q is a finite nonempty set of states, Σ is a finite nonempty input alphabet, $\delta : Q \times \Sigma \to Q$ is a transition function extended to $\delta : Q \times \Sigma^* \to Q$ as in the case of DFAs, q_0 is an initial state, Δ is a finite nonempty output alphabet, and $\tau : Q \to \Delta$ is the output function mapping states to symbols from Δ. In contrast with a DFA, which either accepts or rejects an input word $x \in \Sigma^*$, according to whether or not $\delta(q_0, x) \in F$, a DFAO outputs a symbol from Δ on reading an input word. The output corresponding to input word $x \in \Sigma^*$ is given by $\tau(\delta(q_0, x))$. A DFAO is a generalization of the definition for a DFA $M = (Q, \Sigma, \delta, q_0, F)$ in the following sense. If we define output alphabet $\Delta = \{0, 1\}$ and output function

$$\tau(q) = \begin{cases} 1, & \text{if } q \in F; \\ 0, & \text{otherwise;} \end{cases}$$

then for $M_O = (Q, \Sigma, \delta, q_0, \Delta, \tau)$ we have $L(M) = \{x \in \Sigma^* \ : \ \tau(\delta(q_0, x)) = 1\}$.

For an integer $k \geq 1$ we use the notation Σ_k to refer to the alphabet of the k smallest natural numbers $\Sigma_k = \{0, 1, \ldots, k-1\}$ and we say that a DFAO is a *k-DFAO* if the input alphabet is Σ_k. Additionally, for a natural number n, we use the notation $(n)_k$ to mean the canonical base-k representation of n and for a word $w \in \Sigma_k^*$ we use the notation $[w]_k$ to refer to the number given by interpreting w as a base-k integer. This allows us to provide a definition for a k-automatic sequence, as given in [2]. We say that the sequence $(a_n)_{n\geq 0}$ over a finite alphabet Δ is *k-automatic* if there exists a k-DFAO $M = (Q, \Sigma_k, \delta, q_0, \Delta, \tau)$ such that $a_n = \tau(\delta(q_0, w))$ for all $n \geq 0$ and all w with $[w]_k = n$. If there exists such an M for the sequence $(a_n)_{n\geq 0}$, then we say that M generates $(a_n)_{n\geq 0}$. An important feature of this definition is that the k-DFAO must generate the correct output independent of the number of leading zeros in the input words. From this, we say that a *set* $S \subseteq \mathbb{N}$ is k-automatic if the characteristic function

$$\chi_S(n) = \begin{cases} 1, & \text{if } n \in S; \\ 0, & \text{otherwise;} \end{cases}$$

defines a k-automatic sequence.

As an example of a 2-automatic sequence, consider the Thue-Morse sequence $\mathbf{t} = (t_n)_{n\geq 0}$, where the nth term, t_n, is given by $t_n = |(n)_2|_1 \bmod 2$, the number of 1's (mod 2) in the base-2 representation of n. The first 15 terms are given in Table 2.1. We can see that this sequence is 2-automatic by observing that it is generated by the 2-DFAO, M, in Figure 2.1. In M, an odd number of 1's read from input corresponds to state q_1, whereas an even number of 1's read corresponds to state q_0.

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$(n)_2$	ϵ	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110
t_n	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1

Table 2.1: The first 15 terms of the Thue-Morse sequence

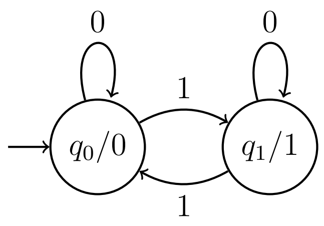
$$\begin{tikzcd}
\arrow[r] & q_0/0 \arrow[loop above, "0"] \arrow[r, bend left, "1"] & q_1/1 \arrow[loop above, "0"] \arrow[l, bend left, "1"]
\end{tikzcd}$$

Figure 2.1: A 2-DFAO, M, generating the Thue-Morse sequence

A survey of results related to the Thue-Morse sequence can be found in [1], and further examples of automatic sequences are given in [2], for example.

2.1.2 First-Order Logic

To automate the proofs of the main results of this chapter, we use an approach where the statements that we wish to prove are written as *first-order logic sentences*. In this section we define the relevant notions in first-order logic and discuss the mechanism through which automata can be used to *decide* statements written in the first-order logical theory of the natural numbers with addition and indexing into automatic sequences.

In this work we are interested in working with the *structure* $\langle \mathbb{N}, + \rangle$, describing the valid *first-order logic formulae*. To define a formula in $\langle \mathbb{N}, + \rangle$, we first define a *term* in $\langle \mathbb{N}, + \rangle$ as any variable x over $\mathbb{N}$ or any summation $x + y$ of variables x, y over $\mathbb{N}$. Using the existential ($\exists$) and universal ($\forall$) quantifiers, equality comparison ($=$), and the logical operators of negation ($\neg$), conjunction ($\wedge$), and disjunction ($\vee$), we can give the following recursive definition of a first-order logic formula.

1. For any term t and variable z over $\mathbb{N}$, we have that $t = z$ is a formula.
2. For formulae φ, ψ, we have $\neg\varphi, \varphi \wedge \psi$, and $\varphi \vee \psi$ are all formulae.
3. For a formula φ and a variable x, we have that $(\forall x)\varphi$ and $(\exists x)\varphi$ are both formulae. In both cases, we say that variable x is quantified in φ.

In a formula, φ, variables which are not quantified are said to be *free* variables. A formula is a *sentence* if there are no free variables. All such formulae evaluate to either true or false. The set of all first-order sentences in $\langle \mathbb{N}, + \rangle$ that evaluate to true is called the *first-order theory* of $\langle \mathbb{N}, + \rangle$ and is written $\mathrm{Th}(\langle \mathbb{N}, + \rangle)$. A sentence is said to be *decidable* if there is an algorithm that determines the truth value of the sentence. A first-order theory $\mathrm{Th}(\mathcal{S})$ is decidable if every first-order sentence in $\mathcal{S}$ is decidable.

Within this structure of $\langle \mathbb{N}, + \rangle$ we can define relations beyond just the additive relation $x + y = z$. For example, for variables x, z over $\mathbb{N}$ we can define the 'less than' relation $x < y$ with the formula $(\forall z)\neg(y + z = x)$. From this, we can create a simple definition for the assignment of a variable, x, to the constant 0 with the formula $(\forall y)(x < y) \vee (x = y)$. This idea can be extended to define any constant $c \in \mathbb{N}$. Given that constants and the comparison relations $<, \leq, >, \geq, \neq$ can each be defined by formulae in $\langle \mathbb{N}, + \rangle$, we will use these symbols in first-order logic formulae in $\langle \mathbb{N}, + \rangle$ for notational convenience. The same goes for other logical connectives $\Rightarrow, \equiv$ that are definable using $\vee, \wedge, \neg$.

The first-order logical theory of the natural numbers with addition, $\mathrm{Th}(\langle \mathbb{N}, + \rangle)$, was first proven to be decidable by Presburger in [32, 33]. For this, it is also known as *Presburger arithmetic*. Later, alternative proofs using finite automata were given by Büchi [7, 8], Elgot [15], and Hodgson [22]. It is this automaton-based approach to deciding first-order logic statements that will be used to obtain the main results of this chapter. The following section describes how automata can be used to decide first-order logic statements. It also introduces the software implementation of this method, `Walnut` [29], used to produce and verify our results.

2.1.3 Decidability and `Walnut`

To work towards proving the decidability of $\mathrm{Th}(\langle \mathbb{N}, + \rangle)$, the idea is to use automata A_k and E_k, where A_k recognizes triples of words x, y, z over Σ_k, where x, y, z have the same length and $[x]_k + [y]_k = [z]_k$, and E_k recognizes pairs of words x, y over Σ_k where $|x| = |y|$ and $[x]_k = [y]_k$, i.e., $x = y$. This is achieved by defining A_k and E_k over the alphabets $\Sigma_k \times \Sigma_k \times \Sigma_k$ and $\Sigma_k \times \Sigma_k$, respectively. In order to refer to the sequence of symbols in Σ_k, in some common index of tuples in $(\Sigma_k)^m$ concatenated to form a word $w = [a_{1,1}, a_{1,2}, \ldots, a_{1,m}][a_{2,1}, a_{2,2}, \ldots, a_{2,m}] \cdots [a_{n,1}, a_{n,2}, \ldots, a_{n,m}]$ in $((\Sigma_k)^m)^*$, define $w(i) = a_{1,i}a_{2,i} \cdots a_{n,i}$. This work is mostly restricted to using base 2. Hence, we take $k = 2$ for the remainder of this discussion. The automaton A_2 is given in Figure 2.2 and E_2 is given in Figure 2.3.

In automata A_2 and E_2, each index of the tuples in the input alphabets corresponds to a non-quantified variable in formulae $x + y = z$ and $x = y$ respectively. By performing automaton operations on A_2 and E_2, we can construct an automaton accepting words corresponding to valid assignments of variables to values in $\mathbb{N}$ for some first-order logic formula φ in $\langle \mathbb{N}, + \rangle$. We now give an outline of these operations on automata and how they correspond to operations used in first-order formulae.

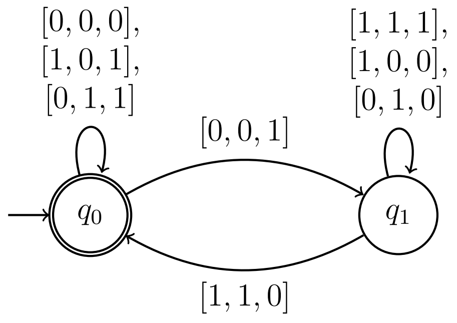

Figure 2.2: Automaton A_2 accepting words $w \in ((\Sigma_2)^3)^*$ with $w(1) = x$, $w(2) = y$, and $w(3) = z$ such that $[x]_2 + [y]_2 = [z]_2$

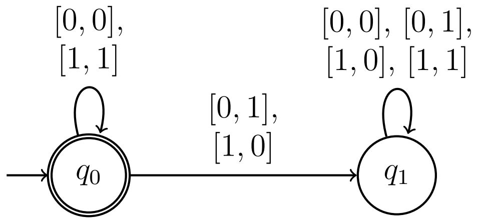

Figure 2.3: Automaton E_2 accepting words $w \in (\Sigma_2 \times \Sigma_2)^*$ such that for $x = w(1)$ and $y = w(2)$ we have $x = y$

Define the function A mapping first-order formulae to their corresponding automata. For example, for $\varphi := (x + y = z)$ and $\psi := (x = y)$ we have $A(\varphi) = A_2$ and $A(\psi) = E_2$. Observe that A_2 and E_2 correspond to the two possible types of formula defined in item 1 of the recursive definition for a first-order logic formula. Given a formula φ with corresponding automaton $A(\varphi) = M$, we can define an automaton $A(\neg\varphi)$ by completing the automaton M (possibly requiring the introduction of an explicit dead state), and then switching the roles of final and non-final states in M.

To obtain an automaton corresponding to the conjunction or disjunction of formula φ with free variables $x_1, \dots, x_m$ and formula ψ with free variables $y_1, \dots, y_n$ it is necessary to first transform $A(\varphi)$ and $A(\psi)$ such that they are defined with respect to the same ordered set of variables. This may require extending the alphabets of the automata, reordering the indices of the tuples corresponding to variables, and redefining the transition function accordingly. The details of this transformation are described, for example, in [37, Section 3.2] and [29, Section 4.1]. After applying the transformation, $A(\varphi \wedge \psi)$ and $A(\varphi \vee \psi)$ are given by the intersection and union, respectively, of the transformed automata.

To complete the demonstration of how to construct an automaton corresponding to any formula φ, it is sufficient to show how to construct an automaton for $(\exists x)\varphi$ given an automaton $A(\varphi)$. This is because if we are able to construct an automaton for $(\exists x)\varphi$, then we can construct an automaton for $(\forall x)\varphi$ by defining $A((\forall x)\varphi) = A(\neg((\exists x)\neg\varphi))$. So let $A(\varphi) = M_\varphi = (Q_\varphi, (\Sigma_2)^m, \delta_\varphi, q_\varphi, F_\varphi)$ and suppose that φ has free variables $x_1, \ldots, x_m$. If x is not a free variable in φ, then $A((\exists x)\varphi) = A(\varphi)$. So suppose that x is a free variable in φ, with $x = x_i$ for some i with $1 \le i \le m$. If $m = 1$ then define

$$A((\exists x)\varphi) = \begin{cases} (\{q_0\}, \Sigma_2, \delta, q_0, \{q_0\}), & \text{if there is a word } w \in \Sigma_2^* \text{ with } \delta_\varphi(q_\varphi, w) \in F_\varphi; \\ (\{q_0\}, \Sigma_2, \delta, q_0, \emptyset), & \text{otherwise.} \end{cases}$$

Otherwise, define non-deterministic automaton $M'_\varphi = (Q_\varphi, (\Sigma_2)^{m-1}, \delta'_\varphi, F_\varphi)$ where $\delta'_\varphi : Q_\varphi \times (\Sigma_2)^{m-1} \to 2^{Q_\varphi}$ is defined by

$$\begin{aligned} \delta'_\varphi(q, [a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_m]) = &\{\delta_\varphi(q, [a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_m]), \\ &\delta_\varphi(q, [a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_m])\}. \end{aligned}$$

It then remains to determinize M'_φ and perform an additional correcting step to ensure that if the word $0^k w$ is accepted by M'_φ for some $k \ge 0$ then the word $0^\ell w$ is accepted for all $\ell \ge 0$. The details of this additional step and an example of when this is necessary can be found in [29, Section 4.2]. Due to this determinization step, this operation can have a worst case asymptotic complexity of $O(2^n)$, where n is the number of states of M'_φ.

A further advantage of working with automata corresponding to first-order formulae is that it gives rise to a set of functions that can be added to the structure $\langle \mathbb{N}, + \rangle$ while maintaining decidability. Namely, any function $f : \Sigma_2^* \to \{0, 1\}$ where 0 corresponds to False, and 1 corresponds to True and $f = \tau(\delta(q_0, w))$ for some τ, δ defined in a 2-DFAO $M_f = (Q, \Sigma_2, \delta, q_0, \{0, 1\}, \tau)$. Let $\mathcal{F}$ be the set of all such functions. Then for all functions $f \in \mathcal{F}$ and every variable x over $\mathbb{N}$, we have that $f(x) = 1$ is a formula in $\langle \mathbb{N}, +, \mathcal{F} \rangle$.

We can construct an automaton corresponding to a formula φ of the form $f(x) = 1$ by taking the 2-DFAO $M_f = (Q, \Sigma_2, \delta, q_0, \{0, 1\}, \tau)$ and defining automaton $M_\varphi = (Q, \Sigma_2, \delta, q_0, F_\varphi)$ where $F_\varphi = \{q \in Q \ : \ \tau(q) = 1\}$. Then the decidability of $\mathrm{Th}(\langle \mathbb{N}, +, \mathcal{F} \rangle)$ follows from the argument for the decidability of $\mathrm{Th}(\langle \mathbb{N}, + \rangle)$ as given above.

All of these operations, and some others that are definable with those described above, are implemented in the software package `Walnut`, developed by Hamoon Mousavi [29]. For a more complete description of the implementation details of `Walnut` and the syntactic translation of statements in first-order logic to commands that can be parsed by `Walnut`, refer to [29] and the code repository publicly available at `https://github.com/hamousavi/Walnut` at the time of this writing.

2.2 Additive Number Theory Background

2.2.1 Introduction

The principal problem in additive number theory, as stated in [30], is that of determining whether for some set $S \subset \mathbb{N}$ there exists a constant h such that every natural number, or every sufficiently large natural number, can be written as the sum of at most h elements of S. If such an h exists, then we are interested in finding the smallest such value for h. A famous example of a problem in additive number theory is given in Lagrange's four-square theorem, which states that every natural number is the sum of at most four square numbers. We say that set $S \subset \mathbb{N}$ is an *additive basis* of order h if every natural number can be written as the sum of at most h elements of S, whereas S is an *asymptotic additive basis* of order h if there exists $M \in \mathbb{N}$ such that every integer $n \geq M$ can be written as the sum of at most h elements of S. Lagrange's four-square theorem then states that the set of square numbers is an additive basis of order 4.

Some classical problems in additive number theory concern generalizations of Lagrange's theorem to other sets of kth powers and other sets of k-gonal numbers. For a fixed integer $k \geq 3$, the nth k-gonal number is given by the formula $\frac{1}{2}((k-2)n^2 - (k-4)n)$. It was stated by Fermat in 1638 and later proved by Cauchy [9] in 1813 that for all $k \geq 3$ the set of k-gonal numbers form an additive basis of order k. The generalization of Lagrange's theorem to other sets of kth powers, however, remains unsolved for almost all values of k greater than 2, in the sense that it is known that for each $k \geq 1$ there exists a constant $h(k)$ such that every natural number is the sum of at most $h(k)$ kth powers [21], but the smallest value for $h(k)$ is not known. The problem of determining exact values for $g(k)$ (resp., $G(k)$), the least integer such that the set of kth powers forms an additive basis of order $g(k)$ (resp., an asymptotic additive basis of order $G(k)$), is known as Waring's problem. A survey of results on Waring's problem can be found in [41].

Another problem of note, that is perhaps the most famous unsolved problem in additive number theory, is Goldbach's conjecture. The conjecture has a strong and a weak form. The weak form states that every integer greater than 5 can be written as the sum of 3 primes, whereas the strong formulation states that every even integer greater than 2 can be written as the sum of 2 primes.

2.2.2 Languages as Additive Bases

Recently, there has been some interest in considering sets of numbers whose base-k expansions match some particular pattern for the aforementioned principal problem in additive number theory. For example, the word analogue of Waring's problem has been considered in [24] and [27]. To frame Waring's problem in a language-theoretic setting, define a base-b kth power to be k consecutive identical blocks of digits over Σ_b without leading 0's. In [24], Kane, Sanna, and Shallit prove that for all $k \geq 2$ there exists a constant $W(k)$ such that every sufficiently large multiple of $E_k := \gcd(2^k - 1, k)$ is the sum of at most $W(k)$ binary kth powers. Madhusudan, Nowotka, Rajasekaran, and Shallit [27] prove the analogue of Lagrange's theorem for binary squares; every sufficiently large natural number is the sum of at most 4 binary squares and 3 binary square summands is not sufficient.

Another example is the sets of base-k palindromes. A word w is a *palindrome* if the reversal of w is equal to w itself. The additive number theory problem for base-k palindromes was first considered by Banks in [3], where it was shown that every natural number can be expressed as the sum of at most 49 decimal palindromes. This number of required summands was reduced to just 3 and generalized to all bases $b \geq 5$ by Cilleruelo, Luca, and Baxter in [10]. Optimal results for the remaining bases were then given by Rajasekaran, Shallit, and Smith in [34], using an approach involving an automaton A accepting the representation of certain sums of palindromes and a decision procedure characterizing the language accepted by A. Section 2.3 will describe and apply a similar method of using automata and a decision procedure to show that certain sets form additive bases of some (optimal) finite order.

Much like the sets of base-k palindromes and base-b kth powers, the sets considered in this work have a natural definition from a language-theoretic perspective. Specifically, we will work with the sets given in Table 2.2, defined by the relationships between the number of 0's and the number of 1's in the binary expansions of natural numbers. The OEIS column refers to the corresponding entry in the *On-Line Encyclopedia of Integer Sequences* [16].

It is important to note that all of the languages in Table 2.2 are context-free languages and not regular languages. The significance of this point is made apparent in Section 2.3.1.

The content that follows in Sections 2.3, 2.5.7, and 2.4 is taken verbatim from [5] and [6], the conference proceedings and unpublished extended versions of a paper for which the author of this thesis is a coauthor. Section 2.3.1 describes the method used to achieve the additive number theory results for the sets listed in Table 2.2, and Sections 2.3.2–2.3.7 describe how the method is applied to each of the sets. Section 2.5 discusses how these same results can be achieved using more "traditional" approaches, and Section 2.4 discusses the limitations of the method, particularly as it relates to classical problems in additive number theory. We conclude the chapter with an examination of data on all N-state automata that accept languages over Σ_2 that are base-2 representations of sets forming (asymptotic) additive bases for small N in Section 2.6.

Set	Language	Entry in OEIS
$S_= = \{n \in \mathbb{N} \ : \ \lvert(n)_2\rvert_0 = \lvert(n)_2\rvert_1\}$	$L_= = (S_=)_2$	A031443
$S_< = \{n \in \mathbb{N} \ : \ \lvert(n)_2\rvert_0 < \lvert(n)_2\rvert_1\}$	$L_< = (S_<)_2$	A072600
$S_\leq = \{n \in \mathbb{N} \ : \ \lvert(n)_2\rvert_0 \leq \lvert(n)_2\rvert_1\}$	$L_\leq = (S_\leq)_2$	A072601
$S_> = \{n \in \mathbb{N} \ : \ \lvert(n)_2\rvert_0 > \lvert(n)_2\rvert_1\}$	$L_> = (S_>)_2$	A072603
$S_\geq = \{n \in \mathbb{N} \ : \ \lvert(n)_2\rvert_0 \geq \lvert(n)_2\rvert_1\}$	$L_\geq = (S_\geq)_2$	A072602
$S_\neq = \{n \in \mathbb{N} \ : \ \lvert(n)_2\rvert_0 \neq \lvert(n)_2\rvert_1\}$	$L_\neq = (S_\neq)_2$	A044951

Table 2.2: Sets considered and their corresponding languages, as presented in [5, 6]

2.3 Additive Number Theory via Approximation by Regular Languages

2.3.1 Method

Suppose we want to show that a given set S of natural numbers forms an additive basis (resp., asymptotic additive basis) of order m. Instead of considering S, we consider a subset S' of S for which the set of base-k representations of its elements forms a regular language. Such a subset is necessarily k-automatic. We then show, perhaps with some small number of exceptions that typically can be handled in some other way, that S' forms an additive basis (resp., asymptotic additive basis) of order m'. Since $S' \subseteq S$, we know that $m' \geq m$. We hope that if S' is chosen appropriately, then in fact $m = m'$. This is *regular underapproximation*.

Analogously, consider a superset S'' of S (that is, a set for which $S \subseteq S''$) for which the set of base-k representations forms a regular language. We then compute the set of numbers *not* representable as a sum of m'' elements of S''. If this set is nonempty (resp., infinite), then S'', and hence S, cannot be an additive basis (resp., an asymptotic additive basis) of order m''. We hope that if S'' is chosen appropriately, then $m = m'' + 1$. This is *regular overapproximation*.

We call these two techniques together the *method of regular approximation*, and we apply it to a number of different sets that have been previously studied. In each case we are able to find the smallest m such that the set forms an additive basis (or asymptotic additive basis) of order m.

As discussed in [4], if S is a k-automatic set, then the set of representations of numbers that are the sum of m elements of S is also k-automatic. Furthermore, there is a pair of simple criteria for deciding, given a k-automatic set, whether it forms an additive basis of finite order (resp., an asymptotic additive basis) [4]. Namely, the greatest common divisor of the elements of the k-automatic set must be 1, and the k-automatic set must not be *sparse*. A set S is said to be sparse if there is not constant d such that the function $\pi_S(n) = |\{x \leq n \ : \ x \in S\}|$ is $O((\log n)^d)$. If these criteria are satisfied, there is an algorithm for determining the least m for which it forms an additive basis (resp., an asymptotic additive basis) of order m. The advantage to this approach is that all (or almost all) of the computation amounts to manipulation of automata, and hence can be computed using existing software tools. In obtaining our results, we made extensive use of two software packages: `Grail`, for turning regular expressions into automata [36], and `Walnut`, for deciding whether a given k-automatic set forms an additive basis of order m [29] (and more generally, answering first-order queries about the elements of a k-automatic set).

2.3.2 Example of the Method: the Set $S_{\geq}$

We start with a very simple example of our method, discussing the additive properties of those numbers with at least as many 0's as 1's in their base-2 expansion. The first few such numbers are

$$0, 2, 4, 8, 9, 10, 12, 16, 17, 18, 20, 24, 32, 33, 34, 35, 36, 37, 38, 40, \ldots.$$

Theorem 1. *Every natural number except* $1, 3, 5, 7$ *is the sum of at most three elements of* $S_{\geq}$.

While this result can be proved with a more "conventional" approach—see Section 2.5.1—the argument requires several special cases. Here we offer an alternative approach using our method of regular approximation.

We start by finding a regular language that is both (a) sufficiently dense and whose representations form (b) a subset of $S_{\geq}$. After a bit of experimentation, we choose the language, $R_{\geq} = 1(01 + 0)^* - (10)^*1 = (10)(10)^*(0(0 + 10)^*(1 + \epsilon) + \epsilon)$ and prove the stronger result stated in Theorem 2.

Theorem 2. *Every natural number except* $1, 3, 5, 7$ *is the sum of at most three natural numbers whose base-*2 *representations lie in the regular language* $R_{\geq} = 1(01+0)^* - (10)^*1 = (10)(10)^*(0(0 + 10)^*(1 + \epsilon) + \epsilon)$.

Proof. First, use the `Grail` command

```
   echo '0*+0*10(10)*(0(0+10)*(1+"")+"")' | retofm | fmdeterm | fmmin
| fmcomp | fmrenum > ge1
```

to create an automaton `ge1` accepting $0^* + 0^* R_{\geq}$. Here `""` is `Grail`'s way of representing the empty word ϵ. Note that every element of $R_{\geq}$ has at least as many 0's as 1's. Also, we added 0^* in two places to get all representations with leading zeros, including all representations of 0. This produces the automaton given in Figure 2.4 below.

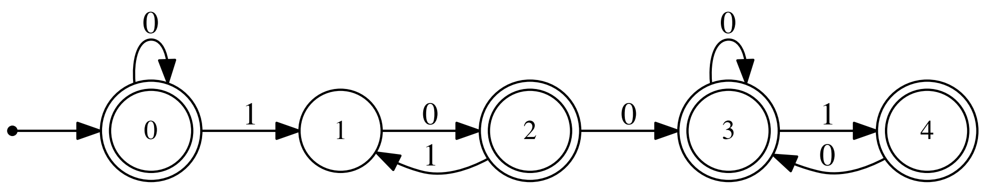

Figure 2.4: Automaton for $0^* R_{\geq}$

Next, we create the corresponding automaton `GE` in `Walnut`, and we use the `Walnut` command

```
      eval geq "E x,y,z (n=x+y+z)&(GE[x]=@1)&(GE[y]=@1)&(GE[z]=@1)":
```

giving us the automaton in Figure 2.5.

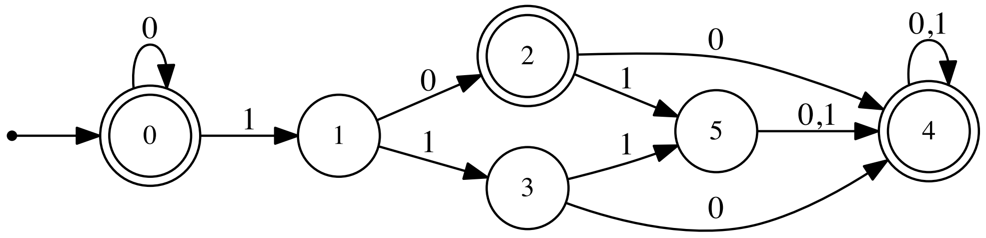

Figure 2.5: Numbers having representations as sums of at most three numbers with base-2 representations in $R_{\geq}$

By inspection we easily see that this latter automaton accepts the base-2 representation of all numbers except $1, 3, 5, 7$. This completes the proof of Theorem 2, which immediately implies Theorem 1. $\square$

We now show that the bound of 3 is optimal.

Theorem 3. *The set $S_{\geq}$ does not form an asymptotic additive basis of order* 2.

Proof. We prove that numbers of the form $2^n - 1$, $n \geq 1$, have no representation as sums of one or two elements of $S_{\geq}$. For one element it is clear. Suppose $2^n - 1 = x + y$ where $x, y \in S_{\geq}$. If both x and y have less than $n-1$ bits, then their sum is at most $2^n - 2$, a contradiction. Similarly, if both x and y have n bits, then their sum is at least 2^n, a contradiction. So without loss of generality x has n bits and y has $m < n$ bits. Since $(x)_2 \notin 1^+$, we can write $(x)_2 = 1^i 0 t$ for $i \geq 1$ and some word t of length $j = n - i - 1$. Then $(y)_2 = 1\bar{t}$. Since $y \in S_{\geq}$ we must have that $\bar{t}$ contains at least $(j+1)/2$ zeros. Then t contains at most $(j-1)/2$ zeros. Then $(x)_2$ contains at most $(j+1)/2 \leq (n-i)/2$ zeros. Since $i \geq 1$, this shows $x \notin S_{\geq}$, a contradiction. $\square$

One advantage to our method of approximation by regular languages is that it can work in cases where a conventional argument is rather complicated, as in the next section. Furthermore, the method also gives an $O(\log n)$-time algorithm to find a representation of any given n as a sum of terms of the set, although the implied constant can be rather large.

Remark 4. We can also prove that Theorem 1 holds even when the summands are required to be distinct. We can prove this using the `Walnut` command

```
eval geq2 "E x,y,z ((n=x)|(n=x+y)|(n=x+y+z))&(x<y)&(y<z)&
       (GE[x]=@1)&(GE[y]=@1)&(GE[z]=@1)":
```

2.3.3 The Set $S_=$

In this section, we discuss those numbers having an equal number of 0's and 1's in their base-2 expansion. Such numbers are sometimes called "digitally balanced".

Theorem 5. *Every natural number, except* $1, 3, 5, 7, 8, 15, 17, 25$*, is the sum of at most three elements of* $S_=$.

To prove this we prove the following stronger result.

Theorem 6. *Every natural number, except* $1, 3, 5, 7, 8, 15, 17, 25, 67$*, is the sum of at most three natural numbers whose base-2 representations lie in the regular language* $R_= = 10(10+01+1100+0011)^* + 1(10+01)^*0 + \epsilon$.

Proof. We used the `Grail` command

```
   echo '0*10(10+01+1100+0011)**+0*1(10+01)*0+0*' | retofm | fmdeterm | fmmin
| fmcomp | fmrenum > e1
```

to find the 16-state automaton below in Figure 2.6.

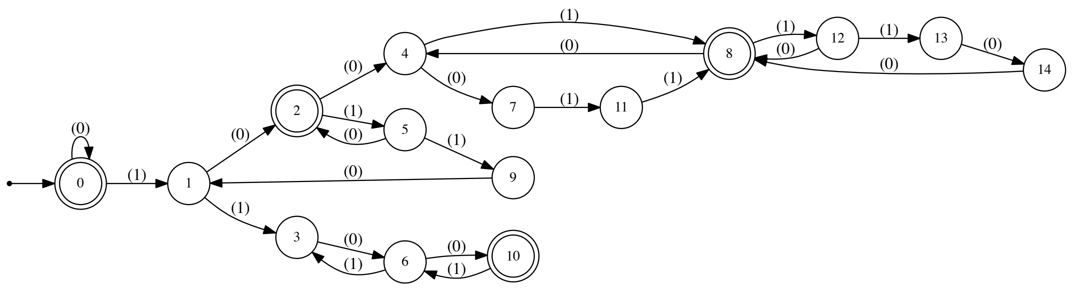

Figure 2.6: Automaton for $0^*R_=$

We then built the corresponding automatic sequence QQ in Walnut and issued the command

```
eval eqq "E x,y,z (n=x+y+z)&(QQ[x]=@1)&(QQ[y]=@1)&(QQ[z]=@1)":
```

which produced the 12-state automaton in Figure 2.7.

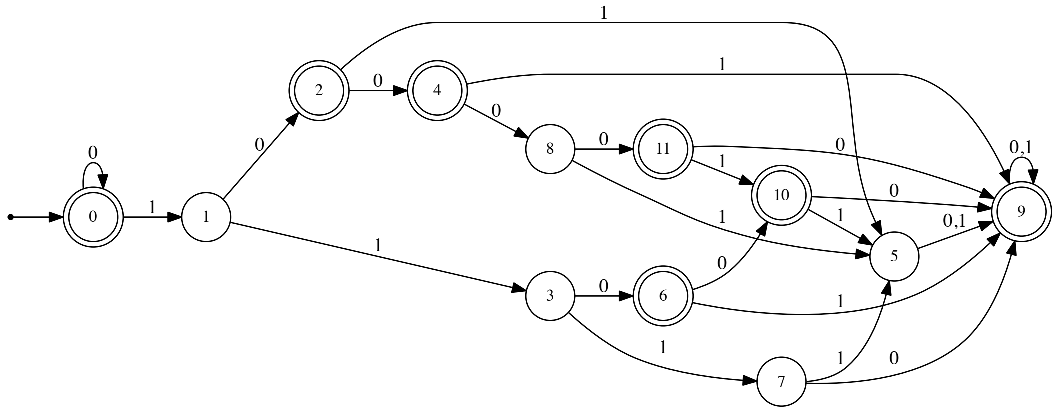

Figure 2.7: Automaton accepting those n that are the sum of at most three numbers with base-2 representations in $R_=$

The total amount of computation time here was 226497 ms, and involved the determinization of an NFA of 1790 states, so this was quite a nontrivial computation for Walnut. By inspection we easily see that this automaton accepts the base-2 representations of all integers except $1, 3, 5, 7, 8, 15, 17, 25, 67$. □

Proof of Theorem 5. We see that $10(10 + 01 + 1100 + 0011)^* + 0^*1(10 + 01)^*0$ is a regular underapproximation of $L_=$. Except for 67, the automaton we obtained matches the statement of the theorem. However, 67 has the representation $67 = 56 + 9 + 2$ in terms of elements of $S_=$. This concludes the proof. □

We now show that the bound of 3 is optimal:

Theorem 7. *There are infinitely many natural numbers that are not the sum of one or two members of $S_=$.*

Proof. We use the method of overapproximation. Consider

$$S = \{n \in \mathbb{N} \ : \ |(n)_2| \text{ is even but } n \text{ is not of the form } 2^k - 1 \ \}.$$

Then it is easy to see that $S_= \subset S$. Furthermore $(S)_2$ is regular, and represented by the regular language

$$R'_= = 1(11)^*(0 + 100 + 101)(00 + 01 + 10 + 11)^*.$$

We use `Grail` on the command

```
echo '0**+0*1(11)*(0+100+101)(00+01+10+11)*' | retofm | fmdeterm | fmmin |
fmcomp | fmrenum > ov1
```

giving us the automaton in Figure 2.8.

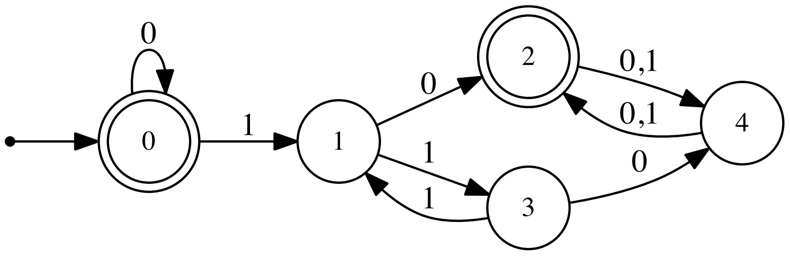

Figure 2.8: Automaton for $0^* + 0^* R'_=$

Then we ask `Walnut` to give us the base-2 representations of all number that are *not* the sum of two members of S. This gives us the automaton in Figure 2.9.

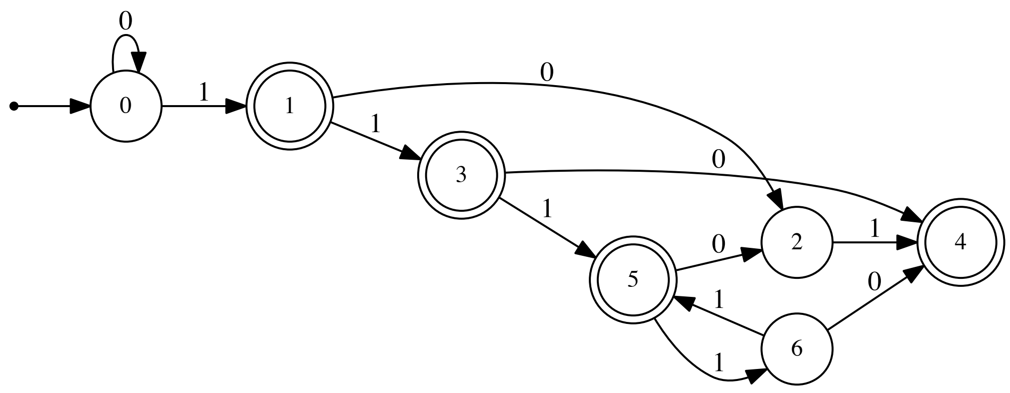
Figure 2.9: Automaton accepting base-2 representations of those n that are not the sum of at most two numbers with base-2 representations in $R'_=$

By inspection we easily see that numbers with base-2 representation $111(11)^*$ and $111(11)^*0$ have no representation. Since this set is infinite, we know that S and hence $S_=$ does not form an asymptotic additive basis of order 2.

$\square$

2.3.4 The Set $S_\leq$

Theorem 8. *Every natural number is the sum of at most* 2 *elements of* $S_\leq$. *The* 2 *is optimal.*

We prove this by means of

Theorem 9. *Every natural number is the sum of at most two natural numbers with canonical base-*2 *representation in* $(1 + 10)^*$.

Proof. We used the Grail command

```
echo '0*(1+10)*' | retofm | fmdeterm | fmmin | fmcomp | fmrenum > le1
```

obtaining the automaton below in Figure 2.10.

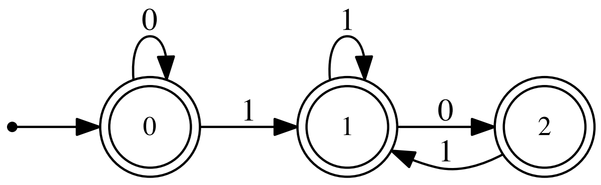

Figure 2.10: Automaton for $0^*(1+10)^*$

We then built the corresponding automatic sequence `LE` in `Walnut` and issued the command

```
eval leq "E x,y (n=x+y)&(LE[x]=@1)&(LE[y]=@1)":
```

which produces a 1-state automaton accepting everything. This concludes the proof. □

2.3.5 The Set $S_<$

Theorem 10. *Every natural number is the sum of at most* 2 *elements of* $S_<$. *The* 2 *is optimal.*

We prove this by means of

Theorem 11. *Every natural number is the sum of at most two natural numbers whose base-*2 *representations lie in* $1(1+10+01)^*$.

Proof. We use the `Grail` command

```
echo '0*+0*1(1+10+01)*' | retofm | fmdeterm | fmmin | fmcomp | fmrenum >
lt1
```

which gives the automaton below in Figure 2.11.

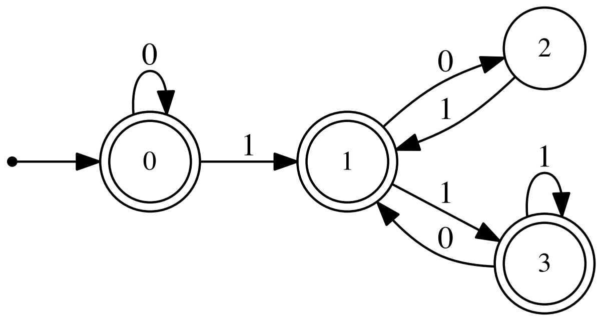

Figure 2.11: Automaton for $0^* + 0^*1(1 + 10 + 01)^*$

Next, we use the `Walnut` command

```
eval lt "E x,y (n=x+y)&(LT[x]=@1)&(LT[y]=@1)":
```

which produces a 1-state automaton accepting everything. This concludes the proof. □

2.3.6 The Set $S_>$

Theorem 12. *Every natural number, except* $1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15, 19, 23, 27, 31, 47, 63$*, is the sum of at most* 3 *elements of* $S_>$*. The* 3 *is optimal.*

We prove this by means of

Theorem 13. *Every natural number, except* $1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15, 19, 23, 27, 31, 47, 63, 79$*, is the sum of at most* 3 *numbers whose base-*2 *representation is given by* $10(01 + 10 + 0)^*0(01 + 10 + 0)^*$.

Proof. We use the grail command

```
   echo '0*+0*10(01+10+0)*0(01+10+0)*' | retofm | fmdeterm | fmmin | fmcomp
| fmrenum > gt1
```

giving us the automaton in Figure 2.12.

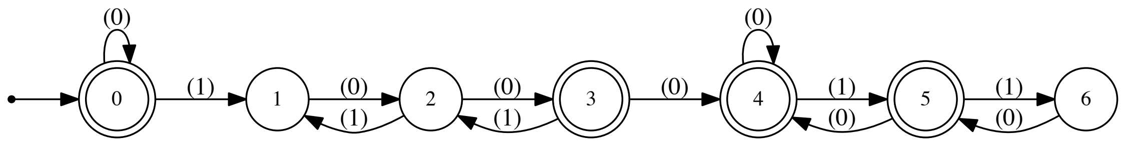

Figure 2.12: Automaton for $0^* + 0^*(10(01+10+0)^*0(01+10+0)^*)$

Then we use the `Walnut` command

```
eval grt "E x,y,z (n=x+y+z)&(GT[x]=@1)&(GT[y]=@1)&(GT[z]=@1)":
```

giving the automaton in Figure 2.13.

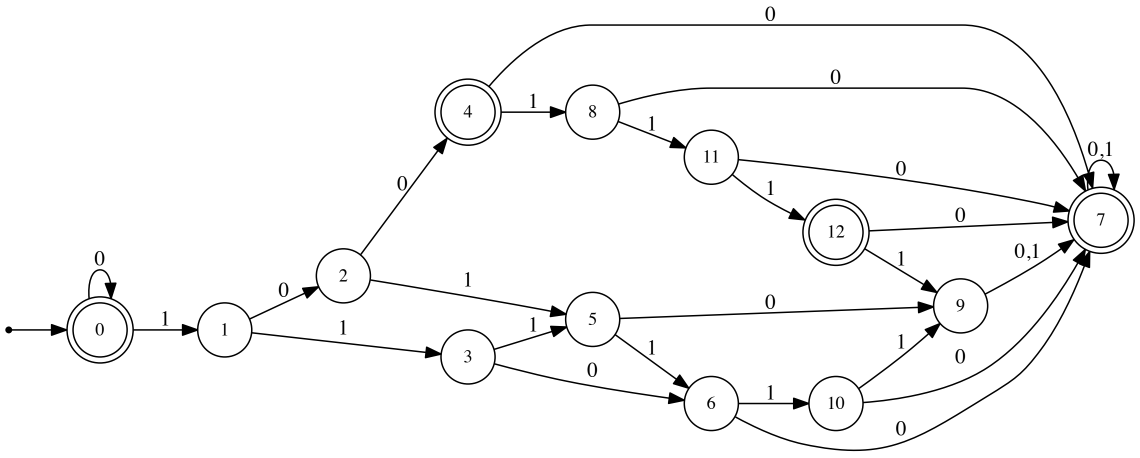

Figure 2.13: Automaton accepting those numbers that are the sum of at most three elements whose base-2 representations are in $10(01+10+0)^*0(01+10+0)^*$

$\square$

Proof of Theorem 12. After noting that 79 has the representation $79 = 4+8+67$ in terms of elements of $S_>$, Theorem 13 implies the first result. To see that two summands do not suffice, note that every element of $S_>$ is an element of $S_\geq$, and we already proved above that two summands do not suffice for $S_\geq$. $\square$

2.3.7 The Set $S_{\neq}$

Theorem 14. *Every natural number is the sum of at most two elements of $S_{\neq}$.*

Proof. We have that every natural number is the sum of at most two elements of $S_<$ by Theorem 10 and $S_< \subset S$. This implies the result. $\square$

2.3.8 The Totally Balanced Numbers

We say that a word $x \in \{0,1\}^*$ is *totally balanced* if

(a) $|x|_1 = |x|_0$; and

(b) $|x'|_1 \geq |x'|_0$ for all prefixes x' of x.

In other words, such a word is a recoding of a word consisting of balanced parentheses, where 1 represents a left parenthesis and 0 represents a right parenthesis. Given a totally balanced word x, we can define its *nesting level* $\ell(x)$ recursively as follows:

(a) $\ell(\epsilon) = 0$;

(b) If $x = 1y0z$, where both y and z are totally balanced, then $\ell(x) = \max(\ell(y)+1, \ell(z))$.

Consider the set of numbers $S_{\rm TB}$ whose base-2 representation is totally balanced; note that all such numbers are even. The elements of $S_{\rm TB}$ form sequence A014486 in the OEIS [16].

Theorem 15. *Every even number except* $8, 18, 28, 38, 40, 82, 166$ *is the sum of at most* 3 *elements of* $S_{\rm TB}$. *There are infinitely many even numbers that are not the sum of at most* 2 *elements of* $S_{\rm TB}$.

We prove the first part of Theorem 15 using the following.

Theorem 16. *Consider*

$$R_{\text{TB}} = \{x \in \{0,1\}^* \ : \ x \textit{ is totally balanced and } \ell(x) \le 3\}.$$

Then every even number except $8, 18, 28, 38, 40, 82, 166$ *is the sum of at most* 3 *natural numbers whose binary representation is contained in* R_{TB}*.*

Proof. The language R_{TB} is accepted by the following automaton.

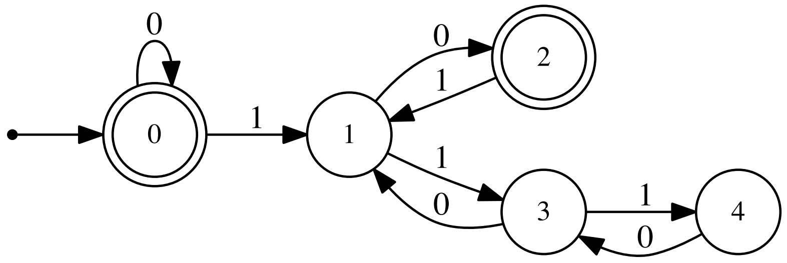

Figure 2.14: Automaton for $0^* R_{\text{TB}}$

Using the `Walnut` command

```
eval bp2 "E x,y,z (2*n=x+y+z)&(TB[x]=@1)&(TB[y]=@1)&(TB[z]=@1)":
```

we get an automaton accepting all n for which $2n$ is representable.

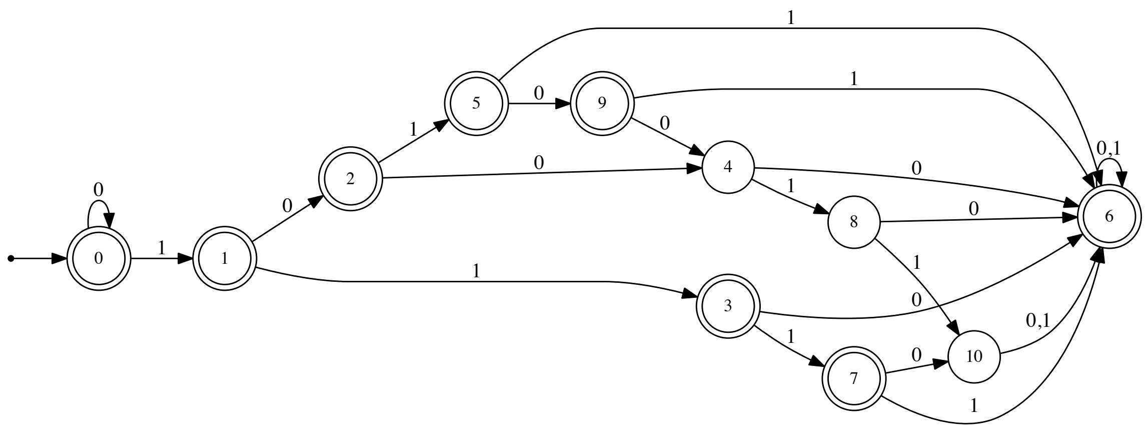

Figure 2.15: Automaton accepting the base-2 representation of those n for which $2n = x + y + z$ with x, y, z having base-2 representations in R_{TB}

By inspection is now easy to verify that the only n not accepted are $4, 9, 14, 19, 20, 41, 83$. $\square$

Proof of Theorem 15. Clearly R_{TB} is a subset of L_{TB}.

To see the second part of the theorem, note that by the proof of Theorem 7 there are infinitely many even numbers (for example, those with base-2 representation $111(11)^*0$) not representable as the sum of two elements of $S_=$, and $S_=$ is an overapproximation of S_{TB}. $\square$

2.3.9 Prefixes of Totally Balanced Numbers

At the suggestion of Georg Zetzsche, in a personal communication with Jeffrey Shallit passed on to the author of this thesis, we can also consider the set, S_{PB}, of numbers whose base-2 representations are prefixes of totally balanced words, as in Section 2.3.8. That is, the base-2 representations are the words $x \in \{0,1\}^*$ such that $|x'|_1 \geq |x'|_0$ for all prefixes x' of x. The elements of S_{PB} form sequence A061854 in the OEIS [16].

Theorem 17. *Every natural number is the sum of at most 2 elements of S_{PB}.*

Proof. Let $L_{\mathrm{PB}} = (S_{\mathrm{PB}})_2$ and observe that L_{PB} is a subset of the language given by the regular expression $1(1+10+01)^*$ as used in Theorem 11. The proof of Theorem 11 implies the result. $\square$

2.4 Limitations of the Method

It is natural to wonder whether more "traditional" problems in additive number theory can be handled by our technique. For example, suppose we try to approach Goldbach's conjecture (i.e., every even number ≥ 4 is the sum of two primes) using a regular underapproximation of the language of primes in base 2. Unfortunately, this technique is guaranteed to fail, because a classical result of Hartmanis and Shank [20] shows that every regular subset of the prime numbers is finite.

Similarly, recent results on the additive properties of palindromes (discussed in Section 2.2.2) cannot be achieved by regular approximation, because every regular language consisting solely of palindromes is *slender*: it contains at most a constant number of words of each length [23].

We could also consider Waring's problem, as discussed in Section 2.2.1, which concerns the additive properties of ordinary integer powers. However, our approach also cannot work here due to the result of Theorem 18, as proved by a co-author of [5, 6], Jason Bell. Before we can state the theorem we first recall that the k-kernel of a set $S \subseteq \mathbb{N}$ is defined to be the number of distinct subsets of the form

$$S_{e,j} = \{n \in \mathbb{N} \ : \ k^e n + j \in S \text{ for } e \geq 0 \text{ and } 0 \leq j < k^e\}.$$

Theorem 18. *Let $k \geq 2$ be a natural number, and let S be a k-automatic subset of $P := \{n^j : n, j \geq 2\}$. Then there is a finite set of integers T such that $S \subseteq \{ck^j : c \in T, j \geq 0\}$. Moreover, if the size of the k-kernel of S is d, then we can take T to be a subset of $\{0, 1, \ldots, k^d - 1\}$.*

Proof. Let S be a k-automatic subset of $P := \{n^j : n, j \geq 2\}$. We claim that for every natural number m, the set

$$S^{(m)} := \{n \in S : n \not\equiv 0 \pmod{k^m}\}$$

is finite. To see this, suppose that there is some m such that $S^{(m)}$ is infinite. Then since $S^{(m)}$ is k-automatic, by the pumping lemma it contains a set of elements of the form

$\{[xy^j z]_k \colon j \geq 0\}$. Let r, s, and t denote the lengths of x, y, and z respectively. We let $X = [x]_k$, $Y = [y]_k$, and $Z = [z]_k$. Then

$$\begin{aligned}[xy^n z]_k &= Z + k^t(Y + k^s Y + \cdots + k^{s(n-1)} Y) + k^{sn+t} X \\ &= k^{sn}\left(k^t X + \frac{k^t Y}{k^s - 1}\right) + \left(Z - \frac{k^t Y}{k^s - 1}\right).\end{aligned}$$

Then $u_n := [xy^n z]$ satisfies the two-term linear recurrence $u_n = r_1 u_{n-1} + r_2 u_{n-2}$ with $r_1 = (k^s + 1)u_{n-1}$ and $r_2 = -k^s u_{n-1}$. In particular, $r_1^2 + 4r_2^2 \neq 0$, and since k^s and 1 are nonzero and not roots of unity, we have that the recurrence is non-degenerate as long as $k^t X + Y k^t/(k^s - 1)$ and $Z - k^t Y/(k^s - 1)$ are nonzero. But since $u_n = [xy^n z]_k \to \infty$ as $n \to \infty$, we see that $k^t X + k^t Y/(k^s - 1)$ must be nonzero; since $u_n \not\equiv 0 \pmod{k^m}$, we see that $Z - k^t Y/(k^s - 1)$ is nonzero. Then, by [39, Theorem 2] we deduce that $P \cap \{u_n \colon n \geq 0\}$ is finite, a contradiction. It follows that $S^{(m)}$ is finite for every $m \geq 1$.

We now finish the proof. Let d denote the size of the k-kernel of S and let $T = S^{(d)} \cup \{0\}$. Then T is a finite set. We claim that $S \subseteq S_0 := \{ck^j \colon c \in T, j \geq 0\}$. To see this, suppose that this is not the case. Then there is some $\ell \in S \setminus S_0$. Pick the smallest natural number $\ell \notin S \setminus S_0$. Since $0 \in T$, we have that ℓ is positive. Also since $T \subseteq S_0$, we have $\ell \notin T$, and so ℓ must be divisible by k^d (since if it were not, it would be in T). Thus $k^d \mid \ell$. The k-kernel of S has size d, and so there exist $i, j \in \{0, 1, \ldots, d\}$ with $i < j$ such that

$$\{n \in \mathbb{N} \colon k^i n \in S\} = \{n \in \mathbb{N} \colon k^j n \in S\}.$$

In particular, if k^j divides n and $k^j n$ is in S, then $k^{i-j} n \in S$. Then, since $k^d \mid \ell$, we see that $k^{i-j}\ell \in S$; furthermore $k^{i-j}\ell < \ell$ since ℓ is positive. Thus, by minimality of ℓ, we have $k^{i-j}\ell \in S_0$, and since S_0 is closed under multiplication by k, we get that $\ell = k^{i-j}\ell k^{j-i}$ is also in S_0, a contradiction. The result follows. $\square$

Thus, every k-automatic subset of $P = \{n^j : \ n, j \geq 2\}$ is necessarily sparse, violating the criterion given in Theorem 1.1 of [4] for k-automatic sets forming additive bases.

2.5 Alternative Methods and Generalizations

This section describes alternative methods for proving the same results for the sets in Table 2.2. It is worth noting that while these arguments give further insight into why these sets form additive bases, the arguments themselves require a lot of case-based reasoning. This is in contrast to the automaton-based approach presented in Section 2.3, where the proofs are straightforward and have the same structure for each of the sets.

2.5.1 The Set $S_{\geq}$

Theorem 1. *Every natural number except* $1, 3, 5, 7$ *is the sum of at most three elements of* $S_{\geq}$.

Proof. Given N we want to represent, let n_1 (resp., n_2, n_3) be the integer formed by taking every 3rd 1, starting with the first 1 (resp., second 1, third 1), in the base-2 representation of N. Provided there are at least four 1's in $(N)_2$, every 1 in $(n_i)_2$, for $1 \leq i \leq 3$, is associated with at least two following zeros, except possibly the very last 1, and hence $n_i \in S_{\geq}$.

This construction can fail on odd numbers whose base-2 representation has three 1's or fewer, so we must treat those as special cases.

For numbers of the form $N = 2^i + 1$ with $i \geq 3$, we can take $n_1 = 2^i + 1$, $n_2 = n_3 = 0$.

For numbers with binary representation 10^i10^j1, we can take $n_1 = [10^{i+j+1}1]_2$, $n_2 = [10^{j+1}]_2$, $n_3 = 0$. This works provided $i + j + 1 \geq 2$ and $j + 1 \geq 1$.

This covers all cases except $N = 1, 3, 5, 7$. □

An argument for the optimality of this result not using the method of overapproximation has already been given in Section 2.3.2.

2.5.2 The Set $S_{=}$

For an alternative proof of the fact that $S_{=}$ forms an asymptotic additive basis of order 3, as in Theorem 5, we first observe that for any $n \in S_{=}$ with $n \neq 0$ we have $4n + 1 \in S_{=}$ and $4n + 2 \in S_{=}$. This allows for the following inductive argument.

Lemma 19. *Every natural number greater than or equal to 45 is the sum of 3 non-zero elements of* $S_{=}$.

Proof. We prove this result by induction. For the base cases, one can verify that each integer n satisfying $45 \leq n \leq 182$ can be written as the sum of 3 non-zero elements of $S_{=}$. Now assume that for all integers m satisfying $45 \leq m < n$, we have that m can be written as the sum of 3 non-zero elements of $S_{=}$. To show that n can be written as the sum of 3 non-zero elements of $S_{=}$ we consider four cases, noting that the cases for $45 \leq n \leq 182$ have already been resolved.

Case 1: $n \equiv 0 \pmod 4$. For $n > 182$ and $m = \frac{n-4}{4}$ we have that $45 \leq m < n$. So by the inductive hypothesis we know that m can be written as the sum of 3 non-zero integers $a, b, c \in S_=$. Now consider $(4a+1)+(4b+1)+(4c+2) = 4(a+b+c)+4 = 4m+4 = n$. Giving n as the 3 non-zero elements of $S_=$.

Case 2: $n \equiv 1 \pmod 4$. Choosing $m = \frac{n-5}{4}$ we have $45 \leq m < n$. This means we can write $m = a+b+c$ for non-zero integers $a, b, c \in S_=$. Then the sum $(4a+1)+(4b+2)+(4c+2) = 4m+5 = n$ gives n as the sum of 3 non-zero integers in $S_=$.

Case 3: $n \equiv 2 \pmod 4$. Choosing $m = \frac{n-6}{4}$ we have $45 \leq m < n$. This means we can write $m = a+b+c$ for non-zero integers $a, b, c \in S_=$. Then the sum $(4a+2)+(4b+2)+(4c+2) = 4m+6 = n$ gives n as the sum of 3 non-zero integers in $S_=$.

Case 4: $n \equiv 3 \pmod 4$. Choosing $m = \frac{n-3}{4}$ we have $45 \leq m < n$. This means we can write $m = a+b+c$ for non-zero integers $a, b, c \in S_=$. Then the sum $(4a+1)+(4b+1)+(4c+1) = 4m+3 = n$ gives n as the sum of 3 non-zero integers in $S_=$. □

Theorem 5. *Every natural number, except 1, 3, 5, 7, 8, 15, 17, 25, is the sum of at most three elements of $S_=$.*

Proof. By Lemma 19 we can write every natural number greater than or equal to 45 as the sum of three elements of $S_=$. For those natural numbers $0 \leq n \leq 44$ one can easily verify by brute force that every n except $n \in \{1, 3, 5, 7, 8, 15, 17, 25\}$ can be written as the sum of at most three elements of $S_=$. □

Similarly, we can prove the optimality of this result without the automaton-based approach.

Theorem 7. *There are infinitely many natural numbers that are not the sum of one or two members of $S_=$.*

Proof. Consider $N = 2^{2k+1}$ for some integer $k \geq 1$. Observe that the largest natural number less than or equal to N in $S_=$ is $2^k(2^{k+1}-1)$, the number with base-2 representation given by k 1's followed by k 0's. Then since $2 \cdot 2^k(2^k - 1) = 2^{k+1}(2^k-1) = 2^{2k+1} - 2^{k+1} < 2^{2k+1}$ we can conclude that 2^{2k+1} cannot be written as the sum of 2 elements of $S_=$ for all $k \geq 1$. □

2.5.3 The Set $S_>$

Theorem 12. *Every natural number, except* $1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15, 19, 23, 27, 31, 47, 63$, *is the sum of at most* 3 *elements of $S_>$. The* 3 *is optimal.*

Proof. We can use the same idea as used in the proof of Theorem 1 in Section 2.5.1. Given N we want to represent, let n_1 (resp., n_2, n_3) be the integer formed by taking every 3rd 1, starting with the first 1 (resp., second 1, third 1), in the base-2 representation of N. Provided that there are at least seven 1's in $(N)_2$, we have that $n_1, n_2, n_3 \in S_>$ and $n_1 + n_2 + n_3 = N$.

If $N \geq 2^{12}$ and $|(N)_2|_1 < 7$ then we have $N \in S_>$, giving N trivially as the sum of at most 3 elements of $S_>$.

For each $N < 2^{12}$ with $|(N)_2|_1 < 7$ that is not included in the list of exceptions in the statement of the theorem we can show by brute force that N can be written as the sum of at most 3 elements of $S_>$. This can be done, for example, by computing the values obtained by summing all possible triples n_1, n_2, n_3 of integers in $S_> \cap \{0, 1, 2, \ldots, 2^{12} - 1\}$ and observing that the only integers less than 2^{12} that cannot be attained with such a sum is exactly the set of exceptions in the statement of the theorem.

The optimality of this result follows from the argument for the optimality of 3 summands in Theorem 3. $\square$

2.5.4 The Set $S_<$

Lemma 20. *For any natural number N, let n be the least integer such that $N < 2^n$. If $N \neq 2^{n-1}$ where n is even, then for $A_N = \{s \in \mathbb{N} : N \leq s \leq N + 2^{\lceil \frac{n}{2} \rceil} - 2\}$, the intersection of $S_<$ and A_N is nonempty.*

Proof. Suppose $N \neq 2^{n-1}$ for all even n, and let $m = \lceil \frac{n}{2} \rceil$. We consider values in A_N modulo 2^m. By the pigeonhole principle there is some integer $s \in A_N$ such that either $s \equiv 2^m - 1 \pmod{2^m}$ or $s \equiv 2^m - 2 \pmod{2^m}$.

Case 1: Suppose that there is some $s \in A_N$ such that $s \equiv 2^m - 1 \pmod{2^m}$. Let w be the canonical base-2 representation of s. Then the m least significant digits of w must all be 1's. If $s < 2^m$ then $s = 2^m - 1$ and clearly $s \in S_<$. If $s \geq 2^m$ we know that $|w|_1 \geq m + 1$, as the most significant digit of w must be a 1. Furthermore, since $N < 2^n$ and $s \equiv 2^m - 1 \pmod{2^m}$ we know that $s \leq 2^n - 1$. This means that $|w| \leq n$, giving $|w|_0 \leq n - (m + 1) < m + 1 \leq |w|_1$. Therefore $s \in S_<$.

Case 2: If there is no such s satisfying the condition on Case 1 then there exists some $s \in A_N$ such that $s \equiv 2^m - 2 \pmod{2^m}$. Again, let w be the canonical base-2 representation of s. Then all but one of the m least significant digits of w are 1's. If $s < 2^m$ then $s = 2^m - 2$, giving $s \in S_<$. If $s \geq 2^m$ then the most significant digit of w is 1 and this leading digit is not one of the $m - 1$ many 1's in the m least significant digits of w. This gives $|w|_1 \geq m$.

Next, since $N < 2^n$ and $s \equiv 2^m - 2 \pmod{2^m}$ and there is no s satisfying the condition on Case 1, we get $s \leq 2^n - 2$. So $|w| \leq n$ and $|w|_0 \leq |w| - m \leq m \leq |w|_1$. The only case when $|w| - m = m$ is when $|w| = n$ and n is even. This is only possible if $N = 2^{n-1}$ for even n, which we assumed not to be the case. Therefore $s \in S_<$. □

Theorem 10. *Every natural number is the sum of at most two elements from $S_<$.*

Proof. For the natural number N, write $N = 2^n - 1 + t$ where $0 \leq t \leq 2^n$ and $n \geq 0$. Again, let $m = \lceil \frac{n}{2} \rceil$. If $t = 2^n$, then $N = 2^{n+1} - 1$ giving $N \in S_<$. If $t = 2^{n-1}$ and n is even, then $N = 2^n - 1 + 2^{n-1}$ and $|N|_0 = 1$ and $|N|_1 = n$, giving $N \in S_<$ for $n \geq 2$. So now suppose that $t < 2^n$ and $t \neq 2^{n-1}$ where n is even. Observe that every natural number in the interval $[2^n - 1 - (2^m - 2), 2^n - 1]$ is in $S_<$. This is because the $\lfloor \frac{n}{2} \rfloor$ most significant digits of the canonical base-2 representation of any number in this interval must all be 1's and at least one of the m least significant digits of the canonical base-2 representation of any number in this interval is a 1. By Lemma 20 there exists an integer c such that $0 \leq c \leq 2^m - 2$ and $t + c \in S_<$. So if we take $u = 2^n - 1 - c$ and $v = t + c$, then we have $u \in S_<$ and $v \in S_<$ and $u + v = 2^n - 1 - c + t + c = 2^n - 1 + t = N$. Therefore N can be written as the sum of at most two elements from $S_<$. □

2.5.5 The Set $S_\leq$

Theorem 8. *Every natural number is the sum of at most two elements from $S_\leq$.*

Proof. This is a direct consequence of Theorem 10, as proved in Section 2.5.4, since $S_< \subset S_\leq$. □

2.5.6 The Set $S_\neq$

Theorem 8. *Every natural number is the sum of at most two elements from $S_\neq$.*

Proof. This is a direct consequence of Theorem 10, as proved in Section 2.5.4, since $S_< \subset S_\neq$. □

2.5.7 Generalization to Other Bases

It would be nice to generalize our results on the preceding languages bases $k > 2$. However, the appropriate generalization is not completely straightforward, except in the case of $S_=$, the digitally balanced numbers in base 2. We can generalize this as follows:

$$S_{k,=} = \{n \in \mathbb{N} \ : \ |(n)_k|_i = |(n)_k|_j \text{ for all } i, j \in \Sigma_k\}.$$

Unfortunately $S_{k,=}$ does not form an additive basis in general, as the gcd of its elements equals 1 if and only if $k = 2$ or $k = 3$.

Theorem 21. *The* gcd *g_k of the elements of $S_{k,=}$ is $(k-1)/2$, if k is odd and $k-1$, if k is even.*

Proof. Let g_k be the gcd in question. Consider the two numbers with base-k representations

$$\begin{array}{c} 1\ 0\ 2\ 3\ \cdots\ (k-3)\ (k-1)\ (k-2) \\ 1\ 0\ 2\ 3\ \cdots\ (k-3)\ (k-2)\ (k-1) \end{array}$$

and take their difference. Now g_k must divide this difference, which is $k-1$.

Next, take any base-k digitally balanced number $n = \sum_{0 \le i < t} a_i k^i$ and compute it modulo $k-1$. We get $j(0 + 1 + \cdots + k - 1)$ where j is the number of occurrences of each digit. But this is $jk(k-1)/2$. It follows that

$$n \equiv \begin{cases} j(k-1)/2 \pmod{k-1}, & \text{if } k \text{ is odd;} \\ 0 \pmod{k-1}, & \text{if } k \text{ is even.} \end{cases}$$

The result follows. □

We can now prove

Theorem 22. *For each k there exists a least integer $D = D(k)$ such that every sufficiently large multiple of g_k is a sum of at most $D(k)$ members of $S_{k,=}$. Furthermore, $D(k) \ge k^{k-1} + 1$.*

Proof. Let T be the set of all words of length k containing each occurrence of Σ_k exactly once, and let U be the words in T that do not begin with the symbol 0. Then UT^* is a regular subset of the language $(S_{k,=})_k$, and is sufficiently dense that Theorem 1.1 of [4] applies.

For the lower bound, let n be the smallest element of $S_{k,=}$ of length tk for some $t \in \mathbb{N}$, and let n' be the largest element of $S_{k,=}$ of length $(t-1)k$. Then a representation of $n-1$ as the sum of elements of $S_{k,=}$ requires at least $\lceil (n-1)/n' \rceil$ summands. Now $n \geq k^{tk-1}+1$ and $n' \leq k^{(t-1)k} - 1$. It follows that $(n-1)/n' > k^{k-1}$, as desired. $\square$

With more work we can prove

Theorem 23. *All natural numbers, except* $1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 16, 17, 18, 20, 23, 24, 25, 27, 28, 29, 31, 35, 39, 46, 50, 212, 214, 216, 218, 220, 222, 224, 226, 228, 230, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 261, 262, 263, 264, 265, 267, 269, 270, 272, 273, 274, 276, 280, 284, 291, 295,$ *are the sum of at most 11 elements of* $S_{3,=}$.

We start with an observation similar to that made at the start of Section 2.5.2.

Observation 24. If N is a positive number that is base-3 digitally balanced, then $27N+5$, $27N+7$, $27N+11$, $27N+15$, $27N+19$ and $27N+21$ are each positive base-3 digitally balanced numbers. This is because we have $5 = (012)_3$, $7 = (021)_3$, $11 = (102)_3$, $15 = (120)_3$, $19 = (201)_3$, and $21 = (210)_3$.

We next show by brute force that some range of values can be written as the sum of 11 non-zero elements of $S_{3,=}$, and then use induction to get 11 non-zero summands in $S_{3,=}$ for values beyond this range.

Lemma 25. *For all* $k \in \{0, 1, 2, \ldots, 26\}$ *there exist integers* $m_1, m_2, \ldots, m_{11} \in \{5, 7, 11, 15, 19, 21\}$ *such that* $k \equiv \sum_{i=1}^{11} m_i \pmod{27}$.

Proof. Easy to verify, we can fill in all 27 cases via a simple brute-force computation. $\square$

Lemma 26. *Every natural number greater than or equal to 622 is the sum of 11 non-zero elements of* $S_{3,=}$.

Proof. We can show by brute-force checking that every integer N satisfying $622 \leq N \leq 17024$ can be written as the sum of 11 non-zero elements of $S_{3,=}$. So assume that for all integers m satisfying $622 \leq m < N$ we have that m can be written as the sum of 11 non-zero elements of $S_{3,=}$.

Suppose that $N \equiv k \pmod{27}$, for $k \in \{0, 1, 2, \ldots, 26\}$. Let $C = \sum_{i=1}^{11} c_i$ where $c_1, c_2, \ldots, c_{11} \in \{5, 7, 11, 15, 19, 21\}$ satisfy $k \equiv \sum_{i=1}^{11} c_i \pmod{27}$. Note such a set of values

$c_1, c_2, \ldots, c_{11}$ exists by Lemma 25. Observe that $C \leq 11 \times 21 = 231$. Consider $m = \frac{N-C}{27}$. For $N > 17024$ we have $m = \frac{N-C}{27} \geq \frac{N-231}{27} \geq \frac{17025-231}{27} \geq 622$ and $m = \frac{N-C}{27} < N$. So we can write $m = \sum_{i=1}^{11} m_i$ for non-zero integers $m_1, m_2, \ldots, m_{11} \in S_{3,=}$ by the inductive hypothesis. Then by Observation 24, for each $1 \leq i \leq 11$ we have $27m_i + c_i$ is a non-zero element of $S_{3,=}$. Thus we can write

$$\sum_{i=1}^{11} 27m_i + c_i = 27\sum_{i=1}^{11} m_i + \sum_{i=1}^{11} c_i = 27m + C = N.$$

Therefore, by induction we have that every integer $N \geq 622$ can be written as the sum of 11 non-zero elements of $S_{3,=}$. ☐

We can now prove Theorem 23.

Proof of Theorem 23. By Lemma 26 we have that every integer greater than or equal to 622 can be written as the sum of at most 11 elements of $S_{3,=}$. Performing a brute-force search on all natural numbers $N < 622$ yields a representation for N as the sum of at most 11 elements of $S_{3,=}$, except where N is one of the values listed in the statement of the theorem. ☐

We do not currently know if the bound 11 is optimal. By Theorem 22 there is a lower bound of 10.

2.5.8 Generalization to k-Context-Free Sets

Another natural generalization to consider is the generalization of the method from k-automatic sets to k-context-free sets. Just as k-automatic sets can be defined in terms of regular languages, k-context-free sets can be defined in terms of context-free languages. We can re-state the definition of a k-automatic sequence as a sequence $(a_n)_{n\geq 0}$ over a finite alphabet Δ for which for each symbol $c \in \Delta$, the set $\{(n)_k : a_n = c\}$ is a regular language. Then, we say a sequence $(b_n)_{n\geq}$ over finite alphabet Δ is k-context-free if for every $c \in \Delta$, the set $\{(n)_k : b_n = c\}$ is a context-free language. And as in the case for k-automatic sets, a set is k-context-free if its associated characteristic function defines a k-context-free sequence.

However, for k-context-free sets the criteria for forming (asymptotic) additive bases do not carry over from the result for k-automatic sets. That is, there are k-context-free sets that are not sparse and have elements with a greatest common divisor of 1, but do not form an asymptotic additive basis of finite order. An example of such a set, given in [4, Example 7.1], is $S = \{n : (n)_2 = 10^n x,\ \text{where } |x| = n \geq 0\} \cup \{0\}$. This set is 2-context-free and has density $\pi_S(x) = \Theta(x^{1/2})$, but it can be shown that at least $n/2$ terms are needed in order to represent $2^n(2^n - 1)$ as a summation of elements of S [4].

This means that the method cannot directly be extended to k-context-free sets. Deeper investigations of other questions concerning k-context-free sets and k-context-free sequences can be found, for example, in [19, 28].

2.6 N-State Automata Accepting Additive Basis Languages

We can also investigate questions about how common or rare it is for an N-state automaton to accept a language over Σ_2 that corresponds to the base-2 representations of a set that forms an (asymptotic) additive basis. As mentioned in Section 2.3.1 and as described by Bell, Hare, and Shallit in [4], an automatic set forms an asymptotic additive basis if and only if the set is not sparse and the greatest common divisor of the elements in the set is 1. Using an algorithm described in [18] we can efficiently determine if the language accepted by an N-state automaton is not sparse and `Walnut` can be used to determine if the greatest common divisor of the values corresponding to the accepted words is 1, following the approach described in [4]. Since the translation between the language of base-2 representations and the numbers represented requires that leading zeros have no effect on the membership of words in the language, we only consider automata that have a self-loop on the initial state on symbol 0. We refer to such automata as *consistent automata*. Table 2.3 displays the number of minimal consistent N-state automata over Σ_2 according to which of the criteria for forming an asymptotic additive basis the corresponding sets of values meet. Additionally, the table shows how many of these sets form an additive basis. Given a set forming an asymptotic additive basis it is easy to determine if it forms an additive basis; a necessary and sufficient condition for a set forming an additive basis that is known to form an asymptotic additive basis is that the set contains 1 as an element.

Table 2.3: Number of consistent N-state automata according to criteria met for forming an (asymptotic) additive basis

	Sparse		Non-Sparse			
N	GCD $\neq 1$	GCD $= 1$	GCD $\neq 1$	GCD $= 1$	GCD $= 1$ & Accepts $w = 1$	Total
1	1	0	0	1	1	2
2	1	0	1	6	4	8
3	0	6	25	187	103	218
4	18	58	666	8178	4422	8920
5	144	1232	24914	450802	237854	477092

Of those consistent N-state automata that correspond to sets forming (asymptotic) additive bases, we can use `Walnut` again to determine the orders of the (asymptotic) additive bases. This is done by writing a first-order logic query for asserting that all natural numbers, or all natural numbers greater than some value M, can be written as the sum of $k = 1, 2, 3, \ldots$ summands until the minimum order is found. For consistent automata with 2 or 3 states we can determine the additive basis order of each corresponding set that has a finite (asymptotic) additive basis order. However, even when the number of states is limited to $N = 4$, using `Walnut` to determine the minimum order of the additive basis is too computationally intensive. For example, the set corresponding to the automaton, M_4, depicted in Figure 2.16 does not form an asymptotic basis of order 6 and testing order 7 using `Walnut` requires a step in the computation involving the determinization of an NFA given by a nontrivial product of automata with 7 and 78,125 states respectively. Given the amount of computation required, it has not been determined whether or not this automaton corresponds to a set forming an asymptotic additive basis of order 7. That said, it may still be feasible to test as far as 7 summands, given that testing if the set forms an (asymptotic) additive basis of order 6 requires the determinization of an NFA with more than 66,000 states and `Walnut` is able to do this in less than 5 minutes of computation on a 2017 laptop computer.

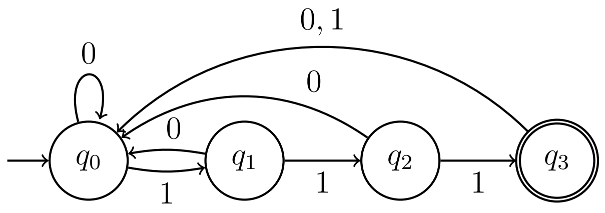

Figure 2.16: Consistent automaton, M_4, on 4 states with asymptotic additive basis order greater than 6

Nevertheless, we can use an alternative approach to prove that the set corresponding to the language accepted by M_4 forms an asymptotic additive basis of order 9. Let L_4 be the language accepted by M_4.

Theorem 27. *Every natural number greater than 160 can be written as the sum of at most 9 elements of $S_4 := [L_4]_2$. The 9 is optimal.*

Proof. Observe that the set S_4 is the set of numbers whose last block of 1's in their canonical base-2 representations is of length congruent to 3 (mod 4). So we have $7 \in S_4$ and $127 \in S_4$, as $(7)_2 = 111$ and $(127)_2 = 1111111$. Table 2.4 describes how to represent all natural numbers $n \geq 7 \cdot 7 + 127 = 176$ by considering each residue class for n modulo 16. In each case, the last term in the representation is congruent to 7 (mod 16).

Table 2.4: Representations of n as the sum of at most 9 elements of S_4

$n \pmod{16}$	Representation	Condition
0	7+7+7+7+7+7+127+$(n-(6\cdot 7+127))$	$n \geq 7\cdot 7+127$
1	7+7+7+7+7+7+$(n-(6\cdot 7))$	$n \geq 7\cdot 7$
2	7+7+7+7+127+$(n-(4\cdot 7+127))$	$n \geq 5\cdot 7+127$
3	7+7+7+7+$(n-(4\cdot 7))$	$n \geq 5\cdot 7$
4	7+7+127+$(n-(2\cdot 7+127))$	$n \geq 3\cdot 7+127$
5	7+7+$(n-(2\cdot 7))$	$n \geq 3\cdot 7$
6	127+$(n-127)$	$n \geq 7+127$
7	n	$n \geq 7$
8	7+7+7+7+7+7+7+$(n-(7\cdot 7))$	$n \geq 8\cdot 7$
9	7+7+7+7+7+127+$(n-(5\cdot 7+127))$	$n \geq 6\cdot 7+127$
10	7+7+7+7+7+$(n-(5\cdot 7))$	$n \geq 6\cdot 7$
11	7+7+7+127+$(n-(3\cdot 7+127))$	$n \geq 4\cdot 7+127$
12	7+7+7+$(n-(3\cdot 7))$	$n \geq 4\cdot 7$
13	7+127+$(n-(1\cdot 7+127))$	$n \geq 2\cdot 7+127$
14	7+$(n-(1\cdot 7))$	$n \geq 2\cdot 7$
15	7+7+7+7+7+7+7+7+$(n-(8\cdot 7))$	$n \geq 9\cdot 7$

Using a brute-force computation on the values of n less than 176 we can then write every natural number, except a set of 79 values less than or equal to 160, as the sum of at most 9 elements of S_4.

To prove that 9 summands is optimal, we can show that there are infinitely many n congruent to $15 \pmod{16}$ for which n cannot be written as the sum of 8 or fewer elements of S_4. Suppose $n \equiv 15 \pmod{32}$, giving $n \equiv 15 \pmod{16}$ as well. Observe that all elements of S_4 are congruent to either $7 \pmod{16}$ or $15 \pmod{16}$, but that those elements congruent to $15 \pmod{16}$ are necessarily congruent to $31 \pmod{32}$. For all choices of $k, m \in \mathbb{N}$ such that $k+m \leq 8$ we have $7k + 15m \equiv 15 \pmod{16}$ if and only if $k = 0$ and $m = 1$. So the only way to get a residue of $15 \pmod{16}$ with 8 or fewer summands from S_4 is with a single element congruent to $15 \pmod{16}$. But all such elements are congruent to $31 \pmod{32}$. Therefore we cannot write n as the sum of 8 or fewer elements of S_4. $\square$

Table 2.5 gives the number of minimal consistent N-state automata that correspond to sets that have each finite asymptotic additive basis order from 1 to 6. Computations were only able to be completed up to $N = 4$ due to the large running times and the significant memory resources required by `Walnut` to perform these computations. Similar data is given in Table 2.6, but for (non-asymptotic) additive basis orders of sets corresponding to consistent automata with 1, 2, or 3 states.

Table 2.5: Number of consistent N-state automata according to the asymptotic additive basis orders of the corresponding sets

	Asymptotic Additive Basis Order						
N	1	2	3	4	5	6	≥ 7
1	1	0	0	0	0	0	0
2	1	3	2	0	0	0	0
3	2	129	45	6	4	0	$1^{\dagger}$
4	12	6420	1326	206	146	37	31

† This set has asymptotic additive basis order 7.

Table 2.6: Number of consistent N-state automata according to the additive basis orders of the corresponding sets

	Additive Basis Order										
N	1	2	3	4	5	6	7	8	9	10	≥ 11
1	1	0	0	0	0	0	0	0	0	0	0
2	1	2	0	1	0	0	0	0	0	0	0
3	0	65	21	8	2	3	1	2	0	1	0

The code used to generate this data is publicly available at `https://github.com/FinnLidbetter/additiveNumberTheoryAutomata`.

2.7 Open Problems

We conclude this chapter with some open problems related to this work.

1. Does $S_{3,=}$ form an asymptotic additive basis of order 10?

   The largest value less than 8,000,000 that cannot be written as the sum of at most 10 elements of $S_{3,=}$ is 4,793,143. If it can be shown that there is a sufficiently large interval of values greater than or equal to 4,793,144 that can all be written as the sum of at most 10 elements of $S_{3,=}$, then the same method as used to prove Theorem 23 could work.

2. More generally, what is the value of $D(k)$, as in the statement of Theorem 22, for values of $k \geq 4$?

3. What other interesting sets can be shown to form an (asymptotic) additive base using the method of regular approximation?

4. Compute the number of N-state automata corresponding to sets forming (asymptotic) additive bases and the orders of these bases for larger N.

5. Does the ratio of N-state automata recognizing languages corresponding to sets that are (asymptotic) additive bases to all N-state automata approach some constant as N tends to infinity? If so, what is this constant?

6. Let $B(N)$ be the greatest value such that there is an N-state automaton accepting a language corresponding to a set forming an asymptotic additive basis of least finite order $B(N)$. What is the sequence $B(N)$? The first three terms are $1, 3, 7$.

7. Let $b(N)$ be the greatest value such that there is an N-state automaton accepting a language corresponding to a set forming an additive basis of least finite order $b(N)$. What is the sequence $b(N)$? The first 3 terms are 1, 4, 10.

   A lower bound for $b(N)$, that is tight for the first 3 terms, is given by $b(N) \geq 2^{N+1} - 2^{N-1} - 2$. This is because there is an N-state automaton recognizing a non-sparse language L such that $[L]_2$ has 1 and $2^{N+1} - 2^{N-1} - 1$ as its two smallest elements. Therefore, at least $m = 2^{N+1} - 2^{N-1} - 2$ summands (all of which are 1's) will be needed to represent m. The 3-state automaton giving $b(3) = 10$ is shown in Figure 2.17.

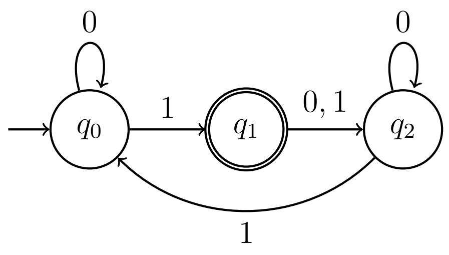

Figure 2.17: Consistent 3-State automaton corresponding to a set with additive basis order 10

Chapter 3

Counting Subwords

3.1 Preliminaries

3.1.1 Introduction

An idea used in many different topics within the study of formal languages is that of counting occurrences of letters within words. For example, the results of Chapter 2 are centered around languages defined by the relative counts of letters within words. The Parikh map (see, e.g., [31]), a tool in formal language theory with a variety of applications, is another example of this idea of counting occurrences of letters within words. In this chapter we consider a generalization of counting occurrences of letters within words to counting occurrences of words as subwords within words. Recall that for $a \in \Sigma$ and $w \in \Sigma^*$ we define $|w|_a$ to be the number of occurrences of letter a within word w. We extend this definition such that for words $w, x \in \Sigma^*$, the notation $|w|_x$ is defined to mean the number of (possibly overlapping) occurrences of word x within word w. For example, we have $|\texttt{banana}|_{\texttt{ana}} = 2$. In this chapter we study the languages

$$\begin{aligned} L_{x<y} &= \{w \in \Sigma^* : |w|_x < |w|_y\}, \\ L_{x\le y} &= \{w \in \Sigma^* : |w|_x \le |w|_y\}, \\ L_{x=y} &= \{w \in \Sigma^* : |w|_x = |w|_y\}, \end{aligned}$$

and their complements. In particular, we address the question of when these languages are regular.

The content of this chapter is based upon the published paper, Counting Subwords and Regular Languages [11], and the unpublished extended version of the same name [12].

Both of these are works for which the author of this thesis is a coauthor. As such, large sections of the content within this chapter are copied verbatim from [11, 12]. The order in which some of this content is presented here has been adapted from these papers to better suit this alternative presentation format. The content of the remainder of Section 3.1, all of Section 3.2, and Section 3.3 can be found in [11, 12].

3.1.2 Bordered Words and Periodicity

Let y, z be words with y nonempty. We say that z is *y-bordered* if $z \neq y$ and y is both a prefix and a suffix of z. There are two types of y-bordered words: one where the prefix and suffix y do not overlap in z (that is, where $|y| \leq |z|/2$), and one where they do (that is, where $|y| > |z|/2$). In the first case, we say that z is *disjoint* y-bordered, and in the second case, *overlapping* y-bordered. For example, `entanglement` is disjoint `ent`-bordered, and `alfalfa` is overlapping `alfa`-bordered. For more about borders of words, see, for example, [14].

We will need two lemmas about bordered words. To prove the first of these two lemmas and some other results in this chapter it will be useful to recall a theorem of Lyndon and Schützenberger as in, for example, [26] or [38, Theorem 2.3.2].

Theorem 28 ([26], as stated in [38])**.** *Let $x, y, z \in \Sigma^+$. Then $xy = yz$ if and only if there exist $u \in \Sigma^+, v \in \Sigma^*$, and an integer $e \geq 0$ such that $x = uv$, $z = vu$, and $y = (uv)^e u = u(vu)^e$.*

Lemma 29. *Suppose $z \in \Sigma^*$ is y-bordered. Then there exist words $u \in \Sigma^+$ and $v \in \Sigma^*$ and an integer $e \geq 0$ such that $y = (uv)^e u$ and $z = (uv)^{e+1} u$.*

Proof. Follows immediately from Theorem 28. □

Lemma 30. *Let $u \in \Sigma^+$, $v \in \Sigma^*$, and $e \geq 0$. Suppose that $y = (uv)^e u$. Define $z_1 = (uv)^{e+1}$ and $z_2 = (uv)^{e+2}$. Let $c = |z_1|_y$ and $d = |z_2|_y - |z_1|_y$. Then $c, d \geq 1$ and $|(uv)^i|_y = (i - e)d + c - d$ for all integers $i > e$.*

Proof. If $x = (uv)^{e+1}$, then $|x|_y = c$. Appending uv to x on the right results in $d \geq 1$ additional copies of y. The result now follows by induction. □

We also recall the following classical result.

Theorem 31. *Let x, y be nonempty words. There exists a word with two distinct factorizations as a concatenation of x's and y's if and only if $xy = yx$.*

Proof. This follows from the so-called "defect theorem" [25], or from [17, Theorem 1]. □

3.1.3 Pattern-Matching Automata

We will need the following well-known result about pattern-matching automata (for example, see [13, §32.3]).

Theorem 32. *Given a word $w \in \Sigma^n$, a DFA $M = (\{q_0, \ldots, q_n\}, \Sigma, \delta, q_0, \{q_n\})$ exists of $n + 1$ states such that $\delta(q_0, x) = q_n$ if and only if w is a subword (resp., suffix) of x. Here the state q_i can be interpreted as asserting that the longest suffix of the input that matches a prefix of w is of length i.*

3.1.4 Interlacing

Suppose y is a subword of every x-bordered word. In this case we say x is *interlaced by* y. For example, it is easy to check that 000100 is interlaced by 1000 when the underlying alphabet Σ is $\{0, 1\}$. The following lemma gives the fundamental property of interlacing:

Lemma 33. *Suppose x is interlaced by y, and suppose z is a word satisfying $|z|_y = |z|_x + k$. Then for all t we have $|zt|_y \geq |zt|_x + k - 1$. In particular, $|t|_y \geq |t|_x - 1$ for all t.*

Proof. Identify the starting positions of all occurrences of x in zt. Since x is interlaced by y, between any two consecutive occurrences of x, there must be at least one occurrence of y. So if zt has i more occurrences of x than z does, then zt must have at least $i - 1$ more occurrences of y than z does. □

3.2 Regularity of Subword Count Comparison Languages

3.2.1 Deterministic Context-Freeness

We now return our attention to the languages $L_{x<y}, L_{x\leq y}, L_{x=y}$ and their complements. It is easy to prove a first result on the hierarchical location of such languages. For a precise definition of the class of deterministic context-free languages see, for example, [38, Section 4.7] or [40, Section 2.4].

Proposition 34. *For all words x and y, the languages $L_{x<y}$, $L_{x\leq y}$, $L_{x=y}$, and their complements $L_{x\geq y}$, $L_{x>y}$, $L_{x\neq y}$ are all deterministic context-free languages.*

Proof. We prove this only for $L_{x=y}$, with the other cases being analogous. We construct a deterministic pushdown automaton M that recognizes $L_{x=y}$ as follows: its states record the last $\max(|x|,|y|)-1$ letters of the input seen so far. The stack of M is used as a counter to maintain the absolute value of the difference between the number of x's seen so far and the number of y's (a flag in the state records the sign of the difference). We have M accept its input if and only if this difference is 0. Since there is only one possible action for every triple of state, input symbol, and top-of-stack symbol, M is deterministic (any "invalid" configurations transition to a dead state d). $\square$

We proceed by working towards necessary and sufficient conditions on when $L_{x<y}$, $L_{x\leq y}$, $L_{x=y}$, and their complements are regular languages.

3.2.2 Regularity

While $L_{x=y}$ is always deterministic context-free, sometimes—perhaps surprisingly—it can also be regular. For example, when the underlying alphabet Σ is unary, then $L_{x=y}$ is always regular. Less trivially, for $\Sigma = \{0,1\}$ it is an easy exercise to show that $L_{01=10}$ is regular, and is recognized by the 5-state DFA in Figure 3.1; however, $L_{01=10}$ is not regular when $\Sigma = \{0,1,2\}$. We can see this by observing that the morphism $h_1 : \{a,b\} \to \Sigma^*$ defined by $h_1(a) = 2012$ and $h_1(b) = 2102$ gives $h_1^{-1}(L_{01=10} = \{x \in \{a,b\}^* : |x|_a = |x|_b\}$, a context-free language. Then, by the closure of regular languages under inverse morphism we can deduce that $L_{01=10}$ is not regular over the alphabet $\Sigma = \{0,1,2\}$. On the other hand, $L_{0011=1100}$ is never regular, even when $\Sigma = \{0,1\}$. Using the morphism $h_2 : \{a,b\} \to \Sigma^*$, defined by $h_2(a) = 001101$ and $h_2(b) = (110010)$, we again get $h_2^{-1}(L_{0011=1100}) = \{x \in \{a,b\}^* : |x|_a = |x|_b\}$.

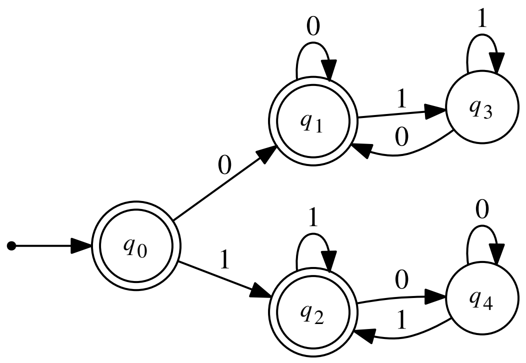

Figure 3.1: A DFA recognizing $L_{01=10}$ over $\Sigma = \{0, 1\}$

We give necessary and sufficient conditions on when $L_{x<y}$ and $L_{x=y}$ are regular in Theorem 35 and Theorem 36 respectively. Note that the argument for the remaining case of $L_{x \le y}$ is very similar to the arguments for $L_{x<y}$ and $L_{x=y}$. Conditions on when the complements of these languages are regular are identical, due to the closure of regular languages under complement.

Theorem 35. *The language $L_{x<y}$ is regular if and only if either x is interlaced by y or y is interlaced by x.*

Proof. $\Longleftarrow$: There are two cases: (i) x is interlaced by y; and (ii) y is interlaced by x.

Case (i): Using Lemma 33, we can build a finite automaton M recognizing $L_{x<y}$ as follows: using the pattern-matching automata for x and y described in Section 3.1.3, on input z the machine M records whether

(a) $|z|_x = |z|_y + 1$;

(b) $|z|_x = |z|_y$;

(c) $|z|_x = |z|_y - 1$, and $|z'|_x \ge |z'|_y - 1$ for all prefixes z' of z;

(d) $|z'|_x \le |z'|_y - 2$ for some prefix z' of z.

Of course we do not maintain the actual numbers $|z|_x$ and $|z|_y$ in M, but only which of (a)–(d) hold. Lemma 33 implies that the four cases above cover all the possibilities. It is not possible to have $|z|_x \geq |z|_y + 2$, and if (d) ever occurs, we know from Lemma 33 that $|z|_x < |z|_y$ for all words z extending z'. So in this case the correct action is for the automaton to remain in state (d), an accepting state that loops to itself on all inputs. The automaton accepts the input if and only if it is in the states corresponding to conditions (c) and (d).

Case (ii): Using Lemma 33, as in Case (i), we can build a finite automaton recognizing $L_{x<y}$ as follows: using the pattern-matching automata for x and y described in Section 3.1.3, on input z the machine M records whether

(a) $|z|_y = |z|_x + 1$;

(b) $|z|_y = |z|_x$, and $|z'|_y \geq |z'|_x$ for all prefixes z' of z;

(c) $|z'|_y \leq |z'|_x - 1$ for some prefix z' of z.

Lemma 33 implies that the three cases above cover all the possibilities. It is not possible to have $|z|_y \geq |z|_x + 2$, and if (c) ever occurs, we know from Lemma 33 that $|z|_x \geq |z|_y$ for all words z extending z'. So in this case the correct action is for the automaton to remain in state (c), a rejecting "dead" state that loops to itself on all inputs. The automaton accepts the input if and only if it is in the state corresponding to condition (a).

$\Longrightarrow$: We proceed by proving the contrapositive. So suppose that there is some y-bordered word r such that x is not a subword of r, and some x-bordered word s such that y is not a subword of s. Using Lemma 29, we know that there are words u, v, p, q and natural numbers e, f such that $r = (uv)^{e+1}u$, and $y = (uv)^e u$, and $s = (pq)^{f+1}p$, and $x = (pq)^f p$.

Suppose that x is a subword of $(uv)^i u$ for some $i \geq 0$. Since x is not a subword of r, we know that $i \geq e+2$. If x is a subword of $(uv)^{e+2}u$ and not a subword of $(uv)^{e+1}u$, then $y = (uv)^e u$ must be a subword of x. But then y is a subword of s, a contradiction. So x is not a subword of $(uv)^i u$ for any i. By exactly the same reasoning we deduce that y is not a subword of $(pq)^j p$ for any $j \geq 0$.

Let $c = |(uv)^{e+1}|_y$ and $d = |(uv)^{e+2}|_y - |(uv)^{e+1}|_y$. Similarly, define $c' = |(pq)^{f+1}|_x$ and $d' = |(pq)^{f+2}|_x - |(pq)^{f+1}|_x$. Consider a word $z = (uv)^i (pq)^j$, where $i > e$ and $j > f$. From above and Lemma 30, we know that $|(uv)^i|_x = 0$ for all $i \geq 0$ and $|(pq)^j|_x = (j-f)d' + c' - d'$ for $j > f$. Let m be the number of additional occurrences of x that straddle the boundary between $(uv)^{e+1}$ and $(pq)^{f+1}$. That is, m is the number of distinct values for k, such that x is a subword of $(uv)^{e+1}(pq)^{f+1}$ starting at index k and $(e+1)|uv|+2-|x| \le k \le (e+1)|uv|+1$. Similarly, we know that $|(uv)^i|_y = (i-e)d+c-d$ for $i > e$ and $|(pq)^j|_y = 0$ for all $j \ge 0$. Let n be the number of additional occurrences of y that straddle the boundary between $(uv)^{e+1}$ and $(pq)^{f+1}$. The precise definition of n is given as above by replacing m and x with n and y respectively. Thus z has $(j-f)d' + c' - d' + m$ occurrences of x and $(i-e)d + c - d + n$ occurrences of y.

Now assume, contrary to what we want to prove, that $L_{x<y}$ is regular. Define $L = L_{x<y} \cap (uv)^e(uv)^+(pq)^f(pq)^+$. Then L is regular. Define a morphism $h : \{a,b\}^* \to \Sigma^*$ as follows: $h(a) = uv$, and $h(b) = pq$. We claim that $h^{-1}(z) = \{a^ib^j\}$. One direction is clear. For the other, suppose $h^{-1}(z)$ included some word other than a^ib^j. Then by Theorem 31, we know that uv and pq commute. But then by the Lyndon-Schützenberger theorem [26], uv and pq are both powers of some word t. But then x would be a subword of $(uv)^\ell u$ for some ℓ, which we already saw to be impossible.

By a well-known theorem (e.g., [38, Theorem 3.3.9]), $h^{-1}(L)$ is regular. But $h^{-1}(L) = \{a^ib^j \ : \ (i-e)d + c - d + n < (j-f)d' + c' - d' + m, \text{ for } i > e, \ j > f\}$ which, using the pumping lemma, is not regular. ☐

Theorem 36. *The language $L_{x=y}$ is regular if and only if either x is interlaced by y or y is interlaced by x.*

Proof. The proof is quite similar to the case $L_{x<y}$, and we indicate only what needs to be changed.

$\Longleftarrow$: Without loss of generality we can assume that x is interlaced by y. Using Lemma 33 we can build a finite automaton recognizing $L_{x=y}$ just as we did for $L_{x<y}$, using case (i). The only difference now is that the accepting state corresponds to (b).

$\Longrightarrow$: Proceeding by contraposition, suppose that there is some y-bordered word r such that x is not a subword of r, and some x-bordered word s such that y is not a subword of s. Once again, we follow the argument used for $L_{x<y}$, but there is one difference.

Recall that $z = (uv)^i(pq)^j$ for some $i > e$ and $j > f$. By the argument for $L_{x<y}$ we know that z has $(j-f)d' + c' - d' + m$ occurrences of x and $(i-e)d + c - d + n$ occurrences of y. Let $A = (-(m+c')) \bmod d'$ and $B = (-(n+c)) \bmod d$. Let w be the shortest suffix of $(uv)^{e+2}$ such that wz has $(i-e)d + c - d + n + B$ occurrences of y; let w' be the shortest prefix of $(pq)^{f+2}$ such that zw' has $(j-f)d' + c' - d' + m + A$ occurrences of x. Then by our construction wzw' has $(j-f+C)d'$ occurrences of x and $(i-e+D)d$ occurrences of y, for some $C, D \ge 0$.

Now assume, contrary to what we want to prove,that $L_{x=y}$ is regular. Define $L' = L_{x=y} \cap w(uv)^e(uv)^+(pq)^f(pq)^+w'$. Then L' is regular. Define $L = \#L'\#$, where # is a new symbol not in the alphabet Σ; then L is regular. Define a morphism $h : \{a, b, a', b'\}^* \to \Sigma^*$ as follows: $h(a') = \#w$, $h(a) = uv$, $h(b) = pq$, and $h(b') = w'\#$. We claim that $h^{-1}(\#wzw'\#) = \{a'a^ib^jb'\}$. One direction is clear, and the other follows from Theorem 31. By a well-known theorem (e.g.,[38, Theorem 3.3.9]), $h^{-1}(L)$ is regular. But $h^{-1}(L) = \{a'a^ib^jb' \ : \ (i - e + D)d = (j - f + C)d', \text{ for } i > e, \ j > f\}$ which, using the pumping lemma, is not regular. □

3.2.3 Testing the Criteria

Given x, y we can test if there is some y-bordered word z such that x is not subword of z, as follows: create a DFA recognizing the language

$$(\Sigma^* x \Sigma^*)^c \ \cap \ y\Sigma^+ \ \cap \ \Sigma^+ y.$$

A simple construction gives such a DFA M with at most $N = (|x| + 1)(|y| + 3)(|y| + 2)$ states and at most $N|\Sigma|$ transitions.

This can be improved to $N' = (|x| + 1)(2|y| + 3)$ states as follows: first build a DFA of $(2|y| + 3)$ states recognizing the language $y\Sigma^+ \ \cap \ \Sigma^+ y$ by "grafting" the DFA, A_1, of $|y| + 3$ states recognizing $y\Sigma^+$ onto the DFA, A_2, of $|y| + 2$ states recognizing $\Sigma^+ y$. This can be done by modifying the pattern-matching DFA described in Theorem 32. Simply replace transitions to the final state in A_1 with transitions to the appropriate states in A_2. The final state of A_1 and the initial state of A_2 both become unreachable. Then form the direct product with the DFA for $(\Sigma^* x \Sigma^*)^c$. The resulting DFA has N' states. We can then use a depth-first search on the underlying transition graph of M to check if $L(M) \neq \emptyset$.

Thus, we have proved:

Corollary 37. *There is an algorithm running in time $O(|\Sigma||x||y|)$ that decides whether the criteria of Theorems 35 and 36 hold.*

Corollary 38. *If there exists a y-bordered word z such that x is not a subword of z, then $|z| < N'$.*

Proof. If $M = (Q, \Sigma, \delta, q_0, F)$ accepts any word at all, then it accepts a word of length at most $|Q| - 1$. □

3.2.4 Improving the Bound in Corollary 38

As we have seen in Corollary 38, if x is not a subword of some y-bordered word, then there is a relatively short "witness" to this fact. We now show that this witness can be taken to be of the form yty for some t of *constant length*. The precise constant depends on the cardinality of the underlying alphabet Σ. In Corollary 40 we prove that if $|\Sigma| \geq 3$, then this constant is 1. In Corollary 45 we prove that if $|\Sigma| = 2$, then this constant is 3.

Theorem 39. *Suppose Σ is an alphabet that contains at least three symbols, and let $x, y \in \Sigma^*$. Without loss of generality assume that $\{0, 1, 2\} \subseteq \Sigma$. If x is a subword of $y0y$ and $y1y$ and $y2y$, then x is a subword of y.*

Proof. Assume, contrary to what we want to prove, that x is not a subword of y. Also assume that $|y| = m$ and $|x| = n$. For x to be a subword of $y0y$ (resp., $y1y$, $y2y$), then, it must be that x "straddles" the y—y boundary. More precisely, when we consider where x appears inside $y0y$, the first symbol of x must occur at or to the left of position $m + 1$ of $y0y$ (resp., $y1y$, $y2y$). Similarly, the last symbol of x must occur at or to the right of position $m + 1$ of $y0y$ (resp., $y1y$, $y2y$).

For $a = 0, 1, 2$, label the x that matches yay as x_a, and assume that the position of the 0 that matches x_0 is i, the position of the 1 that matches x_1 is j, and the position of the 2 that matches x_2 is k. Note that $x_0 = x_1 = x_2 = x$; the indices just allow us to refer to the diagram below. Without loss of generality we can assume $1 \leq i < j < k \leq n$. Thus we obtain a picture as in Figure 3.2. Here we have labeled the two occurrences of y as y and y', so we can refer to them unambiguously. Note that $i \geq 1$ and $k \leq m + 1$. Furthermore, note that $n \leq m + i$.

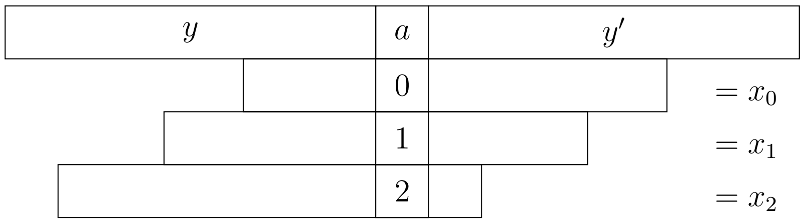

Figure 3.2: Matches of x against $y0y$, $y1y$, and $y2y$

We now use "index-chasing" to show that $x[k] = x[i]$; this will give us a contradiction, since $x[k] = 2$ and $x[i] = 0$. We will use the following identities, which can be deduced by observing Figure 3.2.

$$\begin{align} x_1[\ell] &= y[\ell+m+1-j] \text{ for } 1 \le \ell \le j-1; \tag{3.1}\\ x_2[\ell] &= y[\ell+m+1-k] \text{ for } 1 \le \ell \le k-1; \tag{3.2}\\ x_0[\ell] &= y'[\ell-i] \text{ for } i+1 \le \ell \le n; \tag{3.3}\\ x_1[\ell] &= y'[\ell-j] \text{ for } j+1 \le \ell \le n. \tag{3.4}\end{align}$$

Notice that $j+1 \le k \le n$, so we can take $\ell = k$ in (3.4) to get $x[k] = y[k-j]$. Additionally, $k-j \ge 1$, giving $i+1 \le i+k-j$. Also $i-j < 0 \le n-k$, so $i+k-j \le n$. Thus we can take $\ell = i+k-j$ in (3.3) to obtain $y[k-j] = x[i+k-j]$.

Since $i \ge 1$ and $k-j \ge 1$, we get $i+k-j \ge 2$. Since $j-i \ge 1$ we have $i-j \le -1$ and $i+k-j \le k-1$. Thus we can take $\ell = i+k-j$ in (3.2) to get $x[i+k-j] = y[i+m+1-j]$. Since $1 \le i \le j-1$, we can take $\ell = i$ in (3.1) to get $y[i+m+1-j] = x[i]$. Putting these observations together, we finally obtain

$$2 = x[k] = y[k-j] = x[i+k-j] = y[i+m+1-j] = x[i] = 0,$$

which produces the desired contradiction. □

Corollary 40. *Suppose $|\Sigma| \ge 3$. Then x is a subword of yty for all t with $|t| = 1$ if and only if y is interlaced by x.*

We now turn to case of a binary alphabet. This case is more subtle. For example, consider when $x = 10100$ and $y = 01001010$. Then, as can be verified, x is a subword of the self-overlaps $y(010)^{-1}y$ and $y0^{-1}y$, as well as the words yy, $y0y$, $y1y$, $y00y$, $y01y$, $y10y$, $y11y$, $y000y$, $y001y$, $y010y$, $y011y$, $y100y$, $y101y$. But x is not a subword of $y110y$.

For a binary alphabet Σ, a special role is played by the language

$$A = 01^+ \cup 10^+ \cup 0^+1 \cup 1^+0.$$

We also define the following languages. For each integer $k \ge 1$, let

$$B_{0^k1} := (1 + 01 + \cdots + 0^{k-1}1)^+0^k0^* \text{ and } B_{10^k} := 0^*0^k(1 + 10 + \cdots + 10^{k-1})^+.$$

Similarly, define B_{1^k0} and B_{01^k} by relabeling 0 to 1 and 1 to 0.

Lemma 41. *Suppose $\Sigma = \{0,1\}$ and $x \in A$. Then $y \in B_x$ if and only if x is not a subword of y, but x is a subword of all y-bordered words.*

Proof. We consider the case where $x = 0^k1$ and note that the case where $x = 1^k0$ is given by relabeling 0 to 1 and 1 to 0, and the other two cases are given by a symmetric argument.

$\Longrightarrow$: Suppose that $y \in B_x = B_{0^k1} = (1 + 01 + \cdots + 0^{k-1}1)^+ 0^k 0^*$ and z is a y-bordered word. By the definition of B_x, observe that x is not a subword of y. By Lemma 29 there exist $u \in \Sigma^+$, and $v \in \Sigma^*$, and a natural number $e \geq 0$ such that $y = (uv)^e u$ and $z = (uv)^{e+1}u$.

We first show that $e \leq 1$. If we assume the contrary, then $y = (uv)^{e-2}uvuvu$. We know that vu has the suffix 10^{k+i} for some $i \geq 0$. But since there is at least one 1 in vu we have that 0^k1 is a subword of $vuvu$, giving a contradiction.

If $e = 0$ then $z = uvu = yvy$ for some $v \in \Sigma^*$. Since vy has at least one 1 and y has a suffix of 0^k, we get that x is a subword of $yvy = z$. If $e = 1$ then $y = uvu$ such that vu has the suffix 0^k and there is at least one 1 in vu. Then $z = (uv)^2u = uvuvu$ has 0^k1 as a subword.

$\Longleftarrow$: Assume, to get a contradiction, that there is some $y \in \Sigma^* \setminus B_x = \Sigma^* \setminus B_{0^k1}$ such that x is a subword of all y-bordered words and x is not a subword of y. Then y satisfies at least one of the following cases, and we will get a contradiction in each of these.

Case (i): $y = 0^i$ for some $i \geq 0$. Clearly, x is not a subword of the y-bordered word yy.

Case (ii): y has $x = 0^k1$ as a subword, giving an immediate contradiction.

Case (iii): The suffix of y is 10^i for some $0 \leq i < k$. Then consider the y-bordered word $z = y1y$. If x is a subword of z but x is not a subword of y, then x must straddle the y—y boundary in z. So the 1 in $x = 0^k1$ must align with the 1 between the y's in $z = y1y$. But the suffix of y is 10^i for $i < k$. So x cannot be a subword of $y1y$. □

However, for $x \notin A$, it turns out that if x is not a subword of y, then there is some word t of length 3 such that x is not a subword of yty. To prove this we first give two preliminary lemmas.

Lemma 42. *Suppose $\Sigma = \{0, 1\}$, and let $x, y \in \Sigma^*$ with $|x| = n$ and $|y| = m$. Suppose x is not a subword of y, but x is a subword of yty for all $t \in \Sigma^*$ such that $|t| = 3$ and $x \notin A$. Then for every integer k satisfying $\max\{1, m - n + 2\} \leq k \leq \min\{2m + 3 - n, m + 2\}$ and for all pairs of words $t_1, t_2 \in \Sigma^*$ with $|t_1| = |t_2| = 3$, we have either $x \neq (yt_1y)[k..k + n - 1]$ or $x \neq (yt_2y)[k + 1..k + n]$, or both.*

Proof. Assume, to get a contradiction, that there exist $x, y \in \Sigma^*$ such that x is not a subword of y and $x \notin A$ and that there exist $t_1, t_2 \in \Sigma^*$ with $|t_1| = |t_2| = 3$ and an integer

k satisfying $\max\{1, m-n+2\} \leq k \leq \min\{2m+3-n, m+2\}$ such that $(yt_1y)[k..k+n-1] = (yt_2y)[k+1..k+n] = x$, and furthermore x is a subword of yty for all $t \in \Sigma^*$ with $|t| = 3$. Let $t_1 = a_1b_1c_1$ and $t_2 = a_2b_2c_2$ and $x = x_1x_2 \cdots x_n$. Before proceeding, first observe that $n \geq 3$ since for all x with $|x| \leq 2$ we have that either $x \in A$ or x is not a subword of one of $y000y$ and $y111y$.

Case (i): $\max\{1, m-n+3\} \leq k \leq \min\{2m+3-n, m+1\}$.
If $\max\{1, m-n+4\} \leq k \leq \min\{2m+3-n, m\}$ then $n \geq 4$ and we can write $x = x_1va_1b_1c_1w = va_2b_2c_2wx_n$ for some $v, w \in \Sigma^*$ where $x_1v = va_2$ and $c_1w = wx_n$ and $a_1b_1 = b_2c_2$. Then, by the first theorem of Lyndon-Schützenberger, we have that $v = x_1^i$ and $w = x_n^j$ for integers $i, j \geq 0$. Thus x can be re-written as $x = x_1^i a_1 b_1 x_n^j$ for $x_1, a_1, b_1, x_n \in \Sigma$ and $i, j \geq 1$.

If $k = m-n+3$ then, where $n \geq 3$, we can write $x = x_1va_1b_1 = va_2b_2c_2$ for $v \in \Sigma^*$ and after applying the first theorem of Lyndon-Schützenberger we get $x = x_1^i a_1 b_1$ where $i \geq 1$.

Similarly if $k = m+1$ then we can write $x = a_1b_1c_1w = b_2c_2wx_n$ and applying the first theorem of Lyndon-Schützenberger gives $x = a_1b_1x_n^j$ for $j \geq 1$.

So we have that $x = x_1^i a_1 b_1 x_n^j$ for $a_1, b_1, x_1, x_n \in \Sigma$ and $i, j \geq 0$ and either $i \geq 1$ or $j \geq 1$. We will proceed by getting a contradiction for each possible assignment of a_1, b_1, x_1, x_n to symbols in Σ for all valid i, j. Table 3.1 gives contradictions for all possible assignments where $x_1 = 0$. Note that the remaining cases can be ruled out by relabeling 0 to 1 and 1 to 0.

Case (iii): $k = m-n+2 \geq 1$.
We can write $x = x_1wa_1 = wa_2b_2$ for some $w \in \Sigma^+$, where $x_1w = wa_2$. So by the first theorem of Lyndon-Schützenberger, we get $w = x_1^i$ for some integer $i \geq 1$ and thus $x = x_1^{i+1}x_n$ for $x_1, x_n \in \Sigma$. If $x_1 = x_n$, then $x_1^{i+1}x_n = x_1^{i+2}$. But, since x is not a subword of y we cannot have that x_1^{i+2} is a subword of both $y111y$ and $y000y$, giving a contradiction. If instead we have $x_1 \neq x_n$, then $x_1^{i+1}x_n \in A$, an immediate contradiction.

Case (iv): $k = m+2 \leq 2m+3-n$.
We can write $x = b_1c_1w = c_2wx_n$ and similar to Case (iii), we get $x = x_1x_n^{i+1}$ for $x_1, x_n \in \Sigma$ and $i \geq 1$. By the same argument as in Case (iii), we get a contradiction if $x_1 = x_n$ and if $x_1 \neq x_n$.

□

Lemma 43. *Suppose $\Sigma = \{0,1\}$, and let $x, y \in \Sigma^*$ with $|x| = n$ and $|y| = m$. If x is a subword of yty for all $t \in \Sigma^*$ such that $|t| = 3$ and $x \notin A$, and x is not a subword of y, then*

Table 3.1: Contradictions for each $a_1, b_1, x_n \in \Sigma$ and $x_1 = 0$, where in each row $x = x_1^i a_1 b_1 x_n^j$ and $i, j \geq 0$ and either $i \geq 1$ or $j \geq 1$. The contradictions rely on the assumption that x is not a subword of y.

x_1	a_1	b_1	x_n	Contradiction
0	0	0	0	For all $i, j \geq 0$ we have that x is not a subword of $y111y$.
0	0	0	1	For all $i \geq 0$: If $j = 0$, then x is not a subword of $y111y$; If $j = 1$, then $x = 0^{i+2}1 \in A$; If $j > 1$, then if x is a subword of $y101y$, then y has $0^{i+2}1^{j-1}$ as a suffix. But if x is a subword of $y011y$, then y has 0^{i+1} as a suffix.
0	0	1	0	For all $i \geq 0$: If $j = 0$, then $x = 0^{i+1}1 \in A$; If $j > 0$, then x is not a subword of $y111y$.
0	0	1	1	If $i = 0$ or $j = 0$, then $x \in A$. If $i > 0$ and $j > 0$, then if x is a subword of $y101y$, then y has $0^{i+1}1^j$ as a suffix. But if x is a subword of $y011y$, then y has 0^i as a suffix.
0	1	0	0	If $i = 0$, then $x = 10^{j+1} \in A$. If $i > 0$, then x is not a subword of $y111y$.
0	1	0	1	If $i = 0$ and $j > 0$, then x is not a subword of $y000y$. If $i > 0$ and $j = 0$, then x is not a subword of $y111y$. If $i > 0$ and $j = 1$, then if x is a subword of $y011y$, then y has 0^i1 as a suffix. But if x is a subword of $y111y$, then y has 0^i10 as a suffix. If $i = 1$ and $j > 0$, then if x is a subword of $y001y$, then y has 01^j as a prefix. But if x is a subword of $y000y$, then y has 101^j as a prefix. If $i > 1$ and $j > 1$, then if x is a subword of $y011y$, then y has 0^i1 as a suffix. But if x is a subword of $y111y$, then y has 0^i101^ℓ as a suffix for some $\ell < j$. Since $i > 1$, this is a contradiction.
0	1	1	0	If $i = 0$ and $j = 1$, then $x = 110 \in A$. If $i = 1$ and $j = 0$, then $x = 011 \in A$. If $i > 1$ and $j = 0$, then if x is a subword of $y101y$, then y has 0^i1 as a suffix. But if x is a subword of $y011y$, then y has 0^{i-1} as a suffix. If i=0 and $j > 1$, then if x is a subword of $y101y$, then y has 10^j as a prefix. But if x is a subword of $y110y$, then y has 0^{j-1} as a prefix. If $i > 0$ and $j > 0$, then x is not a subword of $y111y$.
0	1	1	1	For all $j \geq 0$: If $i = 0$, then x is not a subword of $y000y$; If $i = 1$, then $x = 01^{j+2} \in A$; If $i > 1$, then if x is a subword of $y011y$, then y has 0^{i-1} as a suffix. But if x is a subword of $y101y$, then y has 0^i1^{j+1} as a suffix.

Table 3.2: Contradictions for each valid $u \in \Sigma^+$, $v \in \Sigma^*$, and $x_1 = 0$, where in each row $x = x_1(uv)^{i+1}u$ for $i \geq 0$. The contradictions rely on the assumption that x is not a subword of y.

x_1	u	v	Contradiction
0	00	ϵ	x is not a subword of $y111y$.
0	0	0	x is not a subword of $y111y$.
0	01	ϵ	If x is a subword of $y111y$, then y has $0(01)^{i+1}0$ as a suffix. But if x is a subword of $y011y$, then y has $0(01)^{i+1}$ as a suffix.
0	0	1	x is not a subword of $y111y$.
0	10	ϵ	x is not a subword of $y111y$.
0	1	0	If x is a subword of $y111y$ then y has $0(10)^{i+1}$ as a suffix. But if x is a subword of $y011y$ then y has $(01)^{i+1}$ as a suffix.
0	11	ϵ	$x \in A$.
0	1	1	$x \in A$.

for all pairs of words $t_1, t_2 \in \Sigma^*$ with $|t_1| = |t_2| = 3$ we have either $x \neq (yt_1y)[m+1..m+n]$, or $x \neq (yt_2y)[m+3..m+2+n]$, or both.

Proof. Assume, to get a contradiction, that there exist $x, y \in \Sigma^*$ such that x is not a subword of y and $x \notin A$ and that there exist $t_1, t_2 \in \Sigma^*$ with $|t_1| = |t_2| = 3$ such that $(yt_1y)[m+1..m+n] = (yt_2y)[m+3..m+2+n] = x$, and furthermore x is a subword of yty for all $t \in \Sigma^*$ with $|t| = 3$. Let $t_1 = a_1b_1c_1$ and $t_2 = a_2b_2c_2$ and $x = x_1x_2 \cdots x_n$, and assume $|y| = m$.

We can write $x = a_1b_1c_1w = c_2wx_{n-1}x_n$. So $b_1c_1w = wx_{n-1}x_n$ and by the first theorem of Lyndon-Schützenberger there exist $u \in \Sigma^+$ and $v \in \Sigma^*$ and an integer $i \geq 0$ such that $b_1c_1 = uv$ and $x_{n-1}x_n = vu$ and $w = (uv)^iu = u(vu)^i$. This gives $x = x_1wx_{n-1}x_n = x_1(uv)^iuvu = x_1(uv)^{i+1}u$. We now consider each possible $u \in \Sigma^+$ and $v \in \Sigma^*$, seeking a contradiction in each case. The contradictions are summarized in Table 3.2. Note again that the contradictions are given for all cases where $x_1 = 0$; the remaining cases can be obtained by relabeling 0 to 1 and 1 to 0.

□

Theorem 44. *Suppose $\Sigma = \{0, 1\}$ and let $x, y \in \Sigma^*$. If x is a subword of yty for all $t \in \Sigma^*$ such that $|t| = 3$ and $x \notin A$, then x is a subword of y.*

Proof. Define the function $f : \Sigma^* \times \Sigma^* \to \mathbb{N}$ over pairs of words $x, w \in \Sigma^*$ such that x is a subword of w as $f(x, w) = \min\{i \in \mathbb{N} : w[i..i + |x| - 1] = x\}$. Also, define the bitwise complements of elements of Σ as $\overline{0} = 1$ and $\overline{1} = 0$.

Assume, to get a contradiction, that x is not a subword of y and also assume that $|y| = m$ and $|x| = n$. If x is a subword of yty for each $t \in \Sigma^*$ with $|t| = 3$, then for each such t we have $f(x, yty) \leq m+3$ and $f(x, yty) + n - 1 \geq m + 1$. Since the position of x in yty cannot be the same for all valid t, let $t_0 = a_0b_0c_0$ be the choice of t for which $f(x, yty)$ is greatest across all valid t and let $t_4 = a_4b_4c_4 \neq t_0$ be the choice of t for which $f(x, yty)$ is smallest across all valid t. We now consider two cases depending on the position of x in yt_0y.

Case (i): $f(x, yt_0y) = m + 3$. Consider $t_1 = \overline{a_4}0\overline{c_0}$ and $t_2 = \overline{a_4}1\overline{c_0}$. Since t_1 and t_2 differ from t_4 in the first index of t_4, and t_4 gives the leftmost position for x as a subword of yty over all valid choices of t, we know $f(x, yt_1y) \neq f(x, yt_4y)$ and $f(x, yt_2y) \neq f(x, yt_4y)$. Similarly we have $f(x, yt_1y) \neq f(x, yt_0y)$ and $f(x, yt_2y) \neq f(x, yt_0y)$. Applying Lemma 42 to the pairs t_4, t_1 and t_4, t_2 and Lemmas 42 and 43 to the pairs t_1, t_0 and t_2, t_0 we have that $f(x, yt_1y) + n - 1 \geq m + 3$ and $f(x, yt_2y) + n - 1 \geq m + 3$ and that $f(x, yt_1y) \leq m + 1$ and $f(x, yt_2y) \leq m + 1$. So for yt_1y (resp., yt_2y), the position of x is such that it entirely overlaps t_1 (resp., t_2). But since $t_1 \neq t_2$ we know that the positions of x as a subword of yt_1y and yt_2y are distinct, i.e., $f(x, yt_1y) \neq f(x, yt_2y)$.

So suppose without loss of generality that $f(x, yt_1y) > f(x, yt_2y)$. We now perform an index chasing argument, similar to that of the ternary case, using t_0, t_1, t_2 and seeking the contradiction $c_0 = (yt_0y)[m + 3] = (yt_2y)[m + 3] = \overline{c_0}$. We use the same labeling scheme as in the ternary case. So define i, j, k such that $i = m + 4 - f(x, yt_0y)$ and $j = m+4-f(x, yt_1y)$ and $k = m+4-f(x, yt_2y)$, giving $x_0[i] = t_0[3] = c_0$ and $x_1[j] = t_1[3] = \overline{c_0}$ and $x_2[k] = t_2[3] = \overline{c_0}$. Note that in this case we have $i = 1$ by assumption and $j \geq i + 3$ by Lemmas 42 and 43. From Figure 3.3 we obtain the following identities.

$$x_1[\ell] = y[\ell + m + 3 - j] \text{ for } 1 \leq \ell \leq j - 3; \tag{3.5}$$
$$x_2[\ell] = y[\ell + m + 3 - k] \text{ for } 1 \leq \ell \leq k - 3; \tag{3.6}$$
$$x_0[\ell] = y'[\ell - i] \text{ for } i + 1 \leq \ell \leq n; \tag{3.7}$$
$$x_1[\ell] = y'[\ell - j] \text{ for } j + 1 \leq \ell \leq n. \tag{3.8}$$

Since $j + 1 \leq k \leq n$, we can apply (3.8) to get $x_1[k] = y'[k - j]$. Then, since $i \leq j$ and $k \leq n$, we have $i + k \leq n + j$, giving $i + k - j \leq n$. This, together with the inequality $k-j \geq 1$ giving $i+1 \leq i+k-j$, means that we can apply (3.7) to get $y'[k-j] = x_0[i+k-j]$. Next, $j-i \geq 3$ gives $i+k-j \leq k-3$, so applying (3.6) gives $x_2[k+i-j] = y[i+m+3-j]$.

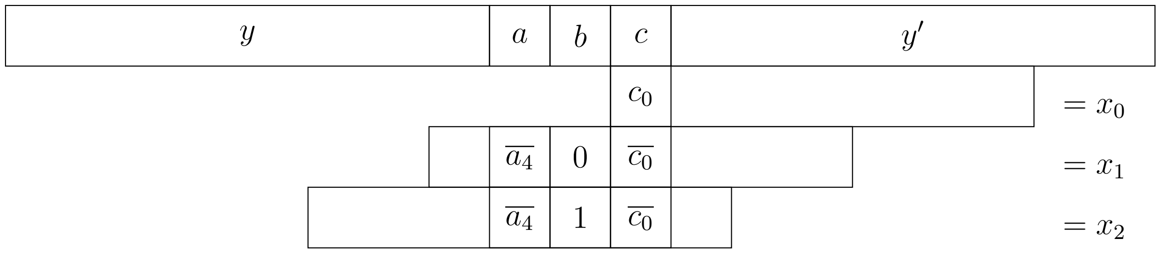

Figure 3.3: Positions of x in yt_0y, yt_1y, yt_2y for Case (i)

Finally, since $1 \le i \le j-3$, we can apply (3.5) to get $y[i+m+3-j] = x_1[i]$. Together this gives the contradiction

$$\overline{c_0} = x_2[k] = y'[k-j] = x_0[i+k-j] = y[i+m+3-j] = x_2[i] = c_0.$$

Case (ii): $f(x, yt_0y) \le m+2$. Consider $t_1 = \overline{a_4 b_0} c_0$ and $t_2 = \overline{a_4} b_0 \overline{c_0}$ and $t_3 = \overline{a_4 b_0 c_0}$. By the same argument as in Case 1 we get that the positions of x as a subword of each of $yt_0y, yt_1y, yt_2y, yt_3y, yt_4y$ are all distinct. Furthermore, by Lemma 42 we have that for each pair t_i, t_j with $0 \le i, j \le 4$ and $i \ne j$ the difference in positions of x as a subword of yt_iy and yt_jy is $|f(x, yt_iy) - f(x, yt_jy)| \ge 2$. We now order these choices of t and relabel t_1, t_2, t_3 if necessary such that $f(x, yt_0y) > f(x, yt_3y) > f(x, yt_1y) > f(x, yt_2y) > f(x, yt_4y)$. At this point we again perform an index-chasing argument using t_0, t_1, t_2. If we have that $t_2[3] = \overline{c_0}$ then the argument given in Case (i) holds to give a contradiction. If instead we have that $t_2[3] = c_0$, then we know that $t_2[2] = \overline{b_0}$ and we will get the contradiction $b_0 = (yt_0y)[m+2] = (yt_2y)[m+2] = \overline{b_0}$. To do this we define i, j, k such that $i = m+3 - f(x, yt_0y)$ and $j = m+3 - f(x, yt_1y)$ and $k = m+3 - f(x, yt_2y)$, giving $x_0[i] = t_0[2] = b_0$ and $x_1[j] = t_1[2]$ and $x_2[k] = t_2[2] = \overline{b_0}$. Since $f(x, yt_0y) > f(x, yt_3y) > f(x, yt_1y)$, Lemma 42 gives $f(x, yt_0y) \ge f(x, yt_3y) + 2$ and $f(x, yt_3y) \ge f(x, yt_1y) + 2$. So $f(x, yt_0y) \ge f(x, yt_1y) + 4$ and thus $j \ge i + 4$. From Figure 3.4 we get the following identities. Note that these identities are centered around $t[2]$ instead of $t[3]$ as in Case 1.

$$x_1[\ell] = y[\ell + m + 2 - j] \text{ for } 1 \le \ell \le j-2; \tag{3.9}$$
$$x_2[\ell] = y[\ell + m + 2 - k] \text{ for } 1 \le \ell \le k-2; \tag{3.10}$$
$$x_0[\ell] = y'[\ell - i - 1] \text{ for } i+2 \le \ell \le n; \tag{3.11}$$
$$x_1[\ell] = y'[\ell - j - 1] \text{ for } j+2 \le \ell \le n. \tag{3.12}$$

Since $j+2 \le k \le n$ we can apply (3.12) to get $x_1[k] = y'[k-j-1]$. Then since $i \le j$ and $k \le n$ we have $i+k \le n+j$, giving $i+k-j-1 < i+k-j \le n$. This, together with

$k-j \geq 2$ giving $i+2 \leq i+k-j$ by Lemma 42, means that we can apply (3.11) to get $y'[k-j-1] = x_0[i+k-j]$. Next, $j-i \geq 4$ gives $i+k-j \leq k-4 < k-2$, so applying (3.10) gives $x_2[k+i-j] = y[i+m+2-j]$. Finally, since $1 \leq i \leq j-4 < j-2$, we can apply (3.9) to get $y[i+m+2-j] = x_1[i]$. Together this gives the contradiction

$$\overline{b_0} = x_2[k] = y'[k-j-1] = x_0[i+k-j] = y[i+m+2-j] = x_2[i] = b_0.$$

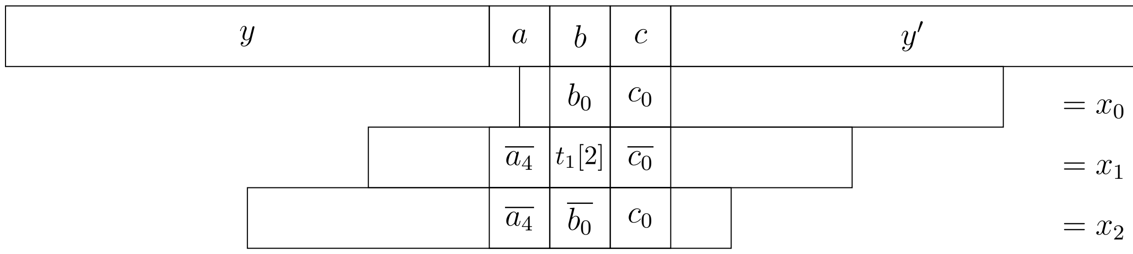

Figure 3.4: Positions of x in yt_0y, yt_1y, yt_2y for Case (ii) where $t_2[3] = c_0$

□

Theorem 45. *Let $\Sigma = \{0,1\}$. Then x is a subword of every y-bordered word if and only if x is a subword of yty for all words t of length 3.*

Proof. If x is a subword of every y-bordered word, then clearly x is a subword of yty for all words t of length 3. For the other direction there are two cases.

Case 1: $x \notin A$. Then by Theorem 44 we know x is a subword of y. So x is also a subword of every y-bordered word.

Case 2: $x \in A$. Then x has the form 01^i, or 0^i1, or 10^i, or 1^i0 for some $i \geq 1$. We consider the case where $x = 01^i$ and note that the case where $x = 1^i0$ follows by a symmetric argument and the other cases are given by relabeling 0 to 1 and 1 to 0. If x is a subword of y then the result follows trivially. So suppose that $x = 01^i$ is not a subword of y, but that x is a subword of yty for all words t of length 3. Then, since x is a subword of $y000y$, we have that 1^i is a prefix of y. Additionally, since x is a subword of $y111y$, we know that 01^j is a suffix of y for some j satisfying $0 \leq j < i$. So we have $y = 1^iw01^j$ for some $w \in \Sigma^*$. Now consider a y-bordered word z. Let k be the index of the first 0 in z. Since z has y as a prefix and a suffix, and $z \neq y$, we know that $|z| \geq |y| + k$. This is because the y-suffix of z must start after the first 0 in z. So we have that there are i consecutive 1's in z starting at some index $\ell > k$. Let k' be the largest index less than ℓ such that $z[k'] = 0$. Then $z[k'..k'+i] = 01^i$. So x is a subword of z. □

Remark 46. The number 3 is optimal in Theorem 45. Consider $x = 10100$, $y = 01001010$. Then x is a subword of every y-bordered word of length $\le 2|y|+2 = 18$, but not a subword of yty with $t = 110$.

3.3 Finiteness

We now examine when $L_{x=y}$ is finite.

Theorem 47. *Let $x, y \in \Sigma^+$. Then $L_{x=y}$ is finite if and only if $|\Sigma| = 1$ and $x \neq y$.*

Proof. There are four cases to consider.

Case (i): $x = a^i$ and $y = a^j$ for integers $i, j > 0$. If $|\Sigma| = 1$, then $L_{x=y}$ is finite if and only if $x \neq y$, for otherwise without loss of generality $i < j$, and for $n \ge j$ the word a^n contains $n - j + 1$ occurrences of a^j, but $n - i + 1$ occurrences of a^i.

Otherwise $|\Sigma| > 1$. Let $b \in \Sigma$ and $b \neq a$. Then for each $z \in b^*$ we have $|z|_x = |z|_y = 0$. Thus $L_{x=y}$ is infinite.

Case (ii): $x = a^i$ and $y = b^j$ for two distinct symbols a, b and $i, j > 0$. Then for each z of the form $(xy)^n$ we have $|z|_x = |z|_y = n$. Thus $L_{x=y}$ is infinite.

Case (iii): $x = a^i$ for some $i > 0$ but y contains two different symbols. Let $b \in \Sigma$ with $b \neq a$. Then for each $z \in b^*$ we have $|z|_x = |z|_y = 0$. Thus $L_{x=y}$ is infinite.

Case (iv): x and y both contain two different symbols. Let $a \in \Sigma^*$. Then for each $z \in a^*$ we have $|z|_x = |z|_y = 0$. Thus $L_{x=y}$ is infinite. □

We could consider the generalization of $L_{x=y}$ to more than two words:

$$L_{x_1=x_2=\cdots=x_n} = \{z \in \Sigma^* \ : \ |z|_{x_1} = |z|_{x_2} = \cdots = |z|_{x_n}\}.$$

The following examples show that deciding the finiteness of $L_{x_1=x_2=\cdots=x_n}$ for $n \ge 3$ is more subtle than the case $n = 2$. Suppose $\Sigma = \{0, 1\}$. Then $L_{0=1=00=11}$ and $L_{0=1=01=10}$ are finite languages, but $L_{00=11=000=111}$ is not.

Consider $L_{0=1=00=11}$. For any maximal subword consisting of 0's, the number of 0's exceeds the number of 00's, and similar for 1 and 11. So $L_{0=1=00=11} = \{\epsilon\}$.

Consider $L_{0=1=01=10}$. Since $|z|_{01} = |z|_{10}$, as shown in Figure 3.1, the words in this language must start and end with the same character. There cannot be a 00 or the number of 0's exceeds that of 01 and 10, and similar for 11. So, the language is a subset of $(01)^*0 \cup (10)^*1 \cup \{\epsilon\}$. But no word z in this language, other than ϵ, has $|z|_0 = |z|_1$. Therefore, $L_{0=1=01=10} = \{\epsilon\}$.

Consider $L_{00=11=000=111}$. It contains $(01)^*$, and hence is infinite.

Lacking a general condition for finiteness, we prove the following sufficient condition.

Theorem 48. *If $|x_1| = \cdots = |x_n|$ then $L_{x_1=x_2=\cdots=x_n}$ is infinite.*

Proof. Let $\ell = |x_1|$. Consider the cyclic order-ℓ de Bruijn word w of length k^ℓ over the cardinality-k alphabet Σ. Such a word is guaranteed to exist for all $k \geq 2$ and $\ell \geq 1$; see, e.g., [35]. Let w' be the prefix of w of length $\ell - 1$. Then $w^i w' \in L_{x_1=x_2=\cdots=x_n}$ for all $i \geq 1$. □

3.4 Counting Subwords and Additive Bases

To finish this chapter and tie together the work in this thesis, we briefly consider applying the additive basis ideas of Chapter 2 to languages defined by comparing the counts of occurrences of subwords, as in the present chapter. For languages $L_{x<y}, L_{x\leq y}, L_{x=y}$ over alphabet Σ_2 where x is interlaced by y, or y is interlaced by x it is easy to state results about whether the corresponding sets $[L_{x<y}]_2, [L_{x\leq y}]_2, [L_{x=y}]_2$ form asymptotic additive bases, and the order of the bases if they exist. This is because by Theorem 36 $L_{x<y}, L_{x\leq y}, L_{x=y}$ are then all regular, and we can use the algorithms given in [4] and [18] to evaluate the criteria for forming an (asymptotic) additive basis and then directly use `Walnut` to determine the order of the basis, provided that the associated automaton has few enough states for the computation to be feasible.

For example, in Section 3.2 we saw that $L_{01=10}$ is a regular language. We can construct an automaton accepting all base-2 representations of numbers with an equal number of occurrences of 01 and 10 in their base-2 representations. This gives a 3-state automaton and using `Walnut` we find that the corresponding set forms an additive basis of order 2. On the other hand, $[L_{01>10}]_2 = \emptyset$ and the greatest common divisor of the elements of $[L_{01<01}]_2$ is not 1. This means that none of $[L_{01<10}]_2, [L_{01>10}]_2, [L_{01\neq 10}]_2$ form (asymptotic) additive bases of finite order.

We can also consider a natural generalization of the sets considered in Chapter 2, listed in Table 2.2, and ask whether the sets $[L_{00<11}]_2, [L_{00\leq 11}]_2, [L_{00=11}]_2$, and their complements, or even more generally the sets $[L_{0^n<1^m}]_2, [L_{0^n\leq 1^m}]_2, [L_{0^n=1^m}]_2$, and their complements, form (asymptotic) additive bases for $n, m \geq 1$. Given that 0^n does not interlace 1^m, nor vice versa, for every $n, m \geq 1$, we cannot use `Walnut` directly. Thus, to attain results on whether these sets form (asymptotic) additive bases we need to rely on methods presented in Section 2.3.1, or otherwise. For example, we obtain the following result on the sets $[L_{0^n=1^m}]$ for all $n, m \geq 3$.

Theorem 49. *For all integers $n, m \geq 3$ the set $[L_{0^n=1^m}]_2$ forms an additive basis of order 2.*

Proof. The language $[L_{0^n=1^m}]_2$ is the set of all natural numbers whose canonical base-2 representations have an equal number of occurrences of 0^n and 1^m as subwords. If $n, m \geq 3$ we can observe that the regular language, R, consisting of all words that do not have either 000 or 111 as a subword is a subset of $L_{0^n=1^m}$ for all $n, m \geq 3$. The state diagram for an automaton recognizing 0^*R is given in Figure 3.5.

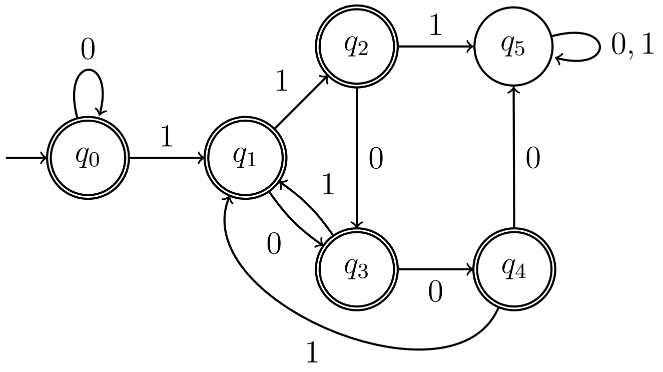

Figure 3.5: Automaton accepting 0^*R

After writing this automaton to a file, `NO3.txt` for `Walnut` to work with, we can get the set of number representable as the sum of two elements with base-2 representations in 0^*R, using the `Walnut` the command

```
eval no3 "E x1,x2 (NO3[x1]=@1)&(NO3[x2]=@1)&(n=x1+x2)":.
```

This returns an automaton accepting all n. Therefore, $[L_{0^n=1^m}]_2$ forms an additive basis of order 2 for all $n, m \geq 3$. □

For the remaining values of n, m it is more difficult to find regular approximations that give optimal additive basis orders. For $n = 1$ and small values of m, empirical results suggest that $[L_{0^n=1^m}]_2$ has asymptotic additive basis order 3. For example, for $m = 2, 3, 4$, the largest values less than 500,000 that cannot be represented as the sum of 3 elements of $[L_{0=1^m}]_2$ are 49, 220, and 1154, respectively. Similarly, for $m = 1$ and $n = 2, 3, 4$, the largest values less than 500,000 that cannot be represented as the sum of 3 elements of $[L_{0^n=1}]_2$ are 239, 2047, and 32,766, respectively.

Fixing $n = 2$ (resp., $m = 2$) and considering small values of m (resp., n), empirical results suggest that $[L_{0^n=1^m}]_2$ form asymptotic additive bases of order 2. We highlight one curious example given by $n = 2, m = 3$. We can verify by a brute-force computation that, with the exception of 77, every natural number less than 500,000 can be represented as the sum of at most 2 elements of $[L_{00=111}]_2$. However, proving this using a regular underapproximation and `Walnut` seems to be difficult. We can, however at least prove the following optimal result on the (non-asymptotic) additive basis order of $[L_{00=111}]_2$.

Theorem 50. *Every natural number is the sum of at most 3 elements of* $[L_{00=111}]_2$.

Proof. Consider the regular language $R_{00=111}$ given by $(((1{+}11)0)^*(111(0(1{+}11))^*00)^*)^*((1{+}11)0)^*(\varepsilon + 1 + 11)$. Using the `Walnut` command

```
eval qq23 "E x,y,z (QQ23[x]=@1)&(QQ23[y]=@1)&(QQ23[z]=@1)&(n=x+y+z)":
```

we get an automaton accepting the base-2 representations of all n. □

The set of numbers with base-2 representations in $R_{00=111}$ do not form an asymptotic additive basis of order 2. If we consider regular subsets of $L_{00=111}$ that are supersets of $R_{00=111}$, we run into issues with evaluating the `Walnut` commands, due to the significant time and memory requirements.

3.5 Open Problems

We conclude this chapter with some open problems related to this work.

1. Find an algorithm that decides if $L_{x_1=\cdots=x_n}$ is finite, given $x_1, \ldots, x_n$.
2. Characterize when $L_{x_1=\cdots=x_n}$ is finite.
3. Find a more straightforward and less case-based proof of Theorem 45.
4. Does $[L_{0^n=1^m}]_2$ form an asymptotic additive basis of order 3 for $n = 1$ and $m \geq 1$ and for $m = 1$ and $n \geq 1$? Similarly, Does $[L_{0^n=1^m}]_2$ form an asymptotic additive basis of order 2 for $n = 2$ and all $m \geq 2$ and for $m = 2$ and all $n \geq 2$?

References

[1] J. P. Allouche and J. Shallit. The ubiquitous Prouhet-Thue-Morse sequence. In *Sequences and their applications*, pages 1–16. Springer, 1999.

[2] J. P. Allouche and J. Shallit. *Automatic sequences: theory, applications, generalizations*. Cambridge University Press, 2003.

[3] W. D. Banks. Every natural number is the sum of forty-nine palindromes. *Integers*, 16(A3), 2016.

[4] J. P. Bell, K. E. Hare, and J. Shallit. When is an automatic set an additive basis? *Proc. Amer. Math. Soc., Series B*, 5(6):50–63, 2018.

[5] J. P. Bell, T. F. Lidbetter, and J. Shallit. Additive number theory via approximation by regular languages. In M. Hoshi and S. Seki, editors, *Developments in Language Theory (DLT 2018)*, volume 11088 of *Lecture Notes in Computer Science*, pages 121–132. Springer, 2018.

[6] J. P. Bell, T. F. Lidbetter, and J. Shallit. Additive number theory via approximation by regular languages. Preprint available at https://arxiv.org/abs/1804.07996, 2018.

[7] J. R. Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

[8] J. R. Büchi. On a decision method in restricted second order arithmetic. In *The Collected Works of J. Richard Büchi*, pages 425–435. Springer, 1990.

[9] A. L. Cauchy. *Démonstration du Theoreme Général de Fermat sur les Nombres Polygones*, volume 6 of *Cambridge Library Collection - Mathematics*, pages 320–353. Cambridge University Press, 2009.

[10] J. Cilleruelo, F. Luca, and L. Baxter. Every positive integer is a sum of three palindromes. *Math. Comp.*, 87(314):3023–3055, 2018.

[11] C. J. Colbourn, R. E. Dougherty, T. F. Lidbetter, and J. Shallit. Counting subwords and regular languages. In M. Hoshi and S. Seki, editors, *Developments in Language Theory (DLT 2018)*, volume 11088 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2018.

[12] C. J. Colbourn, R. E. Dougherty, T. F. Lidbetter, and J. Shallit. Counting subwords and regular languages. Preprint available at https://arxiv.org/abs/1804.11175, 2018.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, 2nd edition, 2001.

[14] A. Ehrenfeucht and D. M. Silberger. Periodicity and unbordered segments of words. *Discrete Math.*, 26:101–109, 1979.

[15] C. C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Amer. Math. Soc.*, 98(1):21–51, 1961.

[16] N. J. A. Sloane et al. The on-line encyclopedia of integer sequences. Available at https://oeis.org, 2018.

[17] G. Gamard, G. Richomme, J. Shallit, and T. J. Smith. Periodicity in rectangular arrays. *Inform. Process. Lett.*, 118:58–63, 2017.

[18] P. Gawrychowski, D. Krieger, N. Rampersad, and J. Shallit. Finding the growth rate of a regular or context-free language in polynomial time. *Internat. J. Found. Comput. Sci.*, 21(4):597–618, 2010.

[19] M. Le Gonidec. On the complexity of a family of k-context-free sequences. *Theoret. Comput. Sci.*, 414:47–54, 2012.

[20] J. Hartmanis and H. Shank. On the recognition of primes by automata. *J. ACM*, 15:382–389, 1968.

[21] D. Hilbert. Beweis für den Darstellbarkeit der ganzen Zahlen durch eine feste Anzahl n-ter Potenzen. *Math. Ann.*, 67:281–300, 1909.

[22] B. R. Hodgson. Décidabilité par automate fini. *Ann. Math. Qué.*, 7(1):39–57, 1983.

[23] S. Horváth, J. Karhumäki, and J. Kleijn. Results concerning palindromicity. *J. Inf. Process. Cybern. EIK*, 23:441–451, 1987.

[24] D. M. Kane, C. Sanna, and J. Shallit. Waring's theorem for binary powers. *arXiv preprint arXiv:1801.04483*, 2018.

[25] M. Lothaire. *Combinatorics on Words*, volume 17 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, 1983.

[26] R. C. Lyndon and M. P. Schützenberger. The equation $a^M = b^N c^P$ in a free group. *Michigan Math. J.*, 9:289–298, 1962.

[27] P. Madhusudan, D. Nowotka, A. Rajasekaran, and J. Shallit. Lagrange's theorem for binary squares. In Igor Potapov, Paul Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[28] Y. Moshe. On some questions regarding k-regular and k-context-free sequences. *Theoret. Comput. Sci.*, 400(1-3):62–69, 2008.

[29] H. Mousavi. Automatic theorem proving in Walnut. Preprint available at https://arxiv.org/abs/1603.06017, 2016.

[30] M.B. Nathanson. *Additive Number Theory The Classical Bases*, volume 164. Springer Science & Business Media, 2013.

[31] R. J. Parikh. On context-free languages. *J. ACM*, 13:570–581, 1966.

[32] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetic ganzer Zahlen, in wlechem die Addition als einzige Operation hervortritt. In *Sprawozdanie z I Kongresu Matematyków Krajów Słowiańskich*, pages 92–101. Warsaw, 1929.

[33] M. Presburger. On the completeness of a certain system of arithmetic of whole numbers in which addition occurs as the only operation. *History and Philosophy of Logic*, 12(2):225–233, 1991. D. Jacquette, trans.

[34] A. Rajasekaran, J. Shallit, and T. Smith. Sums of palindromes: an approach via automata. In R. Niedermeier and B. Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*, volume 96 of *Leibniz International Proceedings in Informatics*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2018.

[35] A. Ralston. De Bruijn sequences — a model example of the interaction of discrete mathematics and computer science. *Math. Mag.*, 55:131–143, 1982.

[36] D. R. Raymond and D. Wood. Grail: A C++ library for automata and expressions. *J. Symbolic Comput.*, 17:341–350, 1994.

[37] L. Schaeffer. Deciding properties of automatic sequences. Master's thesis, University of Waterloo, 2013.

[38] J. Shallit. *A second course in formal languages and automata theory.* Cambridge University Press, 2008.

[39] T. N. Shorey and C. L. Stewart. On the diophantine equation $ax^{2t} + bx^t y + cy^2 = d$ and pure powers in recurrence sequences. *Math. Scand.*, 52:24–36, 1983.

[40] M. Sipser. *Introduction to the theory of computation.* Thomson Course Technology Boston, 3rd edition, 2012.

[41] R. C. Vaughan and T. D. Wooley. Waring's problem: a survey. In M. A. Bennett, B. C. Berndt, N. Boston, H. G. Diamond, A. J. Hildebrand, and W. Philipp, editors, *Number Theory for the Millennium*, volume 3, pages 301–340. A. K. Peters, 2002.