# Automata and Ratio Sets

by

Joseph Victor Fiorillo Meleshko

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, 2022

## Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

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

## Statement of Contributions

This thesis contains content from two papers of which I, Joseph Victor Fiorillo Meleshko, am a co-author. The algorithms and results in Section 3.1 are taken from my joint paper in preparation [8]. The algorithms and results in Section 3.2 are taken, some components verbatim, from my joint paper [3].

**Abstract**

This thesis explores the composition of ratio sets, the subsets of the rationals derived from the quotients of two sets of natural numbers, and examines a variety of specific examples where the comprising sets of natural numbers have specific properties. I present a general algorithm that decides the inclusion of a rational number in a specific ratio set if the comprising sets of natural numbers are a regular language when represented in a given base. I also present an algorithm for deciding the inclusion of a rational number in the ratio set of a few select sets of natural numbers that are not a regular language when represented in any base, namely, the set of natural numbers with representations in a specific base that are palindromes or antipalindromes. Using those algorithms, I examine some of the rational numbers in specific ratio sets and then prove several results regarding the composition of those ratio sets. As well, I present algorithms for computing approximations to real numbers using elements of some specific ratio sets.

**Acknowledgements**

I would first like to express my gratitude to my supervisor Jeffrey Shallit, for steadfastly improving my mathematical writing through a chaotic two years. I would also like to thank Kevin Hare and Bin Ma, for their valuable time invested reading my thesis.

I would like to explicitly thank my mother, Adeline Fiorillo, for her unyielding support despite being so far away. Lastly, I would like to thank my multitudinous friends, whose names I converted into binary, rounded to the nearest palindrome, and then added together:

$$14126977053803366083502516009900024519 4$$

You are the iron supports that kept me upright through the last two years.

## Dedication

To my late grandfather, Vittorio Fiorillo.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

This thesis primarily explores ratio sets, the subsets of the rationals derived from the quotients of two sets of natural numbers, and examines a variety of specific examples where the comprising sets of natural numbers have specific properties. In service of this goal, I present several different algorithms for testing inclusion in a given ratio set and compare their effectiveness and efficiency. As well, I present algorithms for computing approximations to real numbers using elements of some specific ratio sets.

The primary contributions of this thesis are divided into two broad categories. Chapter 2 discusses a general framework for calculating ratio sets when the comprising sets of natural numbers can be expressed as regular languages over the digits of a given base and presents several results regarding such ratio sets. Chapter 3 focuses on ratio sets of natural numbers that have representations in a specific base which are palindromes and antipalindromes, two properties that cannot be expressed as a regular language.

Each algorithm presented was implemented in Python and those implementations and all computed datasets used in this thesis are available in a selection of repositories on my GitHub. There are links to specific repositories and files wherever relevant throughout the remainder of this thesis.

## 1.1 Preliminaries

### 1.1.1 Words and Representations

A fundamental concept for this thesis is the notion of a *word*, frequently known in the literature as a *string*. A word is a sequence of *symbols* from a set of symbols called an *alphabet*. In this thesis, all words and alphabets are finite. A *language* is a finite or infinite set of words over an alphabet. For an alphabet $\Sigma$, the language $\Sigma^*$ is the set of all finite words over the alphabet $\Sigma$. Similarly, the language $\Sigma^+$ is the set of all non-empty finite words over the alphabet $\Sigma$. This thesis primarily studies words over the alphabet $\Sigma_k = \{0, 1, \ldots, k-1\}$ for some natural number $k \geq 2$ and we occasionally refer to those symbols as *digits*. We take special notice of $\Sigma_2 = \{0, 1\}$, which we call the *binary* alphabet.

Let $x = x_0 x_1 \cdots x_i$ and $y = y_0 y_1 \cdots y_j$ be words. The number of symbols of $x$ is written as $|x|$ and referred to as the *length* of $x$. The word with length zero is called the *empty word* and denoted by $\epsilon$. For $0 \leq m \leq n \leq i$, we call $x[m..n] = x_m x_{m+1} \cdots x_n$ a *factor* or *subword* of $x$. We also define $x[n..n] = x[n] = x_n$. If $n < m$, then we define $x[m..n] = \epsilon$. We denote the reverse of $x$ as $x^R = x_i x_{i-1} \cdots x_0$ and define $\epsilon^R = \epsilon$. Also, we denote the *concatenation* of $x$ and $y$ as $xy = x_0 x_1 \cdots x_i y_0 y_1 \cdots y_j$. Given a natural number $n$, we write the concatenation of $n$ copies of $x$ as $x^n$ and define $x^0 = \epsilon$.

This thesis is mainly concerned with natural numbers with specific properties when viewed as words, so we need to formalize a correspondence between words and natural numbers. We have an intuitive interpretation of a word $x = x_i x_{i-1} \cdots x_0$ over the alphabet $\Sigma_{10} = \{0, 1, \ldots, 9\}$. For example, we interpret the word "4956" as the natural number $4 \cdot 10^3 + 9 \cdot 10^2 + 5 \cdot 10^1 + 6 \cdot 10^0$. More generally, we understand $x$ to refer to the natural number

$$\sum_{j=0}^{i} x_j \cdot 10^j.$$

We call this the *most-significant-digit-first base-10 representation* since the base of each exponent is 10 and the largest exponent corresponds to the first digit reading left to right.

The base-10 representation can be generalized by interpreting a word $x = x_i x_{i-1} \cdots x_0$ over $\Sigma_k$ as the natural number

$$\sum_{j=0}^{i} x_j \cdot k^j.$$

Here we are viewing $x$ as a *most-significant-digit-first base-k representation*. When $k = 2$, we call this representation *binary*. If we interpret the word "1101" as binary, then we get

the natural number

$$1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 0 + 1 = 13.$$

However, if we view the word "1101" as a most-significant-digit-first base-3 representation, then we would get the natural number

$$1 \cdot 3^3 + 1 \cdot 3^2 + 0 \cdot 3^1 + 1 \cdot 3^0 = 27 + 9 + 0 + 1 = 37.$$

To resolve this ambiguity, we write $[x]_k$ when we want to interpret the word $x$ as a most significant first base-$k$ representation of a natural number. Hereafter, we omit quotation marks when discussing words as distinguishing between words and natural numbers should be clear from context. For example, we have that $[1101]_2$ is the natural number 13 and $[1101]_{10}$ is the natural number 1101 but the 1101 within each pair of square brackets is a word. Given a language $L \subseteq \Sigma_k$, we define the subset of natural numbers $[L]_k = \{[x]_k \mid x \in L\}$.

For a word $x = x_i x_{i-1} \cdots x_0$ over some $\Sigma_k$, we call $x_i$ the *leading digit*. We note that we have multiple words corresponding to the same natural number when viewed as a most-significant-digit-first base-$k$ representation. For words over the alphabet $\Sigma_2$ and $j \geq 0$,

$$[1101]_2 = [01101]_2 = [0^j 1101]_2 = 13.$$

To resolve this ambiguity, we call words with non-zero leading digit *canonical representations*. Each natural number $n$ has a unique canonical most-significant-digit-first base-$k$ representation that we write as $\langle n \rangle_k$. For example, the canonical most-significant-digit-first base-2 representation of 13 is the word $\langle 13 \rangle_2 = 1101$.

Note that for all natural numbers $k \geq 2$, we define the canonical representation $\langle 0 \rangle_k = \epsilon$. Given a subset of the natural numbers $A$, we define the language $\langle A \rangle_k = \{\langle n \rangle_k \mid n \in A\}$. Given a word $x = x_i x_{i-1} \cdots x_0 \in \Sigma_k^*$, typically a base-$k$ representation, we define the complement of $x$ as $\overline{x} = \overline{x_i}\,\overline{x_{i-1}} \cdots \overline{x_0}$ where $\overline{\sigma} = k - 1 - \sigma$ for $\sigma \in \Sigma_k$.

There is another basic way to create maps between words and natural numbers. A word $x = x_i x_{i-1} \cdots x_0$ is a *least-significant-digit-first base-$k$ representation* for the natural number

$$\sum_{j=0}^{i} x_j \cdot k^{i-j}.$$

For example, we have that

$$[231]_5 = 2 \cdot 5^2 + 3 \cdot 5^1 + 1 \cdot 5^0 = 50 + 15 + 1 = 66,$$

but the word 231 is a least-significant-digit-first base-5 representation for the natural number

$$2 \cdot 5^0 + 3 \cdot 5^1 + 1 \cdot 5^2 = 2 + 15 + 25 = 42.$$

The *canonical least-significant-digit-first base-k representation* of a natural number is the unique least-significant-digit-first base-$k$ representation that has a non-zero final digit. Analogously, we define the canonical least-significant-digit-first base-$k$ representation of 0 to be $\epsilon$. We can convert between most-significant-digit-first base $k$ representation and least-significant-digit-first base $k$ representation with the same $k$ by simply reversing the representation. Hereafter, we assume all representations are most-significant-digit-first unless otherwise specified.

A natural extension to the typical base-$k$ representation is to allow more symbols than the usual $\Sigma_k$. Loxton and van der Poorten examined base-4 representations with the standard alphabet $\Sigma_4$ replaced by $\Sigma = \left\{\overline{1}, 0, 1, 2\right\}$, which we call the "Awful" representation [19]. Here, $\overline{1}$ is a more convenient symbol for $-1$. They note that every natural number has an "Awful" representation, though some differ wildly from the standard base-4 representation. For example, the natural number 10 is written as 22 in the "Awful" representation, which is identical to the usual base-4 representation. The typical base-4 representation of 11 is 23 but 3 is not a valid symbol in the "Awful" representation. Hence, the "Awful" representation of the natural number 11 is written as $1\overline{1}\,\overline{1}$ since

$$1 \cdot 4^2 + (-1) \cdot 4^1 + (-1) \cdot 4^0 = 16 + (-4) + (-1) = 11.$$

Another non-standard representation relevant to this thesis is the *Fibonacci representation*. Recall the Fibonacci numbers where $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$. The Fibonacci representation uses the binary alphabet but instead the word $x = x_i x_{i-1} \cdots x_0$ corresponds to the natural number

$$\sum_{j=0}^{i} x_j \cdot F_{j+2}$$

which we denote by $[x]_F$. For example, we have that

$$[10101]_F = 1 \cdot F_6 + 0 \cdot F_5 + 1 \cdot F_4 + 0 \cdot F_3 + 1 \cdot F_2 = 8 + 3 + 1 = 12.$$

Some natural numbers have multiple Fibonacci representations such as

$$8 = [10000]_F = [1100]_F = [1011]_F.$$

However, Fibonacci representation is unique with the additional requirement that representations have no adjacent ones [17, 36]. Fibonacci representation also inspires the definition of the *Fibbinary numbers*. The Fibbinary numbers are the natural numbers that do not have adjacent ones in their binary representation. We denote the set by $\texttt{FIB} = \{0, 1, 2, 4, 5, 8, 9, 10, 16, 17, \ldots\}$. The Fibbinary numbers form sequence A003714 in the *On-Line Encyclopedia of Integer Sequences* (OEIS).

### 1.1.2 Automata, Languages, and Decision Problems

An important classification for languages is the notion of a *regular language*. Regular languages are often defined in terms of regular expressions, but for the purposes of this thesis it is beneficial to define regular languages in terms of *deterministic finite automata*.

**Definition 1.** A deterministic finite automaton is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q$ is a set of states,
- $\Sigma$ is a finite set called the *input alphabet*,
- $\delta : Q \times \Sigma \to Q$ is a function called the *transition function*. We also define $\delta(q, xy) = \delta(\delta(q, x), y)$ for words $x, y \in \Sigma^+$,
- $q_0 \in Q$ is called the *initial state*, and
- $F \subseteq Q$ is a subset called the *accepting states*.

We say a word $x = x_0 x_1 \cdots x_i$ over $\Sigma$ is *accepted* by a deterministic finite automaton $M$ if $\delta(q_0, x) \in F$. Otherwise, we say the word $x$ is *rejected*. Define $L(M)$ as the language of words accepted by an automaton $M$. We say that $M$ *recognizes* $L(M)$. A language $L$ is a regular language if and only if there exists an deterministic finite automaton such that $L = L(M)$. We define the *size* of an automaton $M$ as its number of states $|Q|$ and we denote the quantity as $|M|$. We occasionally draw an automaton as a directed graph called a *transition diagram*. An example of such a drawing is given in Figure 1.1. We allow $\delta$ to be a partial function. In these cases, we implicitly reject any input where the result of $\delta$ is undefined at any step.

**Example 2.** We construct an automaton to recognize the language

$$\langle \texttt{FIB} \rangle_2 = \{\epsilon, 1, 10, 100, 101, 1000, 1001, 1010, 10000, 10001, \ldots\}.$$

| $q$ | $\delta(q,0)$ | $\delta(q,1)$ |
|---|---|---|
| $q_0$ | $q_3$ | $q_1$ |
| $q_1$ | $q_2$ | $q_3$ |
| $q_2$ | $q_2$ | $q_1$ |
| $q_3$ | $q_3$ | $q_3$ |

Table 1.1: Transition table of $\delta$ for the automaton that recognizes the language $\langle \texttt{FIB} \rangle_2$.

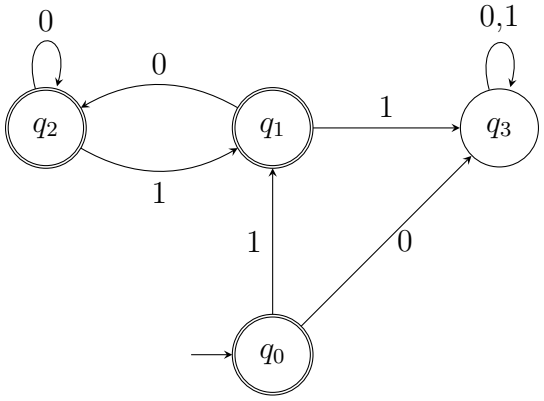


Figure 1.1: Transition diagram for the automaton that recognizes the language $\langle \texttt{FIB} \rangle_2$.

Consider the automaton

$$M = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, q_0, \{q_0, q_1, q_2\}),$$

where $\delta$ is given by Table 1.1. Every non-empty word in $\Sigma_2^*$ that starts with 1 and does not contain the factor 11 is accepted by $M$. As well, M accepts the empty word, which is necessary as $\langle 0 \rangle_2 = \epsilon$ does not contain the factor 11. Therefore, $L(M) = \langle \texttt{FIB} \rangle_2$ and $\langle \texttt{FIB} \rangle_2$ is regular.

We also define a generalization of deterministic finite automata, *nondeterministic finite automata*.

**Definition 3.** A nondeterministic finite automaton is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q$ is a set of states,
- $\Sigma$ is the input alphabet,
- $q_0 \in Q$ is the initial state, and

- $F \subseteq Q$ are the accepting states,

definitions that are identical to Definition 1. The notable change is that we define the transition function $\delta$ as

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$$

where $2^Q$ denotes the power set of $Q$.

We say that a word $x = x_0 x_1 \cdots x_i$ over $\Sigma$ is *accepted* by a nondeterministic finite automata if there is a sequence $y = y_0 y_1 \cdots y_j$ over $\Sigma \cup \{\epsilon\}$ and a sequence of states $s_0, s_1, \ldots, s_{j+1}$ such that $x = y$ and

- $s_0 = q_0$,
- $s_{\ell+1} \in \delta(s_\ell, y_\ell)$ for $0 \leq \ell \leq j$, and
- $s_{j+1} \in F$.

Otherwise, we say that $x$ is *rejected*. For a nondeterministic finite automata $M$, we define $L(M)$ as the set of words accepted by $M$. Nondeterminism is useful, as it can simply the construction of an automaton that recognizes a language. For example, consider the language $L$ of words over $\{0, 1\}$ that end with the factor 000 or 111. Figure 1.2 is the transition diagram for a deterministic finite automaton that recognizes $L$. We can simply the construction with nondeterminism, which gives us Figure 1.3. We note without proof some useful results on nondeterministic and deterministic finite automata [31]:

**Theorem 4.** *Let $L$ be a language.*

1. *Every nondeterministic finite automaton has an equivalent deterministic finite automaton.*
2. *The language $L$ is regular if and only if there exists a nondeterministic finite automaton that recognizes $L$.*
3. *If $L$ is recognized by a nondeterministic finite automaton with $n$ states, then there exists a deterministic finite automaton with at most $2^n$ states that recognizes $L$.*

Lastly, we briefly define a few terms to clarify our discussions of computability. A *decision problem* is a problem where we are given a problem instance of a fixed form and we want to answer some yes or no question about it. We can encode problem instances as

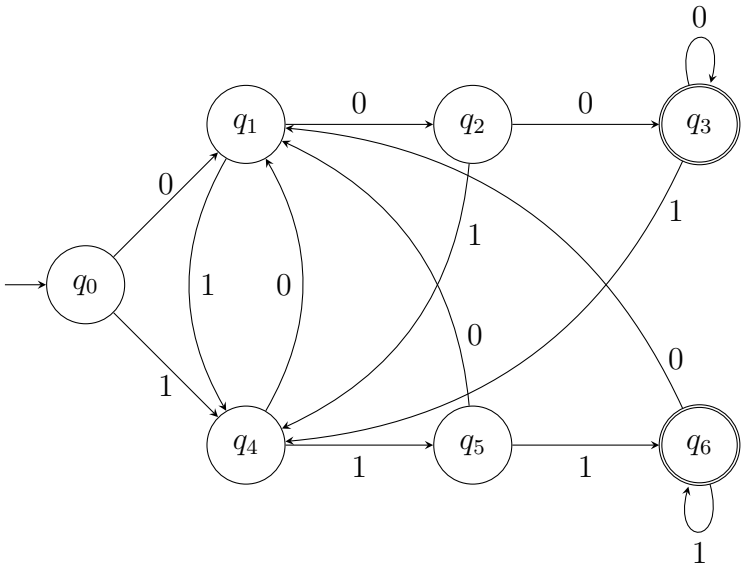


Figure 1.2: Transition diagram for a deterministic finite automaton that recognizes the language of words that end with factor 000 or 111.

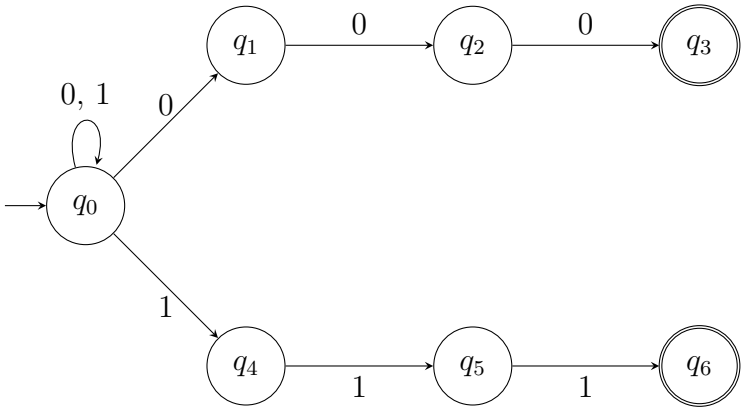


Figure 1.3: Transition diagram for a nondeterministic finite automaton that recognizes the language of words that end with factor 000 or 111.

strings and view a decision problem as a language where a specific string is included in the language if and only if the answer to the decision problem question is yes for the problem instance encoded in the string. For example, consider the following decision problem:

`IS FIBBINARY`
*Instance:* A natural number $n$.
*Question:* Does the binary representation of $n$ have repeated 1s?

If we encode a problem instance as the binary representation of the natural number $n$, the language associated with `IS FIBBINARY` is the language $\langle \texttt{FIB} \rangle_2$. However, we often discuss decision problems instead of languages as reasoning about mathematical objects instead of a complicated string encoding can be significantly easier.

A decision problem is called *decidable* if there exists an algorithm that answers the decision problem question correctly for every problem instance. A decision problem is called *Turing-recognizable* if there exists an algorithm that answers the decision problem question correctly for every problem instance where the answer is "yes". If the answer is "no" for the given problem instance, the algorithm can either answer "no", or fail to come to an answer. These two definitions also apply to languages as we can easily interpret a language as a decision problem with problem instance "A string $x$" and question "$x \in L$?" Similarly, we can describe sets of natural numbers as decidable or Turing-recognizable by viewing inclusion in the set as a decision problem analogously to languages. We typically formalize these algorithms as *Turing Machines*, but we omit a formal definition here, as it is not a primary component of this thesis.

## 1.2 Ratio Sets and Previous Work

Let $\mathbb{Q}^+$ denote the non-negative rationals. Let $\mathbb{N} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \ldots\}$ denote the natural numbers. Let $A, B \subseteq \mathbb{N}$ be two subsets of the natural numbers. Define the *ratio set*

$$R(A, B) = \left\{ \frac{a}{b} \;\middle|\; a \in A, b \in B, b \neq 0 \right\} \subseteq \mathbb{Q}^+$$

and define $R(A) = R(A, A)$. The ratio set $R(A, B)$ is sometimes known as a *quotient set* or denoted by $A/B$ in the literature [1, 3, 4, 6, 9, 12–14, 16, 22, 23, 29, 30]. When the ratio set $R(A, B)$ we are discussing is clear, we denote the smallest $a$ and $b$ such that $a/b = p/q$, $a \in A$, and $b \in B$ for a given $p/q$ as $\underline{a}(p/q)$ and $\underline{b}(p/q)$ respectively.

A preliminary question regarding ratio sets is the cardinality of $R(A)$ for a finite set $A$. Many results on the cardinality of finite ratio sets resulted from work extending previous results regarding

- *sum sets* $A + A = \{a_1 + a_2 \mid a_1, a_2 \in A\}$,
- *difference sets* $A - A = \{a_1 - a_2 \mid a_1, a_2 \in A\}$, and
- *product sets* $A \cdot A = \{a_1 \cdot a_2 \mid a_1, a_2 \in A\}$.

Chu extended previous work examining finite sets $A$ where $|A + A| > |A - A|$ to prove that the proportion of subsets $A$ of $\{1, 2, \cdots n\}$ where $|A \cdot A| > |R(A)|$ approaches 0 when $n \to \infty$ [6]. Roche-Newton [25] and Shkredov [30] examined how ratio sets can be used to probe the Erdős-Szemerédi conjecture [10]. Cilleruelo and Guijarro-Ordóñez [7], as well as Sanna [28], investigated the cardinality of $R(A)$ when $A$ is randomly constructed according to a distribution. This thesis does not explore finite sets and instead examines ratio sets of infinite sets and explores their density in $\mathbb{R}^+$ and $\mathbb{N}$.

Let $\mathbb{R}^+$ denote the non-negative reals. We say a set $C \subseteq \mathbb{R}^+$ is *dense in* $\mathbb{R}^+$ if for every $r \in \mathbb{R}^+$ and $\epsilon > 0$ there exists $c \in C$ such that $|c - r| < \epsilon$. Let $A, B \subseteq \mathbb{N}$. We are interested in when $R(A, B)$ is dense in $\mathbb{R}^+$. When $R(A)$ is dense in $\mathbb{R}^+$ we say that $A$ is $(R)$*-dense*. For example, if $A = \mathbb{N}$ then $R(A) = \mathbb{Q}^+$, which is dense in $\mathbb{R}^+$ so we have that $\mathbb{N}$ is $(R)$-dense. Several more sparse subsets of $\mathbb{N}$ were proven to be $(R)$-dense, such as $P_\alpha = \{p_i^\alpha \mid p_i \in P\}$ for natural numbers $\alpha > 0$ where $P = \{p_1 = 2, p_2 = 3, p_3 = 5, \ldots\}$ is the set of primes [35]. More generally, $\{p_i^{\alpha_i} \mid i \geq 1\}$ is $(R)$-dense when $\alpha_i = O(i^{\frac{3}{8}})$ [5].

We say that $A \subseteq \mathbb{N}$ has the property $(S)$, so named for Steinhaus, if for each $\alpha \in \mathbb{R}^+$ there exists sequence $a_1, a_2, \ldots$ such that $a_i \leq a_{i+1}$, $a_i \in A$, and

$$\lim_{i \to \infty} \frac{a_i}{i} = \alpha.$$

Some useful sufficient conditions for $(R)$-density derived from the property $(S)$ are due to Narkiewicz and Šalát [24]:

**Theorem 5.** *Let* $A = \{a_1 < a_2 < \cdots\} \subseteq \mathbb{N}$.

1. *If* $A$ *has the property* $(S)$*, then* $A$ *is* $(R)$*-dense.*
2. $A$ *has the property* $(S)$ *if and only if*

$$\lim_{n \to \infty} \frac{a_{n+1}}{a_n} = 1.$$

Šalát then later proved the following theorem [27]:

**Theorem 6.** *A set $A \subseteq \mathbb{N}$ has the property $(S)$ if and only if $R(A, B)$ is dense in $\mathbb{R}^+$ for every infinite set $B \subseteq \mathbb{N}$.*

Another focus of this thesis is the relationship between $(R)$-density and a similar concept, *asymptotic density*. Given a subset $A \subseteq \mathbb{N}$, let $A(n) = |\{a \leq n \mid a \in A\}|$. We first define *upper asymptotic density*

$$\overline{d}(A) = \limsup_{n\to\infty} \frac{A(n)}{n},$$

and *lower asymptotic density*

$$\underline{d}(A) = \liminf_{n\to\infty} \frac{A(n)}{n}.$$

If $\overline{d}(A) = \underline{d}(A)$ we call the value $d(A) = \overline{d}(A) = \underline{d}(A)$ the asymptotic density. Equivalently, we can define $d(A)$ as

$$d(A) = \lim_{n\to\infty} \frac{A(n)}{n}$$

when the limit exists. Asymptotic density gives a useful criterion for $(R)$-density, as was first proved by Šalát [26, Thm. 4]:

**Theorem 7.** *For every $A \subseteq \mathbb{N}$, if the asymptotic density $d(A) > 0$ then $A$ is $(R)$-dense.*

Šalát's bounds [26, Thm. 1] were later improved by Strauch and Tóth [33, Thm. 1] to the following:

**Theorem 8.** *For every $A \subseteq \mathbb{N}$, if the lower asymptotic density $\underline{d}(A) \geq \frac{1}{2}$ then $A$ is $(R)$-dense. Conversely, if $0 \leq \gamma < \frac{1}{2}$ then there exists $A \subset \mathbb{N}$ such that $\underline{d}(A) = \gamma$ and $A$ is not $(R)$-dense.*

Let $A = \{a_1, a_2, \ldots\} \subseteq \mathbb{N}$ where $a_i < a_{i+1}$ for $i \geq 0$. Consider the sequence

$$\frac{a_1}{a_1}, \frac{a_1}{a_2}, \frac{a_2}{a_2}, \frac{a_1}{a_3}, \frac{a_2}{a_3}, \frac{a_3}{a_3}, \ldots, \frac{a_1}{a_i}, \frac{a_2}{a_i}, \ldots, \frac{a_i}{a_i}, \ldots$$

which we can split into *blocks* $A_1, A_2, \ldots$ where

$$A_i = \left(\frac{a_1}{a_i}, \frac{a_2}{a_i}, \ldots, \frac{a_i}{a_i}\right).$$

We call $A_1, A_2, \ldots$ the *block sequence*. We also define the *dispersion* of a block

$$D(A_i) = \max\left\{\frac{a_1}{a_i}, \frac{a_2 - a_1}{a_i}, \frac{a_3 - a_2}{a_i}, \ldots, \frac{a_i - a_{i-1}}{a_i}\right\}$$

which is the largest distance between the terms of a block. The dispersion of $A$ is defined as

$$\underline{D}(A) = \liminf_{i\to\infty} D(A_i).$$

Bukor and Csiba provided a few bounds on $\underline{D}(A)$ and some methods for estimating the dispersion of a given set. More relevant to our purposes, Filip, Mišík, and Tóth worked on relating dispersion to $(R)$-density.

**Theorem 9** ([34])**.** *Let $A \subseteq \mathbb{N}$.*

1. *If $\underline{D}(A) = 0$, then $A$ is $(R)$-dense.*
2. *If $A$ is $(R)$-dense, then $\underline{D}(A) \leq \frac{1}{2}$.*

The same three authors worked to relate dispersion to asymptotic density.

**Theorem 10** ([11])**.** *Let $A \subseteq \mathbb{N}$.*

1. *If $d(A) > 0$, then $\underline{D}(A) = 0$.*
2. *If $\underline{d}(A) < \overline{d}(A)$, then*

$$\underline{D}(A) \leq \frac{(1-\beta)(1-\alpha)}{\beta(1-\alpha)}$$

*where $\alpha = \underline{d}(A)$ and $\beta = \overline{d}(A)$.*

Finally, we briefly mention some previous work on ratio sets that is more tangential to this thesis. Mišík has extended the asymptotic density conditions for $(R)$-density [33] to logarithmic density [20] and then later derived more unique conditions for $(R)$-density in terms of logarithmic density along with Tóth [21]. There is a variety of work exploring the density of ratio sets in the context of $p$-adic numbers [1, 9, 12, 13, 22, 23, 29]. Ratio sets have also been generalized to higher dimensions through $k$-directions sets, which were explored by Leonetti and Sanna [18] and Antony et al. [2].

## 1.3 Goals of this Thesis

We use many of the theorems presented in Section 1.2 to better understand the ratio sets $R(A, B)$ that we examine in Chapter 2 and Chapter 3. However, these theorems are not powerful enough to completely characterize a specific ratio set, especially when $A \neq B$. Consider the odd natural numbers $A = \{2i + 1 \mid i \in \mathbb{N}\} \subset \mathbb{N}$. We know that $d(A) = \frac{1}{2}$ so Theorem 7 gives us that $R(A)$ is $(R)$-dense. Since $A$ is $(R)$-dense, Theorem 9 tells us that $\underline{D}(A) \leq \frac{1}{2}$. However, a brief inspection of $R(A)$ reveals that $R(A) \cap \mathbb{N} = A$, which is a simple but notable result not immediately obvious from the density results.

When presented with sets $A, B \subseteq \mathbb{N}$, the main questions we ask in this thesis are:

1. What elements are in $R(A, B)$?
2. How precisely can we describe the elements of $R(A, B)$?
3. If an element is in $R(A, B)$, how many ways can they be represented as $a/b$ for $a \in A$ and $b \in B$?
4. How large is the smallest representation $a/b$ for a given element of $R(A, B)$ and how efficiently can we find it?
5. How well can we approximate an $r \in \mathbb{R}^+$ with elements from $R(A, B)$?

In short, we want to characterize $R(A, B)$ with the most simplified, efficient, and comprehensive description possible. We attack this problem from a computational perspective, as opposed to the largely mathematical analysis in the literature. The first step towards a complete description is to attempt to answer the following decision problem:

`RECOGNIZABLE RATIO SET MEMBERSHIP`
*Instance:* Two Turing-recognizable sets $A, B \subseteq \mathbb{N}$ encoded as Turing machines $T_A$ and $T_B$ that enumerate $A$ and $B$ respectively, and non-negative rational $p/q \in Q^+$.[1]
*Question:* Is $p/q \in R(A, B)$?

A naive approach to try to answer `RECOGNIZABLE RATIO SET MEMBERSHIP` is to systematically generate the entire ratio set by brute force. We can prove that `RECOGNIZABLE`

[1] A set is Turing-recognizable if and only if there is a Turing machine that enumerates it [Thm. 3.21 31].

RATIO SET MEMBERSHIP is Turing-recognizable by enumerating $R(A, B)$. We accomplish this by simulating $T_A$ and $T_B$ to get some enumeration of $A = \{a_1, a_2, \ldots\}$ and $B = \{b_1, b_2, \ldots\}$. Whenever we get a new $a_i$ and $b_i$, we continue the enumeration of the sequence

$$c_{1,1}, c_{1,2}, c_{2,1}, c_{2,2}, c_{1,3}, c_{2,3}, c_{3,1}, c_{3,2}, c_{3,3}, \ldots, c_{1,i}, c_{2,i}, \ldots, c_{i-1,i}, c_{i,1}, c_{i,2}, \ldots, c_{i,i}, \ldots \tag{1.1}$$

where $c_{i,j} = \frac{a_i}{b_j}$. We note that this is not necessarily the same as the block sequence, because the $a_i$ and $b_i$ are not guaranteed to be enumerated by $T_A$ and $T_B$ in ascending order. When enumerating, we skip $c_{i,j}$ if $b_j = 0$ and we also skip any $c_{i,j}$ that as already appeared previously in the enumeration. This enumerated sequence is precisely $R(A, B)$, which means that RECOGNIZABLE RATIO SET MEMBERSHIP is Turing-recognizable.

To attempt to decide if a specific $p/q \in R(A, B)$ one could simulate the enumeration of (1.1) and check if each entry is equal to $p/q$. If $p/q \in R(A, B)$, then $p/q$ appears somewhere in the enumeration and the algorithm eventually terminates. However, if $p/q \notin R(A, B)$, then $p/q$ is not in the sequence and this algorithm runs indefinitely. In other words, this algorithm provides us no way to prove that $p/q \notin R(A, B)$.

Another issue is that enumerating (1.1) to try and answer RECOGNIZABLE RATIO SET MEMBERSHIP can be extremely inefficient, even if it does eventually terminate. For example, consider $n = 77$ and $R(\texttt{FIB})$. For this case, we assume that FIB is enumerated in ascending order for the purposes of computing (1.1). We have that $77 \in R(\texttt{FIB})$ since $77 \cdot 133 = 10241$ and $133, 10241 \in \texttt{FIB}$. We could have come to that result by enumerating (1.1) for $A = B = \texttt{FIB}$, but 77 first appears as the 355785th term in the enumeration. We revisit $R(\texttt{FIB})$ in greater depth in Section 2.3, but first we make a stronger, though somewhat obvious, statement about RECOGNIZABLE RATIO SET MEMBERSHIP:

**Proposition 11.** RECOGNIZABLE RATIO SET MEMBERSHIP *is not decidable.*

*Proof.* Assume for the sake of contradiction RECOGNIZABLE RATIO SET MEMBERSHIP is decidable. Let $A$ be a set that is Turing-recognizable but not decidable.[2] Clearly, $\{1\}$ is also Turing-recognizable. If RECOGNIZABLE RATIO SET MEMBERSHIP was decidable, we would be able to decide if $n/1 = n$ is contained in $R(A, \{1\}) = A$ for all $n \in \mathbb{N}$. If so, $A$ is decidable, a contradiction. □

Since RECOGNIZABLE RATIO SET MEMBERSHIP is generally intractable, this thesis instead examines some restricted versions of the problem. Chapter 2 examines the related,

[2]Sets with this property do exist [31, Thm. 4.11].

though still general, decision problem:

`REGULAR RATIO SET MEMBERSHIP`
*Instance:* A natural number $k \in \mathbb{N}$, two sets of natural numbers $A, B \subseteq \mathbb{N}$ such that $\langle A \rangle_k$ and $\langle B \rangle_k$ are regular languages, and non-negative rational $p/q \in \mathbb{Q}^+$.
*Question:* Is $p/q \in R(A, B)$?

We occasionally call a ratio set $R(A, B)$ where $A$ and $B$ satisfy the condition in `REGULAR RATIO SET MEMBERSHIP` a *regular ratio set*. We expect that $A$ and $B$ are encoded as automata $M_A$ and $M_B$ such that $[L(M_A)]_k = A$ and $[L(M_B)]_k = B$. Section 2.1 builds up the decision procedure for `REGULAR RATIO SET MEMBERSHIP` that is the primary tool of analysis for the remainder of the results in Chapter 2. Chapter 3 presents several specialized algorithms built for deciding inclusion in $R(A, B)$ when $A$ and $B$ are sets of natural numbers that have representations that are palindromes or antipalindromes in a given base. The language of palindromes and the language of antipalindromes are both not regular, so they are beyond the scope of the algorithms in Chapter 2. However, the concepts introduced in Chapter 2 still inform our approach to the more general problems of Chapter 3.

# Chapter 2

# Ratio Sets of Regular Languages

## 2.1 A Decision Procedure Based on Deterministic Finite Automata

Our goal in this section is to prove the following result:

**Theorem 12.** `REGULAR RATIO SET MEMBERSHIP` *is decidable in* $O(k^2(p+q) \cdot |M_A| \cdot |M_B|)$ *time, for the given instance of* `REGULAR RATIO SET MEMBERSHIP`*:* $k \in \mathbb{N}$*,* $p/q \in \mathbb{Q}^+$*, and* $A, B \subseteq \mathbb{N}$ *encoded as automata* $M_A$ *and* $M_B$ *such that* $[L(M_A)]_k = A$ *and* $[L(M_B)]_k = B$*.*

We decide `REGULAR RATIO SET MEMBERSHIP` by building an automaton that accepts the representation $\langle a, b \rangle_k$ of all pairs $(a, b)$ where $a \in A$, $b \in B$, and $p/q = a/b$ for a given $p/q \in Q^+$ and then examining what, if any, inputs are accepted. When we constrain this general algorithm to a specific problem, we fix a specific $k$ and have a constant-sized automata description of $A$ and $B$. This results in the complexity to answer a rational $p/q \in R(A, B)$ for a regular ratio set $R(A, B)$ to be $O(p + q)$ time, or in the case of a natural number $n \in R(A, B)$, linear time $O(n)$. We begin by building simple automata that answer parts of `REGULAR RATIO SET MEMBERSHIP` and then present the full automata and proof of Theorem 12 in Subsection 2.1.5.

### 2.1.1 Finite-State Transducers

The first step on the path to deciding `REGULAR RATIO SET MEMBERSHIP` is to build automata that can multiply natural numbers. A crucial insight is that $n = a/b$ if and only

if $b \cdot n = a$. Hence, we can avoid computing quotients by reinterpreting the problem as multiplication. In order to build intuition for the more complicated automata later in the section, we briefly introduce another extension to deterministic finite automata: *finite-state transducers*.

**Definition 13.** A finite-state transducer, or simply a *transducer*, is a 5-tuple $M = (Q, \Sigma, \delta, q_0, \Delta, \lambda)$, where

- $Q$ is a set of states,
- $\Sigma$ is the input alphabet,
- $\delta : Q \times \Sigma \to Q$ is a transition function defined as usual, and
- $q_0 \in Q$ is the initial state,

definitions that are identical to Definition 1. Instead of a set of accepting states $F$, we have an *output alphabet* $\Delta$ and an *output function* $\lambda : Q \times \Sigma \to \Delta$. Instead of accepting or rejecting an input $x = x_0 x_1 \cdots x_i$, a finite-state transducer produces an output $y_0 y_1 \cdots y_i$, where $y_j = \lambda(\delta(q_0, x[0..j-1]), x_j)$. In other words, at each step of the input, a finite-state transducer adds a symbol to the output in addition to the usual transition. We can draw a finite-state transducer in a similar transition diagram to a deterministic finite automaton where we label the transition from $q$ to $\delta(q, \sigma)$ with $\sigma/\lambda(q, \sigma)$ for the appropriate $q \in Q$ and $\sigma \in \Sigma$. An example of such a drawing is given in Figure 2.1.

**Example 14.** Let $A = \{1, 5, 9, 17, 21, 37, 41, 65, 69, 73, \ldots\}$ be the odd elements of FIB. A minor fact is that $A = \{4n + 1 \mid n \in \texttt{FIB}\}$. We can build a finite-state transducer to convert between $\langle \texttt{FIB} \rangle_2$ and $\langle A \rangle_2$. Consider the finite-state transducer

$$M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{0, 1\}, \lambda),$$

where $\delta$ and $\lambda$ are given by Table 2.1. The transducer $M$ requires that the input $n \in \texttt{FIB}$ is given in least-significant-digit-first binary representation. As well, the output is typically longer than the input. Take for example converting $4 \mapsto 17$, $|\langle 4 \rangle_2| = |100| = 3$ but $|\langle 17 \rangle_2| = |10001| = 5$. In fact, all output words are two symbols longer than the input, so for natural number $n$ we pad the most-significant-digit-first binary representation of $n$ with two zeroes at the front and then reverse the string so taken together we input to the transducer $(00\langle n \rangle_2)^R$. Returning to the example $4 \mapsto 17$, we input into $M$ the word $(00\langle 4 \rangle_2)^R = (00100)^R = 00100$. The list of states we pass through is

$$q_0 \mapsto q_3 \mapsto q_3 \mapsto q_1 \mapsto q_0 \mapsto q_3,$$

| $q$ | $\delta(q,0)$ | $\delta(q,1)$ | $\lambda(q,0)$ | $\lambda(q,1)$ |
|---|---|---|---|---|
| $q_0$ | $q_3$ | $q_1$ | 1 | 1 |
| $q_1$ | $q_0$ | $q_2$ | 0 | 0 |
| $q_2$ | $q_0$ | $q_2$ | 1 | 1 |
| $q_3$ | $q_3$ | $q_1$ | 0 | 0 |

Table 2.1: Transition table of $\delta$ and $\lambda$ for the finite-state transducer between `FIB` and the odd natural numbers in `FIB`.

which results in output word

$$\lambda(q_0,0)\lambda(q_3,0)\lambda(q_3,1)\lambda(q_1,0)\lambda(q_0,0) = 10001 = (10001)^R = (\langle 17\rangle_2)^R.$$

### 2.1.2 Multiplication Transducers

Let us examine the calculation $7625 \cdot 38$ using the typical long multiplication algorithm. We view this calculation as the "input" 7625 to the algorithm "multiplication by 38" resulting in the "output" $38 \cdot 7625 = 289750$. In this case, we call 7625 the *multiplicand* and 38 the *multiplier*. Recall the typical long multiplication algorithm where we view $38 \cdot 7625$ as

$$7 \cdot 10^3 \cdot 38 + 6 \cdot 10^2 \cdot 38 + 2 \cdot 10^1 \cdot 38 + 5 \cdot 10^0 \cdot 38 \tag{2.1}$$

We often express Equation (2.1) graphically; we supply such a drawing for $7625 \cdot 38$ in Figure 2.2a. There are four important observations we take from this process:

*Observation* 15.

1. This algorithm functions identically in other bases. We simply demonstrated the algorithm in base 10 for the simplicity of explanation.
2. In each step, we use exactly one digit from the input to compute the value we add to the partial sum.
3. After step $n$, the $n$ least significant digits of the partial sum are fixed for the remainder of the algorithm and they correspond to the $n$ least significant digits of the final result.
4. While the final step fixes multiple digits of the final output, we can reinterpret this as multiple implicit steps where we add $10^i \cdot 0 \cdot 38 = 0$ to the partial sum and fix a single additional digit.

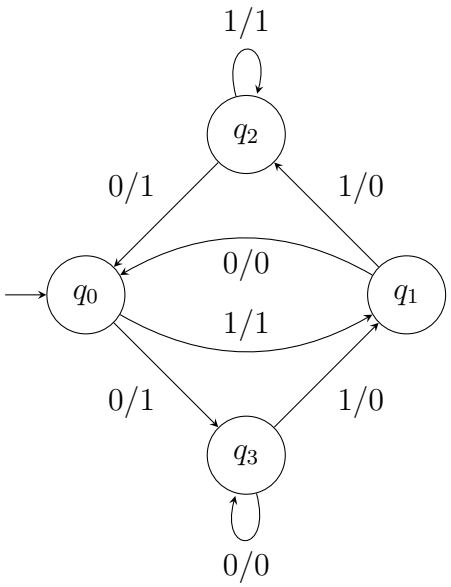


Figure 2.1: Transition diagram for the finite-state transducer between `FIB` and the odd natural numbers in `FIB`.

We call the component of the partial sum that is not fixed at each step the *carry*. We can extend the multiplication algorithm by expressing each step in terms of the carry and new symbol and by explicitly including the implicit steps described in Observation 15.4. We call this the *extended long multiplication algorithm* and we can formally define it as a series of equations.

**Definition 16.**

- Let $k \geq 2$ be the base wherein we implement the algorithm.
- Let $n \geq 1$ be the multiplier.
- Let $a = [a_i a_{i-1} \cdots a_0]_k$ and $b = [b_i b_{i-1} \cdots b_0]_k$ such that $n \cdot b = a$. (We typically assume that the smaller of $a$ or $b$ is padded with zeroes while the longer one starts with a non-zero character.)
- Let $c_0, c_1, \ldots, c_{i+1} \in \mathbb{N}$ be the carries, where $c_j$ is the carry after step $j$ of the algorithm.

$$
\begin{array}{rr}
 & 7\,6\,2\,5 \\
\times & 3\,8 \\
\hline
 & 0 \\
+ & 1\,9\,0 \quad (= 10^0 \cdot 5 \cdot 38) \\
\hline
 & 1\,9\,0 \\
+ & 7\,6\,0 \quad (= 10^1 \cdot 2 \cdot 38) \\
\hline
 & 9\,5\,0 \\
+ & 2\,2\,8\,0\,0 \quad (= 10^2 \cdot 6 \cdot 38) \\
\hline
 & 2\,3\,7\,5\,0 \\
+ & 2\,6\,6\,0\,0\,0 \quad (= 10^3 \cdot 7 \cdot 38) \\
\hline
 & 2\,8\,9\,7\,5\,0
\end{array}
$$

(a) Long multiplication with partial sums.

$$
\begin{array}{rr}
 & 0\,0\,7\,6\,2\,5 \\
\times & 3\,8 \\
\hline\hline
 & 0 \quad \text{(first carry, } c_0\text{)} \\
+ & 1\,9\,0 \quad (= 5 \cdot 38) \\
\hline
 & \mathbf{1\,9}\,\mathit{0} \\
\hline\hline
 & 1\,9 \quad \text{(second carry, } c_1\text{)} \\
+ & 7\,6 \quad (= 2 \cdot 38) \\
\hline
 & \mathbf{9}\,\mathit{5} \\
\hline\hline
 & 9 \quad \text{(third carry, } c_2\text{)} \\
+ & 2\,2\,8 \quad (= 6 \cdot 38) \\
\hline
 & \mathbf{2\,3}\,\mathit{7} \\
\hline\hline
 & 2\,3 \quad \text{(fourth carry, } c_3\text{)} \\
+ & 2\,6\,6 \quad (= 7 \cdot 38) \\
\hline
 & \mathbf{2\,8}\,\mathit{9} \\
\hline\hline
 & 2\,8 \quad \text{(fifth carry, } c_4\text{)} \\
+ & 0\,0 \quad (= 0 \cdot 38) \\
\hline
 & \mathbf{2}\,\mathit{8} \\
\hline\hline
 & 2 \quad \text{(sixth carry, } c_5\text{)} \\
+ & 0 \quad (= 0 \cdot 38) \\
\hline
 & \mathbf{0}\,\mathit{2} \\
\hline\hline
 & 2\,8\,9\,7\,5\,0
\end{array}
$$

(b) Extended long multiplication algorithm.

Figure 2.2: Long multiplication of $7625 \cdot 38$.

Step $j$ of the extended long multiplication algorithm is the equation

$$n \cdot b_j + c_j = c_{j+1} \cdot k + a_j, \tag{2.2}$$

where $1 \leq j \leq i$. We call Equation (2.2) the *carry equation*. We note that $a_j$ and $c_{j+1}$ are defined such that

$$a_j \equiv n \cdot b_j + c_j \pmod{k} \tag{2.3}$$

and $0 \leq a_j \leq k - 1$. We also say that the carry equations starting from $c_0$ and with the index of $c_j$ increasing are the *least significant digit carry equations*. We sometimes refer to the unchecked carry equation calculating $c_j$ of smallest index $j$ as the *least significant digit carry equation*. Similarly, the carry equations starting from $c_{i+1}$ and with the index of $c_j$ decreasing are the *most significant digit carry equations*. We sometimes refer to the unchecked carry equation calculating $c_j$ of largest index $j$ as the *least significant digit carry equation*. We do not define a specific subset of the carry equations as the most significant digit or least significant digit carry equations. Instead, we use this terminology to aid our discussion when we describe verifying the series of equations from a specific direction, either upwards or downwards in terms of index.

Figure 2.2b presents the extended long multiplication algorithm applied to $7625 \cdot 38$ with the carry $c_j$ after each step in bold and the new digit $a_j$ of the output in italics. To keep the series of equations general, we include $c_0 = 0$ for the first step and we note that the last step results in an unused carry $c_{i+1} = 0$. The following property of the carries is important to the construction of our transducers and automata:

**Lemma 17.** *Each carry* $0 \leq c_j < n$ *for* $0 \leq j \leq i + 1$.

*Proof.* We prove the lemma by induction on $j$. By definition we have $c_0 = 0$ so $0 \leq c_0 < n$, which forms the base case for our induction.

We then assume for the sake of induction that $0 \leq c_j < n$. Step $j$ of the extended long multiplication algorithm is

$$n \cdot b_j + c_j = c_{j+1} \cdot k + a_j.$$

We can rewrite this as

$$\begin{aligned} c_{j+1} &= \frac{n \cdot b_j + c_j - a_j}{k} \\ &\leq \frac{n \cdot (k-1) + c_j - 0}{k} \\ &< \frac{n \cdot (k-1) + n}{k} \\ &< \frac{n \cdot (k-1+1)}{k} \\ &< \frac{n \cdot k}{k} \\ &< n, \end{aligned}$$

as required.

As well, we have that $n \cdot b_j + c_j \geq 0$ so if $c_{j+1} < 0$, then $a_j \geq (-c_{j+1}) \cdot k \geq k$, which is a contradiction. □

Using the carry equation, we can review what is required during one step of the extended long multiplication algorithm. We use the carry $c_j$ and symbol $b_j$ along with the given $n$ to compute a new carry $c_{j+1}$ and a single symbol $a_j$. A process that takes as input a single symbol and uses some constant quantity of saved information to output another single symbol is a process that can be cleanly modeled with a finite-state transducer. This leads us to the definition of a *multiplication transducer*.

**Definition 18.** Given natural numbers $k, n \in \mathbb{N}$, the base-$k$ multiplication transducer for $n$ is a finite-state transducer

$$M = (\{q_0, q_1, \ldots, q_{n-1}\}, \{0, 1, \ldots, k-1\}, \delta, q_0, \{0, 1, \ldots, k-1\}, \lambda),$$

where

$$\lambda(q_i, j) \equiv j \cdot n + i \pmod{k}$$

and $\delta(q_i, j) = q_{i'}$ for

$$i' = \frac{j \cdot n + i - \lambda(q_i, j)}{k}.$$

The transducer $M$ implements the extended long multiplication algorithm using $\delta$ and $\lambda$ to calculate the carry equation after each input and using the current state to store the carry after each step. It starts in $q_0$ as the first carry $c_0 = 0$. The function $\lambda$ calculates the

next output symbol according to Equation (2.3). Using the result from $\lambda$ along with the input and current carry from the current state, the function $\delta$ calculates the next carry.

One important note is that since the extended long multiplication algorithm starts from the least significant digit, the multiplication transducer calculates $n \cdot b$ for $b = [b_\ell b_{\ell-1} \cdots b_0]_k$ by computing $b_0 b_1 \cdots b_\ell \mapsto (\langle n \cdot b \rangle_k)^R$. In short, multiplication transducers assume least-significant-digit-first representation. This $b_\ell b_{\ell-1} \cdots b_0$ must be suitably padded with zeroes, as the transducer $M$ outputs exactly as many symbols as it reads. The required padding is the difference in length between $\langle b \rangle_k$ and $\langle n \cdot b \rangle_k$, which is either $\lfloor \log_k n \rfloor$ or $\lceil \log_k n \rceil$ but it can depend on $b$.

**Example 19.** Figure 2.3 is a base-3 multiplication transducer for $n = 5$. Take for example, input $b = 19$. Our input is $(0^{\lceil \log_3 5 \rceil} \langle 19 \rangle_3)^R = (00201)^R = 10200$. The list of states we pass through is

$$q_0 \mapsto q_1 \mapsto q_0 \mapsto q_3 \mapsto q_1 \mapsto q_0,$$

which results in output word $21101 = (\langle 95 \rangle_3)^R$.

**Example 20.** Figure 2.1 is a base-2 multiplication transducer for $n = 4$ but modified to begin with an implicit carry of 1 since the objective of the transducer is to perform $b \mapsto 4b + 1$.

Multiplication transducers can be used directly as a tool to analyze some specific ratio sets. Let $k \geq 2$ be a natural number and $\{d_0, d_1, \ldots, d_i\} \subseteq \Sigma_k$. We define the *restricted digit sets*

$$S(k, \{d_0, d_1, \ldots, d_i\}) = \{n \mid n \in \mathbb{N}, \langle n \rangle_k \in \{d_0, d_1, \ldots, d_i\}^*\},$$

which is the natural numbers that have base-$k$ representation using digits exclusively in $\{d_0, d_1, \ldots, d_i\}$.

Let $A = S(k, \{d_0, d_1, \ldots, d_i\})$ and $B = S(k, \{c_0, c_1, \ldots, c_j\})$ be restricted digit sets. Let $M_n$ be the base-$k$ multiplication transducer for $n$ We can develop a procedure to decide $n \in R(A, B)$ by examining $M_n$. We start with $M_n$ and then delete every transition from $M_n$ where that transition involves reading a symbol that is not in $\{c_0, c_1, \ldots, c_j\}$ or outputting a symbol that is not in $\{d_0, d_1, \ldots, d_i\}$. Call the resulting transducer $M'_n$. Walks through the new transducer are still valid multiplications, but only a subset of the possible inputs are permitted. The input natural numbers that are permitted satisfy the requirements of our restricted digit sets. This means that that $n \in R(A, B)$ if and only if there is a circuit $q_0 \mapsto \ldots \mapsto q_0$ in $M'_n$. (We ignore the trivial circuit that is the self-loop $q_0 \mapsto q_0$.)

**Example 21.** Let $A = S(3, \{0, 1\})$ and $B = S(3, \{0, 2\})$. Consider the question of if $5 \in R(A, B)$. We can edit the base-3 multiplication transducer for 5 given in Figure 2.3

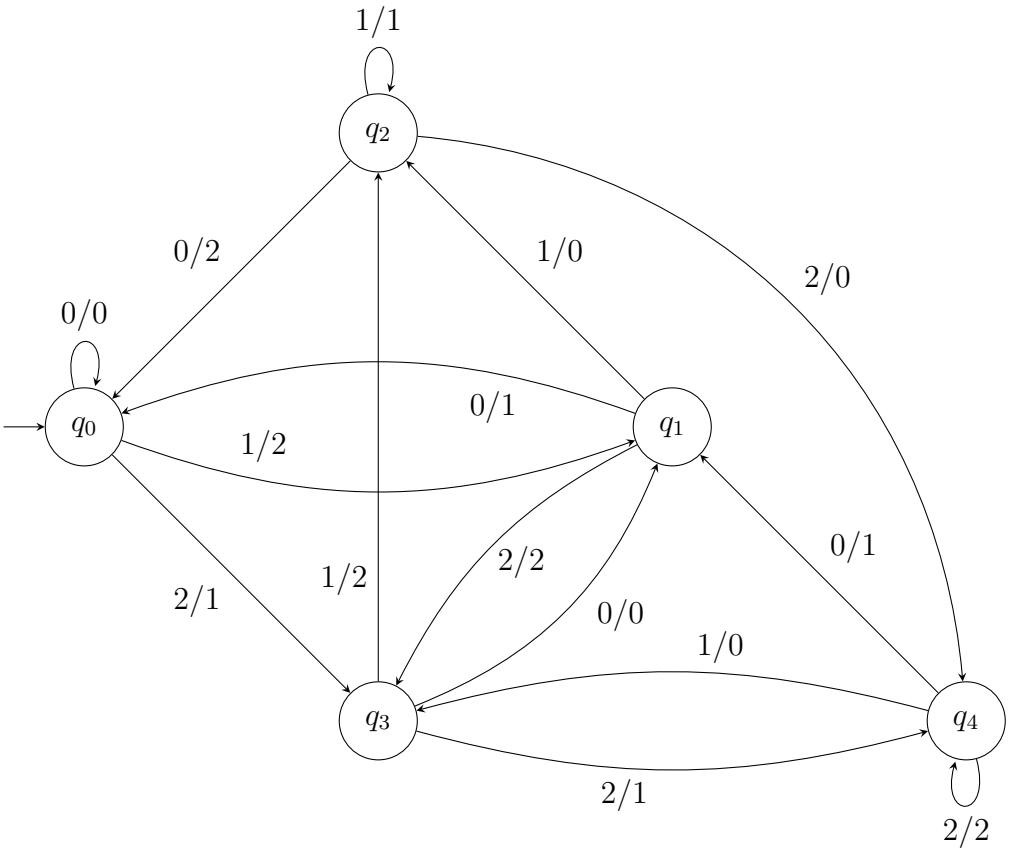


Figure 2.3: Base-3 multiplication transducer for 5.

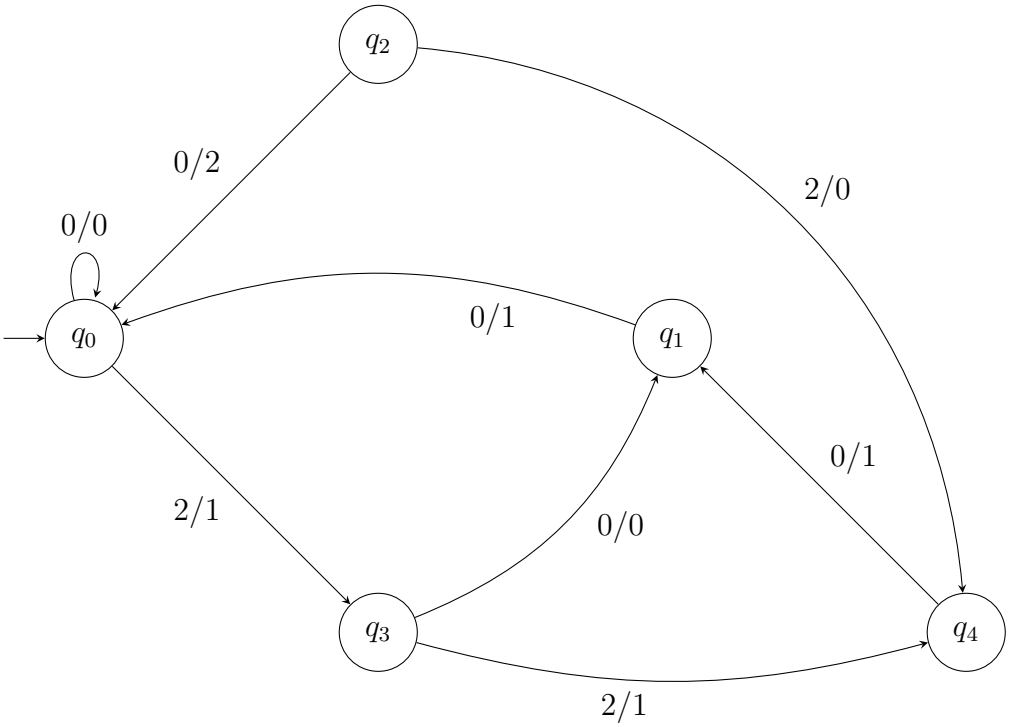


Figure 2.4: Base-3 multiplication transducer for 5 with input restricted to $S(3, \{0, 2\})$ and output restricted to $S(3, \{0, 1\})$.

by deleting every transition where we read the symbol 1 or output the symbol 2. The resulting transducer is given in Figure 2.4. We note that $q_0 \mapsto q_3 \mapsto q_1 \mapsto q_0$ is a valid walk, which corresponds to $[002]_3 \cdot 5 = [101]_3$. This gives us that $5 = \frac{10}{2}$, $10 \in A$, and $2 \in B$, so $5 \in R(A, B)$.

**Example 22.** Let $A = S(3, \{0, 1\})$ and $B = S(3, \{1, 2\}) = \mathbb{N}$. Consider the question of if $5 \in R(A, B)$. We can edit the base-3 multiplication transducer for 5 given in Figure 2.3 by deleting every transition where it reads the symbol 0 or outputs the symbol 2. The resulting transducer is given in Figure 2.5. There is no walk $q_0 \mapsto \cdots \mapsto q_0$. Therefore, $5 \notin R(A, B)$.

Sisneros-Thiry preformed a more detailed analysis of the ratio sets of restricted digit sets from a purely multiplication transducer approach [32].

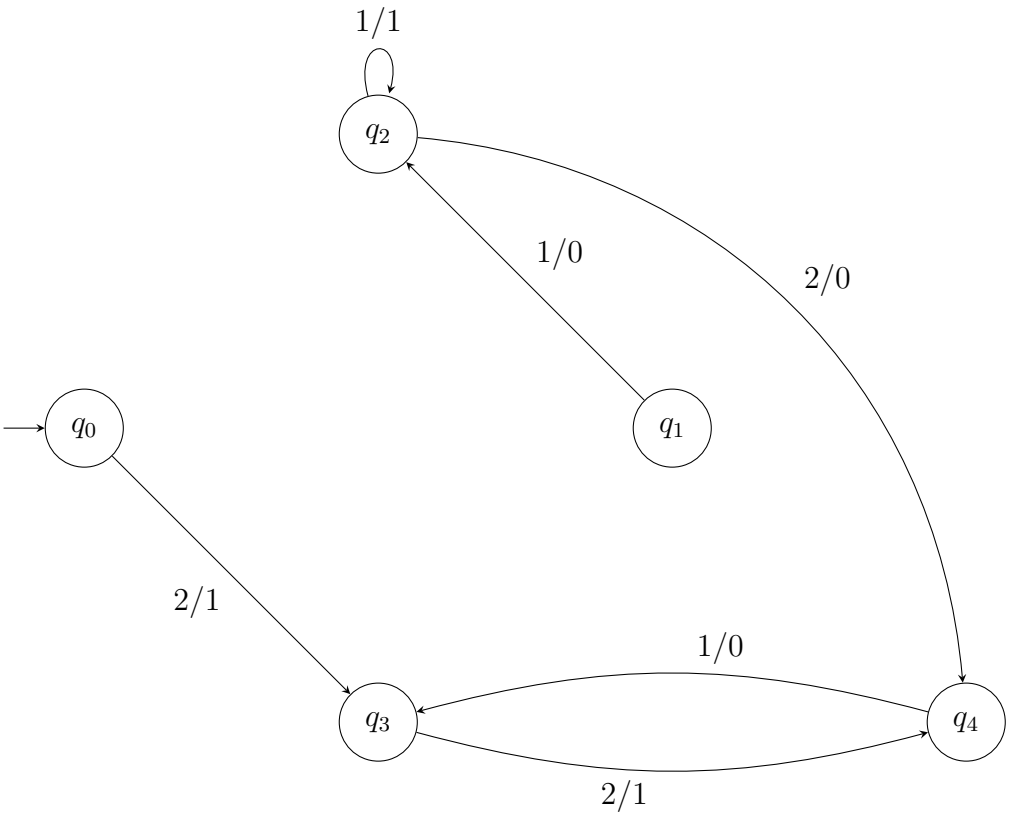


Figure 2.5: Base-3 multiplication transducer for 5 with input restricted to $S(3, \{1, 2\})$ and output restricted to $S(3, \{0, 1\})$.

### 2.1.3 Multiplication Automata

We want to expand the analysis of ratio sets to ratio sets of subsets of the natural numbers that are more complicated than what can be expressed by removing edges from transducers. Furthermore, we want to make stronger claims about specific solutions $(a, b)$ to $n \cdot b = a$. In order to do so, we build automata that emulate the functionality of multiplication transducers. Multiplication transducers implement the carry equation $n \cdot b_j + c_j = c_{j+1} \cdot k + a_j$ by first computing $a_j$ using $b_j$ and $c_j$ and then afterwards using $a_j, b_j$, and $c_j$ to compute $c_{j+1}$. The core change moving from transducers to automata is that we must read $a_j$ and $b_j$ simultaneously and verify their validity and only afterward computing the next carry. The $(a_j, b_j)$ pair is valid for a given $c_j$ if

$$a_j \equiv n \cdot b_j + c_j \pmod{k}. \tag{2.4}$$

If Equation (2.4) is satisfied, then the automaton computes $c_{j+1}$ and includes the transition to the next state. If Equation (2.4) is not satisfied, then the automaton does not include the transition with the $(a_j, b_j)$ pair.

In order to do the checks on $(a_j, b_j)$ pairs, we need to define some method of encoding $a$ and $b$ so that the correct pairs are read by the automaton at the correct time.

**Definition 23.** We define an encoding $\langle a, b \rangle_k \in (\Sigma_k \times \Sigma_k)^*$.

- Let $\langle a \rangle_k = a_i a_{i-1} \cdots a_0$ and $\langle b \rangle_k = b_j b_{j-1} \cdots b_0$.
- Let $m = \max\{i, j\}$.
- Let $a_\ell = 0$ for $i < \ell \leq m$ and $b_\ell = 0$ for $j < \ell \leq m$.

Then

$$\langle a, b \rangle_k = (a_m, b_m)(a_{m-1}, b_{m-1}) \cdots (a_0, b_0),$$

where $(a_\ell, b_\ell) \in (\Sigma_k \times \Sigma_k)$. Note that this encoding is analogous to most-significant-digit-first representation, as the first symbol $(a_m, b_m)$ contains the most significant digit of $a$ or $b$. If $a_m$ is the most significant digit of $a$, then the remaining symbol $b_m$ is either the most significant digit of $b$ or a zero padded in front of the most significant digit. Similarly, if $b_m$ is the most significant digit of $b$, then $a_m$ may be a zero padded before the most significant digit of $a$. In either case, at least one of $a_m$ and $b_m$ is non-zero.

We can build an automaton for $n \in \mathbb{N}$ that is nearly identical to the base-$k$ multiplication transducer for $n$ that can accept $(\langle a, b \rangle_k)^R$ where $b \cdot n = a$.

**Definition 24.** Given natural numbers $k, n \in \mathbb{N}$, the *base-k least-significant-digit-first multiplication automaton for n* is an automaton

$$M = (\{q_0, q_1, \ldots, q_{n-1}\}, \{0, 1, \ldots, k-1\} \times \{0, 1, \ldots, k-1\}, \delta, q_0, \{q_0\}),$$

where $\delta(q_i, (j_a, j_b)) = q_{i'}$ for

$$i' = \frac{n \cdot j_b + i - j_a}{k}$$

if and only if

$$j_a \equiv n \cdot j_b + i \pmod{k}.$$

Otherwise, $\delta(q_i, (j_a, j_b))$ is undefined. We reject any input that leads to an undefined delta.

**Example 25.** Figure 2.6 is a base-3 least-significant-digit-first multiplication automaton for $n = 5$. It is essentially identical to Figure 2.3 except for the edge labels. One accepted input is

$$(2,1)(1,0)(1,2)(0,0)(1,0) = ((1,0)(0,0)(1,2)(1,0)(2,1))^R = (\langle 95, 19 \rangle_3)^R.$$

The list of states we pass through in evaluating $(\langle 95, 19 \rangle_3)^R$ is

$$q_0 \mapsto q_1 \mapsto q_0 \mapsto q_3 \mapsto q_1 \mapsto q_0,$$

which is identical to the input $b = 19$ in Example 19.

The definition of base-$k$ least-significant-digit-first multiplication automata is useful for understanding the automata constructed for non-regular ratio sets in Chapter 3. We could use these automata as the basis for our solution to `REGULAR RATIO SET MEMBERSHIP` though one deficiency still present is that the input must still be given as least-significant-digit-first representations. This is because in this case, the extended long multiplication algorithm is evaluated from least significant digit to most significant digit. If the input were most-significant-digit-first, then a simple breath-first-search of the automata for the accepting state would find the smallest pair of natural numbers proving that the original $n$ is in the ratio set. In order to accomplish this, we reformulate the carry equation to allow us to use $c_{j+1}$ to find $c_j$.

**Definition 26.** Given natural numbers $k, n \in \mathbb{N}$, the *base-k most-significant-digit-first multiplication automaton for n* is an automaton

$$M = (\{q_0, q_1, \ldots, q_{n-1}\}, \{0, 1, \ldots, k-1\} \times \{0, 1, \ldots, k-1\}, \delta, q_0, \{q_0\}),$$

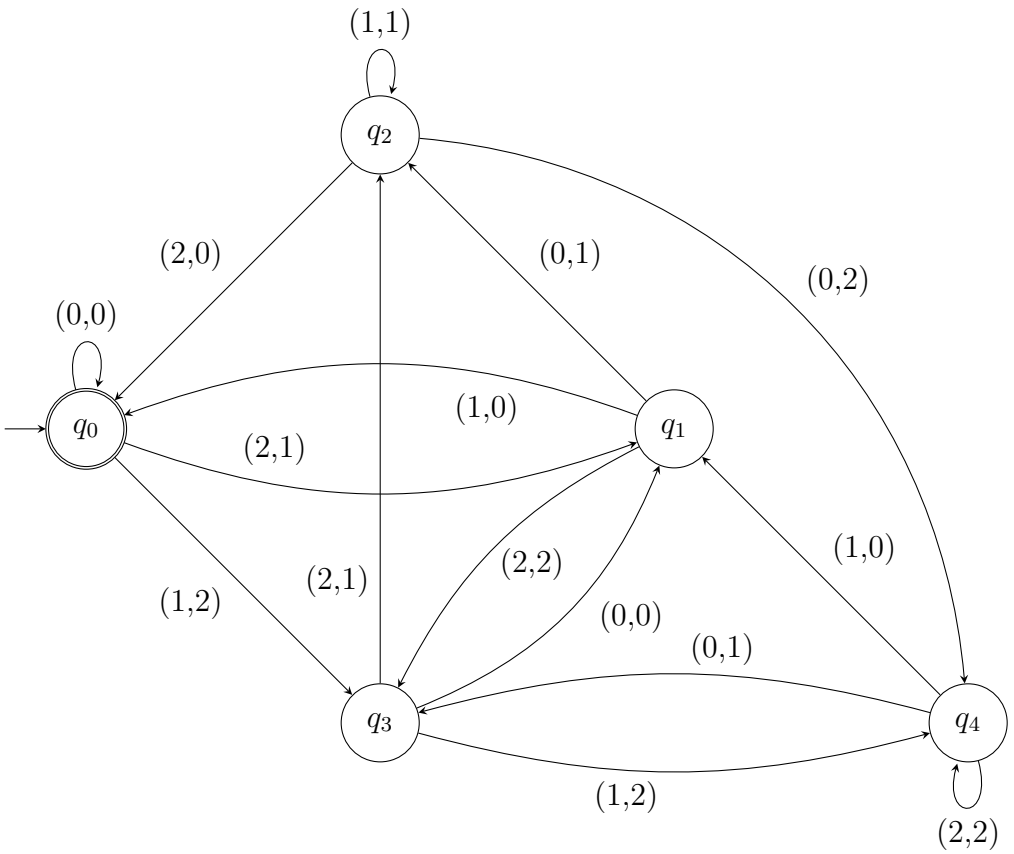


Figure 2.6: Base-3 least-significant-digit-first multiplication automaton for 5.

where $\delta(q_i, (j_a, j_b)) = q_{i'}$ for

$$i' = i \cdot k + j_a - n \cdot j_b$$

if and only if $0 \leq i' \leq n - 1$. Otherwise, $\delta(q_i, (j_a, j_b))$ is undefined. We reject any input that leads to an undefined delta.

**Example 27.** Figure 2.7 is a base-3 least-significant-digit-first multiplication automaton for $n = 5$. It is essentially identical to Figure 2.6 except for each transition being reversed. One accepted input is

$$(1, 0)(0, 0)(1, 2)(1, 0)(2, 1) = \langle 95, 19 \rangle_3.$$

The list of states we pass through in evaluating $\langle 95, 19 \rangle_3$ is

$$q_0 \mapsto q_1 \mapsto q_3 \mapsto q_0 \mapsto q_1 \mapsto q_0,$$

which is the reverse of the list for the input $(\langle 95, 19 \rangle_3)^R$ in Example 19.

We reject when $i' < 0$ or $i' \geq n$. While we are verifying the extended long multiplication algorithm in reverse, if we were to verify a multiplication as usual from the least significant digit onwards, then the carries are not smaller than zero or larger than our initial $n$. Therefore, we can enforce those constraints on the carry and keep the automaton description finite. We can also run a breadth first search starting from $q_0$ searching for $q_0$, ignoring the $(0, 0)$ loop, to get the smallest representation. In Example 27, we can find that the shortest path is $(1, 0)(2, 1) = \langle 5, 1 \rangle_3$. This automaton is what we use as the basis for the more complicated alterations.

### 2.1.4 Extension to Rational Numbers

Before integrating the verification of inclusion in a regular language into our multiplication automata, we generalize the carry equations to rationals so we can verify $p/q = a/b$ and determine if a rational is included in a regular ratio set. To do so, we need to further generalize our understanding of the extended long multiplication algorithm to find automata that implement multiplication for rational numbers. This is because a naive replacement of the multiplier in the extended long multiplication algorithm with a rational quickly violates the properties of the algorithm that we require. Figure 2.8 is the long multiplication of $147 \cdot 38/3$ using $(38/3)$ as the multiplier, which demonstrates how the long multiplication algorithm may not calculate increasing large factors of the output when the multiplier is a rational.

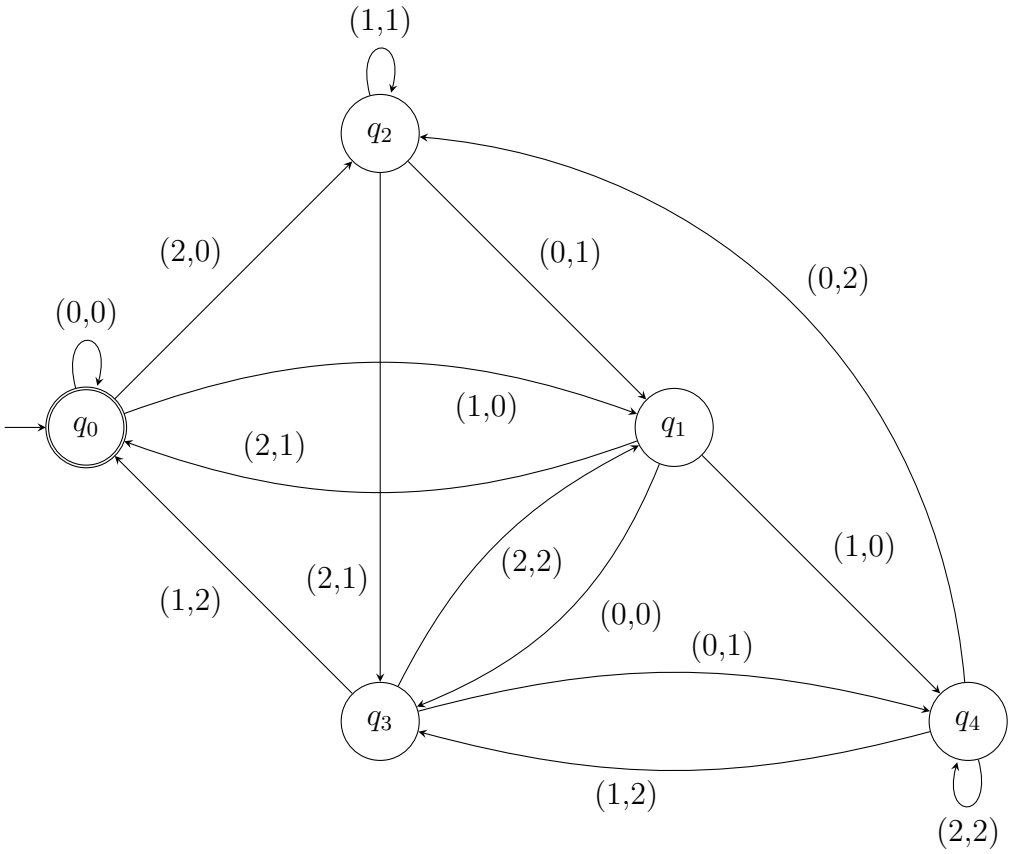


Figure 2.7: Base-3 most-significant-digit-first multiplication automaton for 5.

$$
\begin{array}{rl}
& 1\,4\,7 \\
\times & 1\,2.6\,6\cdots \\
\hline\hline
& 0 \\
+ & 8\,8.6\,6\cdots \quad (= 10^0 \cdot 7 \cdot 38/3) \\
\hline
& 8\,8.6\,6\cdots \\
+ & 5\,0\,6.6\,6\cdots \quad (= 10^1 \cdot 4 \cdot 38/3) \\
\hline
& 5\,9\,5.3\,3\cdots \\
+ & 1\,2\,6\,6.6\,6\cdots \quad (= 10^2 \cdot 1 \cdot 38/3) \\
\hline
& 1\,8\,6\,2
\end{array}
$$

Figure 2.8: Naive rational long multiplication with partial sums.

The machines we defined so far check the multiplication $b \cdot n = a$, because that equality holds if and only if $n = a/b$. Analogously, the equality $b \cdot p = a \cdot q$ holds if and only if $p/q = a/b$. The key insight is that we can simultaneously check $b \cdot p = m$ and $a \cdot q = m$ for some $m \in \mathbb{N}$, with each equation using their own set of carry equations and carries. We can accomplish this without knowing $m$. We only need to check that $b \cdot p$ and $a \cdot q$ agree on each digit. However, we end up with add more ambiguity with this type of automaton. This is because the process does not calculate an $a$ for a given $b$, only verifies that a pair $(a, b)$ is valid. For each previous automaton and transducer, at each step there was a unique $a_j$ for each $b_j$ such that $(a_j, b_j)$ lead to a valid state. This property does not persist in the automata we build for checking inclusion of rationals in a regular ratio set.

**Definition 28.** We extend the carry equations to rational numbers:

- Let $k \in \mathbb{N}$,
- let $p/q \in \mathbb{Q}^+$,
- let $a = [a_i a_{i-1} \cdots a_0]_k$,
- let $b = [b_i b_{i-1} \cdots b_0]_k$, and
- let $c_{a,0}, c_{a,1}, \ldots, c_{a,i+1}, c_{b,0}, c_{b,1}, \ldots, c_{b,i+1} \in \mathbb{N}$ be two sets of carries.

Step $j$ begins by asserting that $b \cdot p$ and $a \cdot q$ agree on the next digit, which means that

$$p \cdot b_j + c_{b,j} \equiv q \cdot a_j + c_{a,j} \pmod{k}.$$

If they do agree, then we define that digit as $0 \leq m_j \leq k - 1$ and we can determine the carries

$$c_{b,j+1} = \frac{p \cdot b_j + c_{b,j} - m_j}{k}$$

and

$$c_{a,j+1} = \frac{q \cdot a_j + c_{a,j} - m_j}{k}$$

with the usual carry equation for each carry. We can unify the two sets of carries into one set that relates to both $a$ and $b$. Let $c_j = c_{b,j} - c_{a,j}$. We can compute $c_{j+1}$ as

$$\begin{aligned}
c_{j+1} &= c_{b,j+1} - c_{a,j+1} \\
c_{j+1} &= \frac{p \cdot b_j + c_{b,j} - m_j}{k} - \frac{q \cdot a_j + c_{a,j} - m_j}{k} \\
k \cdot c_{j+1} &= p \cdot b_j + c_{b,j} - m_j - q \cdot a_j - c_{a,j} + m_j \\
k \cdot c_{j+1} &= p \cdot b_j - q \cdot a_j + (c_{b,j} - c_{a,j}) + (m_j - m_j) \\
k \cdot c_{j+1} &= p \cdot b_j - q \cdot a_j + c_j. \qquad (2.5)
\end{aligned}$$

We call Equation 2.5 the *rational carry equation.*

**Corollary 29.** *Each carry* $-q < c_j < p$ *for* $0 \leq j \leq i+1$.

*Proof.* Lemma 17 gives us that $0 \leq c_{a,j} < q$ and $0 \leq c_{b,j} < p$. Therefore, the carry $c_j = c_{b,j} - c_{a,j}$ is less than $p - 0 = p$. Similarly, the carry $c_j$ is greater than $0 - q = -q$. □

Since the rational carry equation is just simplification of two different sets of carry equations merged together, it retains the property that one more digit is verified after each step. This resolves the concerns raised by the naive attempt to modify the extended long multiplication algorithm. However, the rational carry equation is difficult to display graphically in an analogous method to the extended long multiplication algorithm. Similarly to the carry equations, the rational carry equations give the desired result $b \cdot p = a \cdot q$ if each equation is valid and $c_0 = c_{j+1} = 0$. We can incorporate this new system into a new set of automata. We could create most-significant-digit-first automata or least-significant-digit-first automata with the rational carry equations analogously to the multiplication automata, but we only present most-significant-digit-first automata here, as all of the remaining results on regular ratio sets are based off of most-significant-digit-first automata.

**Definition 30.** Given natural number $k \in \mathbb{N}$ and rational $p/q \in Q^+$, the *base-k most-significant-digit-first multiplication automaton for* $p/q$ is an automaton

$$M = (\{q_{-q+1}, q_{-q+2}, \ldots, q_{-1}, q_0, q_1, \ldots, q_{p-1}\}, \{0, 1, \ldots, k-1\} \times \{0, 1, \ldots, k-1\}, \delta, q_0, \{q_0\}),$$

where $\delta(q_i, (j_a, j_b)) = q_{i'}$ for

$$i' = i \cdot k + q \cdot j_a - p \cdot j_b$$

if and only if $-q + 1 \leq i' \leq p - 1$. Otherwise, $\delta(q_i, (j_a, j_b))$ is undefined. We reject any input that leads to an undefined delta.

This new construction remains general to natural numbers, as $n = n/1$ is a valid rational and the automaton constructed for $n/1$ simply checks $b \cdot n = a \cdot 1 = a$.

**Example 31.** Figure 2.9 is the base-2 most-significant-digit-first multiplication automata for 2/3. One accepted input is

$$(0,1)(1,0)(1,1)(1,0)(0,1) = \langle 14, 21 \rangle_2.$$

Another notable property of this automata is that $q_0$ and $q_{-1}$ both have 3 transitions out of them, an example of the case where there is not a unique factor $a_j a_{j-1} \cdots a_0$ for a given partial input $b_j b_{j-1} \cdots b_0$.

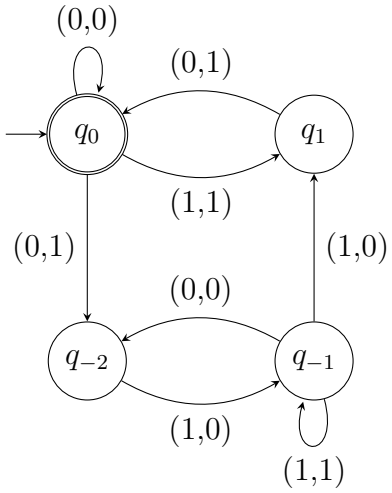


Figure 2.9: Base-2 most-significant-digit-first multiplication automaton for 2/3.

### 2.1.5 Restricting to a Regular Language

We return to the core decision problem of this chapter, `REGULAR RATIO SET MEMBERSHIP`. We assume that when given an instance of `REGULAR RATIO SET MEMBERSHIP`, we are also given automata $M_A$ and $M_B$ that accept words that are base-$k$ representations of natural numbers in $A$ and $B$ respectively. We can assume without loss of generality that $M_A$ and $M_B$ also accept representations that are padded with zeroes. This is important, as the encoded input $\langle a, b\rangle_k$ typically includes one of $\langle a\rangle_k$ or $\langle b\rangle_k$ being padded with zeroes up to the length of the other. We can determine if $p/q \in R(A, B)$ by combining the base-$k$ most-significant-digit-first multiplication automaton for $p/q$ with $M_A$ and $M_B$. We essentially apply the typical algorithm that builds an automaton to accept the intersection of two regular languages based on the automata that accept them. The main difference with our constructed automata is that we move through $M_A$ based on the first element of every $(a_j, b_j)$ pair and move through $M_B$ with the second element.

**Definition 32.**

- Given natural number $k \in \mathbb{N}$,
- sets $A, B \subseteq \mathbb{N}$,
- deterministic finite automaton $M_A = (Q_A, \Sigma_k, \delta_A, q_{A,0}, F_A)$ where $[L(M_A)]_k = A$,

- deterministic finite automaton $M_B = (Q_B, \Sigma_k, \delta_B, q_{B,0}, F_B)$ where $[L(M_B)]_k = B$, and
- rational $p/q \in Q^+$,

the *base-k $R(A,B)$ automaton for $p/q$*, $M(k, R(A,B), p/q)$, is an automaton

$$M(k, R(A,B), p/q) = (Q, \Sigma, \delta, (q_0, q_{A,0}, q_{B,0}), F),$$

where

- $Q = \{q_{-q+1}, q_{-q+2}, \ldots, q_{p-1}\} \times Q_A \times Q_B$,
- $\Sigma = \Sigma_k \times \Sigma_k$, and
- $F = \{q_0\} \times F_A \times F_B$.

We define $\delta$ as

$$\delta((q_i, q_{A,\alpha}, q_{B,\beta}), (j_a, j_b)) = (q_{i'}, \delta_A(q_{A,\alpha}, j_a), \delta_B(q_{B,\beta}, j_b)),$$

for

$$i' = i \cdot k + q \cdot j_a - p \cdot j_b$$

if and only if $-q + 1 \leq i' \leq p - 1$, $\delta_A(q_{A,\alpha}, j_a)$ is defined, and $\delta_B(q_{B,\beta}, j_b))$ is defined. Otherwise, $\delta(q_i, (j_a, j_b))$ is undefined. We reject any input that leads to an undefined delta.

A word $\langle a, b \rangle_k$ is accepted by $M(k, R(A,B), p/q)$ if and only if each of the primitive automata, the base-$k$ most-significant-digit-first multiplication automaton for $p/q$, $M_A$, and $M_B$, all accept their part of the input. This occurs exactly when $a/b = p/q$, $M_A$ accepts $a$, and $M_B$ accepts $b$. In this case, we have found $a \in A$ and $b \in B$ such that $a/b = p/q$. Therefore, $p/q \in R(A,B)$ if and only if $M(k, R(A,B), p/q)$ has some accepting path.

The automata $M(k, R(A,B), p/q)$ has $(p + q - 1) \cdot |M_A| \cdot |M_B|$ states. Each state has at most $k^2$ transitions out of it. Therefore, we can perform a breadth first search on $M(k, R(A,B), p/q)$ in

$$O(k^2(p + q - 1) \cdot |M_A| \cdot |M_B| + (p + q - 1) \cdot |M_A| \cdot |M_B|) = O(k^2(p + q) \cdot |M_A| \cdot |M_B|)$$

| Set $A_{k,1}$ | First terms | $\underline{d}(A_{k,1})$ | OEIS number |
|---|---|---|---|
| $A_{3,1}$ | $\{1, 3, 4, 5, 9, 10, 11, 12, 13, 14, \ldots\}$ | $1/2$ | A132141 |
| $A_{4,1}$ | $\{1, 4, 5, 6, 7, 16, 17, 18, 19, 20, \ldots\}$ | $1/3$ | A053738 |
| $A_{5,1}$ | $\{1, 5, 6, 7, 8, 9, 25, 26, 27, 28, \ldots\}$ | $1/4$ | Not in OEIS |
| $A_{6,1}$ | $\{1, 6, 7, 8, 9, 10, 11, 36, 37, 38, \ldots\}$ | $1/5$ | Not in OEIS |

Table 2.2: Sets of natural numbers whose base-$k$ representations start with 1.

time. This breadth first search of $M(k, R(A, B), p/q)$ is a decision algorithm for `REGULAR RATIO SET MEMBERSHIP`. Notably, this breadth first search also finds the smallest $a$ and $b$ such that $\langle a, b\rangle_k$ is accepted.

This algorithm serves as a proof for Theorem 12 from the beginning of this chapter. The given worst-case time complexity $O(k^2(p+q) \cdot |M_A| \cdot |M_B|)$ is achievable using a two-dimensional array of size $k^2 \times (p+q-1) \cdot |M_A| \cdot |M_B|$ to store the transition table of the constructed automata and then searching through the table with a breadth first search. However, in practice this approach is inefficient and extremely memory intensive. I have implemented this algorithm in the programming language Python and used it to compute all of the results in the rest of this chapter. The automata constructed by my Python program is stored in a Python dictionary, which is itself implemented by a hash table. This implementation gives the algorithm an asymptotically slower worst-case time bound. In practice, the code runs faster, uses significantly less memory, and becomes significantly easier to analyze and store. The algorithm and all computed results can be found on GitHub.

## 2.2 Representations Starting with One

We begin our explorations of specific ratio sets with a simple problem previously examined by Brown et al. [4]. Let $A_{k,i} = \{n \mid n \in N, \langle n\rangle_k[0] = i\}$ be the set of natural numbers whose base-$k$ representations start with the digit $i$. In this section we are exploring sets $A_{k,1}$ for $k \geq 3$. Table 2.2 contains examples of the first few sets $A_{k,1}$. Brown et al. prove the following theorem:

**Theorem 33.** *The set $A_{k,1}$ is $(R)$-dense for $k = 2, 3, 4$ but not for $k \geq 5$.*

The sets $A_{2,1} = \mathbb{N}$ and $A_{3,1}$ are $(R)$-dense, due to Theorem 7 regarding the relationship between $(R)$-density and asymptotic density. However, determining if $A_{k,1}$ is $(R)$-dense for

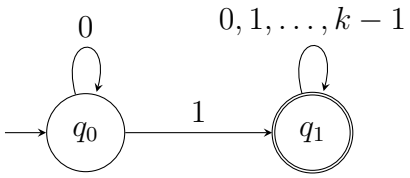


Figure 2.10: Automata that recognizes $A_{k,1}$.

$k \geq 4$ is more complicated. We add to the understanding of $R(A_{k,1})$ by using our `REGULAR RATIO SET MEMBERSHIP` algorithm to compute a portion of $\mathbb{N} \cap R(A_{k,1})$ for $k \geq 3$. We use that data to lead us to a more general result and subsequently our own proof of Theorem 33. All the raw data discussed in this subsection and the code used to generate it is available on GitHub. We examine the data extensively in this subsection in order explore the various ways the `REGULAR RATIO SET MEMBERSHIP` algorithm can be used to lead us to useful results.

Figure 2.10 is an automaton that accepts strings that begin with the digit 1, potentially padded with zeroes. This automaton serves as the encoding for $A_{k,1}$ that we incorporate into the multiplication automata.

**Example 34.** Figure 2.11 depicts the automaton $M(3, R(A_{3,1}), 5)$ that accepts $\langle a, b \rangle_3$ such that $a/b = 5$ and $a, b \in A_{3,1}$. (To simplify the drawing, the transition diagram in Figure 2.11 only includes accepting states and states that can lead to an accepting state.) Since the automaton in Figure 2.10 has $q_1$ as its only accepting state, the only accepting state in Figure 2.11 is $(0, 1, 1)$. It is not surprising that there is an accepting path, as $\langle 5, 1 \rangle_3$ is accepted since $\langle 5 \rangle_3 = 12$. However, using automata has the advantage of allowing us to easily extract various properties of the valid solutions $\langle a, b \rangle_3$. For example, we can deduce that $[12202(1022)^i]_3$ and $[01011(0021)^i]_3$ is a valid $a$ and $b$ pair for each $i \geq 0$. Similarly, since $[1221^i001]_3$ and $[0101^i102]_3$ is a valid $a$ and $b$ pair for each $i \geq 0$, we have a pair of solutions that each contain precisely $n$ ones for each $n \geq 2$.

While examination of all the pairs $\langle a, b \rangle_k$ such that $n = a/b$, $a \in A$, and $b \in B$ for some ratio set $R(A, B)$ is interesting, the typical first step along to path of understanding a ratio set is to simply examine many $n$ and attempting to deduce a pattern. We consider the question of if natural number $n \in R(A_{k,1})$ for $k = 3, 4, 5, 6$ up to approximately $n = 50000$. Table 2.3 contains the first few elements $n$ where the `REGULAR RATIO SET MEMBERSHIP` algorithm tells us that $n$ is not an element of the ratio set. We note that every examined $n \leq 59048 = 3^{10} - 1$ was an element of $A_{3,1}$. The data suggests that the natural numbers $n \in \mathbb{N}$ that are not an element of $R(A_{4,1})$ are of the form $2 \cdot 4^i$ for $i \geq 0$. However, the

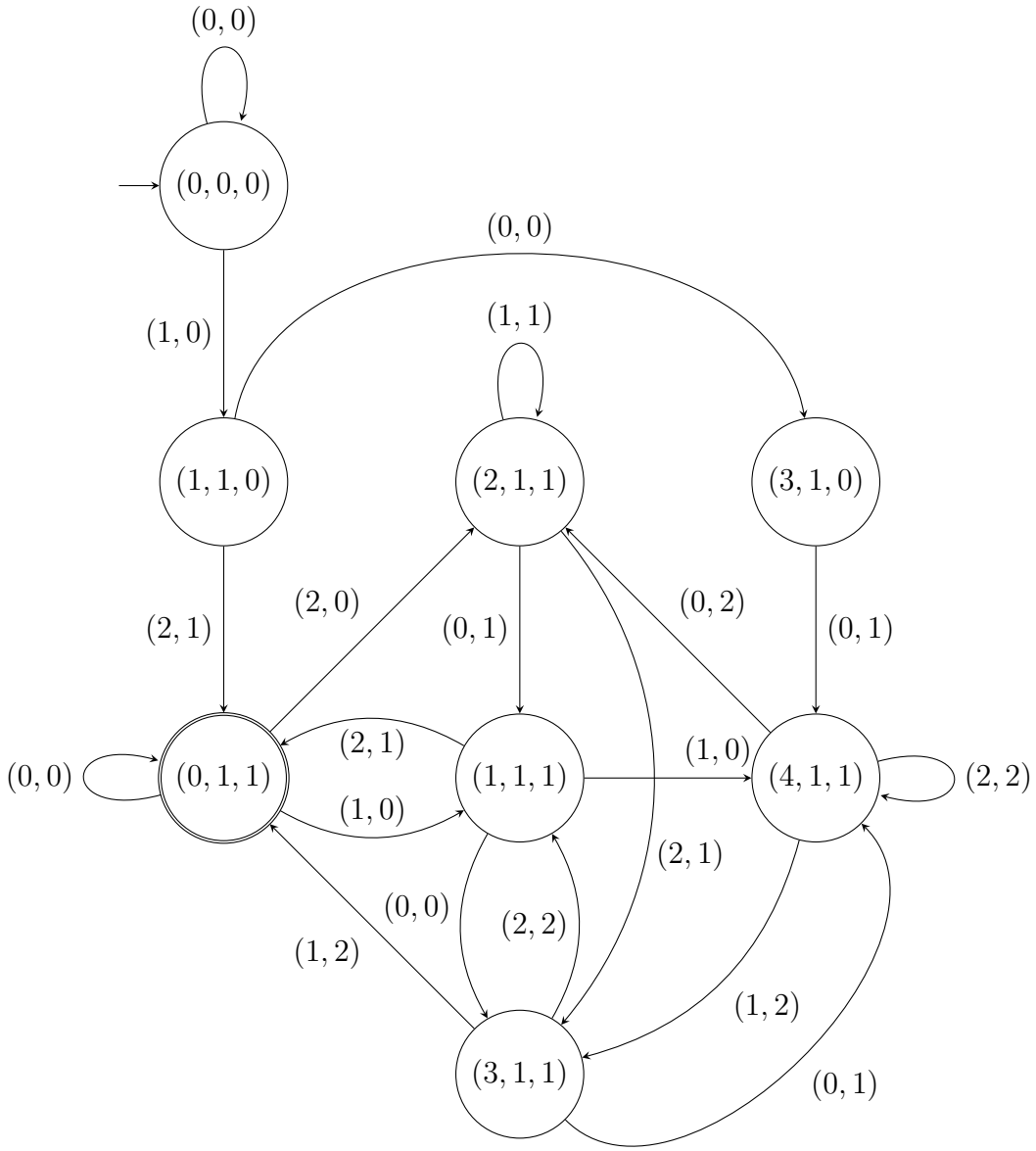


Figure 2.11: Automaton $M(3, R(A_{3,1}), 5)$.

| $A_{k,1}$ | First natural numbers $n < 50000$ not an element of $R(A_{k,1})$ |
|---|---|
| $A_{3,1}$ | $\{\}$ |
| $A_{4,1}$ | $\{2, 8, 32, 128, 512, 2048, 8192, 32768\}$ |
| $A_{5,1}$ | $\{2, 10, 11, 12, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 250, 251, 252, \ldots\}$ |
| $A_{6,1}$ | $\{2, 3, 12, 13, 14, 15, 16, 17, 18, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, \ldots\}$ |

Table 2.3: First natural numbers $n < 50000$ not an element of $R(A_{k,1})$.

| $R(A_{4,1})$ | | | $R(A_{5,1})$ | | | $R(A_{6,1})$ | | |
|---|---|---|---|---|---|---|---|---|
| $n$ | $\underline{a}(n)$ | $\underline{b}(n)$ | $n$ | $\underline{a}(n)$ | $\underline{b}(n)$ | $n$ | $\underline{a}(n)$ | $\underline{b}(n)$ |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | 18 | 6 | 3 | 27 | 9 | 4 | 36 | 9 |
| 9 | 261 | 29 | 13 | 637 | 49 | 19 | 1311 | 69 |
| 33 | 4125 | 125 | 63 | 15687 | 249 | 109 | 46761 | 429 |
| 129 | 65661 | 509 | 313 | 390937 | 1249 | 649 | 1680261 | 2589 |
| 513 | 1049085 | 2045 | 1563 | 9767187 | 6249 | 3889 | 60470061 | 15549 |
| 2049 | 16779261 | 8189 | 7813 | 244148437 | 31249 | 23329 | 2176805661 | 93309 |

Table 2.4: Natural numbers $n \in R(A_{k,1})$ such that $\underline{b}(n) > \underline{b}(n')$ for each $n' < n$.

data is more complicated for $A_{5,1}$ and $A_{6,1}$ so we examine some related properties to try and gain further insight.

Another interesting metric is the $n$ such that $\underline{b}(n)$ is larger than $\underline{b}(n')$ for all $n' < n$. These are the "worst cases" for trying to upper bound the size of the smallest representation $n = a/b$. We examine these $n$ for $R(A_{4,1})$, $R(A_{5,1})$, $R(A_{6,1})$ in Table 2.4. We omit $R(A_{3,1})$ but note that every $n$ examined had smallest $b \in \{1, 4, 5\}$. A notable observation from these tables is that the worst case $n$ are always slightly more than $\frac{1}{2} \cdot k^i$ for some $i \in \mathbb{N}$. Furthermore, the $\underline{b}(n)$ for the worst cases $n$ seem to be consistently slightly less than $2 \cdot k^i$.

Drawing a parallel between Table 2.3 and Table 2.4, it appears that the worst cases seem to occur directly after a run of natural numbers that are not elements of $R(A_{k,1})$. In this case, we can reformulate Table 2.3 by specifically examining the runs of natural numbers not in the ratio set. This leads us to Table 2.5, which contains the size and starting index of runs of natural numbers that are not elements of $R(A_{k,1})$. Each of the runs begins at $2 \cdot k^i$ and ends at $\left\lfloor \frac{k}{2} \cdot k^i \right\rfloor$ for some $i \in \mathbb{N}$. We also include Table 2.6, which contains every run of consecutive $n \in \mathbb{N}$ that have the same $\underline{b}(n)$ for $R(A_{6,1})$. (We denote the case where there is no accepted $b$ with $\underline{b}(n) = -1$.) We observe that each $\underline{b}(n)$ is of

| $R(A_{4,1})$ | | $R(A_{5,1})$ | | $R(A_{6,1})$ | |
|---|---|---|---|---|---|
| Size of gap | Starting $n$ | Size of gap | Starting $n$ | Size of gap | Starting $n$ |
| 1 | 2 | 1 | 2 | 2 | 2 |
| 1 | 8 | 3 | 10 | 7 | 12 |
| 1 | 32 | 13 | 50 | 37 | 72 |
| 1 | 128 | 63 | 250 | 217 | 432 |
| 1 | 512 | 313 | 1250 | 1297 | 2592 |
| 1 | 2048 | 1563 | 6250 | 7777 | 15552 |

Table 2.5: Gaps in $\mathbb{N} \cap R(A_{k,1})$.

the form $2 \cdot 6^k - i$ for $i \in \{1, 2, \ldots, 5\}$. Informed by the intervals and patterns presented in these tables, we come to the following theorem:

**Theorem 35.** *Let $k \geq 4$.*

$$\mathbb{N} \cap R(A_{k,1}) = (\mathbb{N} \setminus \{0\}) \setminus \bigcup_{i=1}^{\infty} [2 \cdot k^i, \frac{k}{2} \cdot k^i].$$

*Furthermore, if $n \in \mathbb{N} \cap R(A_{k,1})$, then there exists $a, b \in A_{k,1}$ such that $n = a/b$ and $b = 1$ or $b = 2 \cdot k^i - \sigma$ for $\sigma \in \{1, 2, \ldots, k-1\}$ and $i \geq 1$.*

*Proof.* We begin by dividing each interval $[k^i, k^{i+1})$ into several disjoint intervals.

$$\left[k^i, k^{i+1}\right) = \left[k^i, 2 \cdot k^i\right) \cup \left[2 \cdot k^i, \frac{k}{2} \cdot k^i\right] \cup \bigcup_{j=1}^{\infty} \left\{ \bigcup_{\ell=1}^{k-1} \left[ \frac{k^{j+1}}{2 \cdot k^j - \ell} \cdot k^i, \frac{k^{j+1}}{2 \cdot k^j - (\ell+1)} \cdot k^i \right) \right\}$$

We consider the natural numbers in each interval separately.

The natural numbers $n \in [k^i, 2 \cdot k^i) \cap \mathbb{N}$ are the natural numbers with base-$k$ representations that start with 1. These are clearly in $R(A_{k,1})$, as $n = \frac{n}{1}$ and $n, 1 \in A_{k,1}$.

The natural numbers $n \in [2 \cdot k^i, \frac{k}{2} \cdot k^i] \cap \mathbb{N}$ are not in $R(A_{k,1})$. If there was such an $n$, we would have that $n = \frac{a}{b}$ for some $a, b \in A_{k,1}$ but

$$a = n \cdot b \in \left[2 \cdot k^i \cdot k^{i'}, \frac{k}{2} \cdot k^i \cdot 2 \cdot k^{i'}\right) = \left[2 \cdot k^{i+i'}, k^{i+i'+1}\right)$$

since $b \in [k^{i'}, 2 \cdot k^{i'})$ for some $i' \in \mathbb{N}$. Therefore, the base-$k$ representation of $a$ begins with a symbol other than 1 so $a \notin A_{k,1}$, which is a contradiction.

| Size of run | $\underline{b}(n)$ | Starting $n$ |
|---|---|---|
| 1 | 1 | 1 |
| 2 | $-1$ | 2 |
| 1 | 9 | 4 |
| 1 | 8 | 5 |
| 6 | 1 | 6 |
| 7 | $-1$ | 12 |
| 1 | 69 | 19 |
| 2 | 11 | 20 |
| 2 | 10 | 22 |
| 3 | 9 | 24 |
| 4 | 8 | 27 |
| 5 | 7 | 31 |
| 36 | 1 | 36 |
| 37 | $-1$ | 72 |
| 1 | 429 | 109 |
| 2 | 71 | 110 |
| 1 | 70 | 112 |
| 2 | 69 | 113 |
| 2 | 68 | 115 |
| 1 | 67 | 117 |
| 12 | 11 | 118 |
| 14 | 10 | 130 |
| 18 | 9 | 144 |
| 24 | 8 | 162 |
| 30 | 7 | 186 |
| 216 | 1 | 216 |
| 217 | $-1$ | 432 |
| 1 | 2589 | 649 |
| 2 | 431 | 650 |
| 1 | 430 | 652 |
| 2 | 429 | 653 |
| 1 | 428 | 655 |
| 2 | 427 | 656 |
| 9 | 71 | 658 |

Table 2.6: Runs of $n \in R(A_{6,1})$ that have the same $\underline{b}(n)$.

Lastly, we consider intervals $\left[\frac{k^{j+1}}{2\cdot k^j-\ell}\cdot k^i, \frac{k^{j+1}}{2\cdot k^j-(\ell+1)}\cdot k^i\right)$ for $i \geq 1, j \geq 1, \ell \in \{1, 2, \ldots, k-1\}$. We fix a specific $i, j, \ell$ and consider the natural numbers $n$ in that range. We consider $n$ multiplied by $2 \cdot k^j - \ell$ and observe that

$$\begin{aligned} n \cdot (2 \cdot k^j - \ell) &\in \left[\frac{k^{j+1}}{2 \cdot k^j - \ell} \cdot k^i \cdot (2 \cdot k^j - \ell), \frac{k^{j+1}}{2 \cdot k^j - (\ell+1)} \cdot k^i \cdot (2 \cdot k^j - \ell)\right) \\ &\in \left[k^{i+j+1}, \frac{(2 \cdot k^j - \ell)}{2 \cdot k^j - (\ell+1)} \cdot k^{i+j+1}\right) \subseteq [k^{i+j+1}, 2 \cdot k^{i+j+1}). \end{aligned}$$

This gives us that $n = \frac{n\cdot(2\cdot k^j-\ell)}{2\cdot k^j-\ell}$, where $n \cdot (2 \cdot k^j - \ell), (2 \cdot k^j - \ell) \in A_{k,1}$. □

Furthermore, we get the following corollary:

**Corollary 36.** *Let $k \geq 4$. For each real number*

$$r \in \mathbb{R}^+ \setminus \bigcup_{j=-\infty}^{\infty} (2 \cdot k^j, \frac{k}{2} \cdot k^j) = R'$$

*and $\epsilon > 0$ there exists $p/q \in R(A_{k,1})$ such that $|r - p/q| < \epsilon$.*

*Proof.* Choose $i$ such that $k^{-i+1} < \epsilon$. Let $r' = \lfloor r \cdot k^i \rfloor$. We note that $\left|r - \frac{r'}{k^i}\right| < \frac{\epsilon}{k}$.

If $r' = 0$ then $0 < r < k^{-i} < \epsilon$. We have that $k^{-i} \in R(A_{k,1})$, as $1, k^i \in A_{k,1}$ and $k^{-i} = 1/k^i$. We then observe that $|r - k^{-i}| < \epsilon$, as required. Therefore, we can assume without loss of generality that $r' > 0$.

Given our restriction on $r$, we get that $r'$ is a natural number such that

$$r' \in \mathbb{N} \setminus \bigcup_{j=-\infty}^{\infty} (2 \cdot k^j \cdot k^i, \frac{k}{2} \cdot k^j \cdot k^i).$$

Therefore, either $r' = p/q$ for some $p, q \in A_{k,1}$, $r' = 2 \cdot k^\ell$ for some $\ell \in \mathbb{N}$, or $r' = \frac{k}{2} \cdot k^\ell$ for some $\ell \in \mathbb{N}$.

If $r' = p/q$ for some $p, q \in A_{k,1}$, then

$$\frac{r'}{k^i} = \frac{p}{q \cdot k^i} \in R(A_{k,1})$$

since $q \cdot k^i \in A_{k,i}$ if and only if $q \in A_{k,i}$. In this case, we get that

$$\left| r - \frac{p}{q \cdot k^i} \right| < \epsilon$$

as desired.

If $r' = 2 \cdot k^\ell$, then $r = 2 \cdot k^{\ell - i}$ since $2 \cdot k^{\ell-i} + m \notin R'$ for $0 < m < k^{-i}$. We have that $[1(k-1)^\ell]_k \in A_{k,1}$ and

$$\begin{aligned} r - \frac{[1(k-1)^\ell]_k}{k^i} &= \frac{2 \cdot k^\ell}{k^i} - \frac{[1(k-1)^\ell]_k}{k^i} \\ &= \frac{2 \cdot k^\ell - [1(k-1)^\ell]_k}{k^i} \\ &= \frac{1}{k^i} = k^{-i} < \epsilon. \end{aligned}$$

If $r' = \frac{k}{2} \cdot k^\ell$ for some $\ell \in \mathbb{N}$, then $r = \frac{k}{2} \cdot k^{\ell - i} + m$ for $m \leq \frac{\epsilon}{k}$. We have that $[1(k-1)^{\ell+1}]_k \in A_{k,1}$. Let $s = \frac{k^{\ell+1}}{[1(k-1)^{\ell+1}]_k} \in R(A_{k,1})$. Then

$$\begin{aligned} s - (r - m) &= \frac{k^{\ell+1}}{[1(k-1)^{\ell+1}]_k} - \frac{k^{\ell+1}}{2 \cdot k^i} \\ &= \frac{2 \cdot k^i \cdot k^{\ell+1}}{2 \cdot k^i \cdot [1(k-1)^{\ell+1}]_k} - \frac{[1(k-1)^{\ell+1}]_k \cdot k^{\ell+1}}{2 \cdot k^i \cdot [1(k-1)^{\ell+1}]_k} \\ &= \frac{(2 \cdot k^i - [1(k-1)^{\ell+1}]_k) \cdot k^{\ell+1}}{2 \cdot k^i \cdot [1(k-1)^{\ell+1}]_k} \\ &= \frac{1 \cdot k^{\ell+1}}{2 \cdot k^i \cdot [1(k-1)^{\ell+1}]_k} \\ &< \frac{k^{\ell+1}}{2 \cdot k^i \cdot 2 \cdot k^{\ell+1}} \\ &< \frac{k^{\ell+1}}{4 \cdot k^{i+\ell+1}} \\ &< \frac{k^{\ell+1-i-\ell-1}}{4} \\ &< \frac{k^{-i}}{4} \\ &< k^{-i} < \frac{\epsilon}{k}. \end{aligned}$$

Therefore, $|r - s| < \frac{2\epsilon}{k} < \epsilon$, as required. □

Corollary 36 shows that $R(A_{4,1})$ is $(R)$-dense since $(2 \cdot 4^i, \frac{4}{2} \cdot 4^i) = \emptyset$. Similarly, it shows that $R(A_{k,1})$ is not $(R)$-dense for $k \geq 5$ since $(2 \cdot k^i, \frac{k}{2} \cdot k^i) \neq \emptyset$. This is our generalization of Theorem 33.

## 2.3 Fibbinary Numbers

We now return to the analysis of Fibbinary numbers $\texttt{FIB} = \{0, 1, 2, 4, 5, 8, 9, 10, 16, 17, \ldots\}$, which form sequence A003714 in the OEIS. We have already shown that $\langle \texttt{FIB} \rangle_2$ is regular, so the ratio sets $R(\texttt{FIB}, \mathbb{N})$ and $R(\texttt{FIB})$ are tractable for analysis with our `REGULAR RATIO SET MEMBERSHIP` algorithm.

**Example 37.** To build $M(2, R(\texttt{FIB}, \mathbb{N}), p/q)$, we need an automaton that recognizes $\langle \texttt{FIB} \rangle_2$ and $\langle \mathbb{N} \setminus \{0\} \rangle_2$. We have already seen Figure 1.1, which is an automaton that recognizes $\langle \texttt{FIB} \rangle_2$. We use that automaton except that we omit $q_3$ so that transitions that lead to $q_3$ are implicit failures. This slightly simplifies $M(2, R(\texttt{FIB}, \mathbb{N}), p/q)$. We also note that $\langle \mathbb{N} \setminus \{0\} \rangle_2$ is accepted by the automaton in Figure 2.10 when we set $k = 2$. These are all we need for $M(2, R(\texttt{FIB}, \mathbb{N}), p/q)$. We give an example of such a construction in Figure 2.12, which is the automaton $M(2, R(\texttt{FIB}, \mathbb{N}), 3)$. By examining the transition diagram, we can easily conclude that $\underline{b}(3) = 3$.

**Example 38.** Figure 2.13 is the transition diagram of the automaton $M(2, R(\texttt{FIB}), 5)$ restricted to states that could lead to an accepting state. The full automaton has 11 states. By examining the automaton, we find that every $a$ and $b$ such that $a, b \in \texttt{FIB}$ and $a/b = 5$ are of the form $\langle a \rangle_2 = 101(00^*101)^*0^*$ and $\langle b \rangle_2 = 001(00^*001)^*0^*$.

Before we examine the ratio sets, we present a few facts about Fibbinary numbers and their ratio sets:

*Observation* 39.

1. The Fibbinary numbers
$$\texttt{FIB} \subseteq \bigcup_{i=0}^{\infty} \left[ 2^i, \frac{4}{3} \cdot 2^i \right).$$

2. Given Item 1, we get that
$$R(\texttt{FIB}) \subseteq \bigcup_{i=0}^{\infty} \left( \frac{3}{4} \cdot 2^i, \frac{4}{3} \cdot 2^i \right).$$

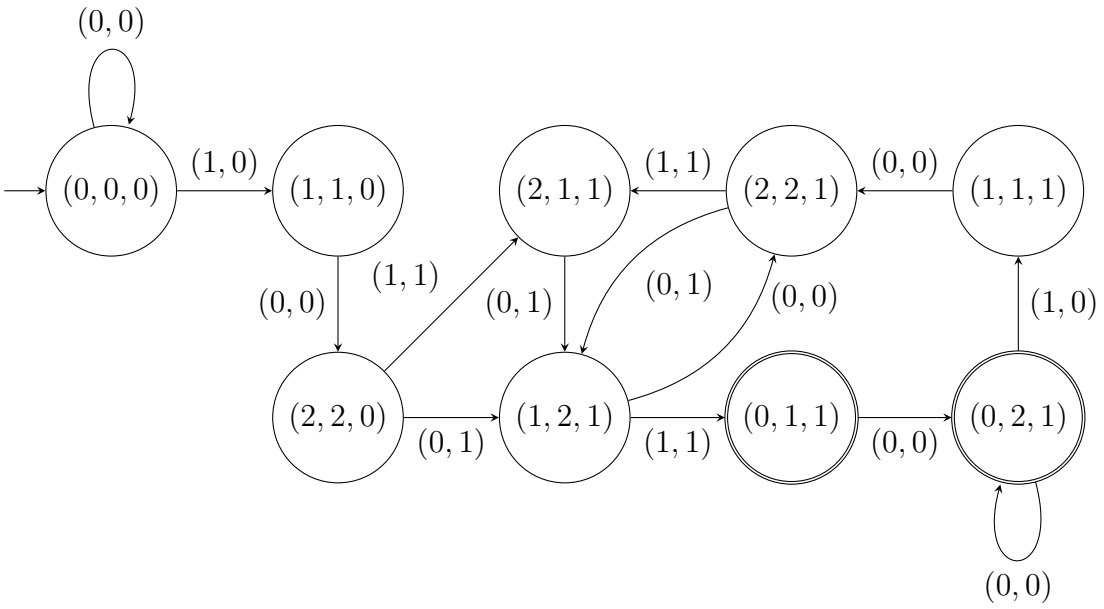


Figure 2.12: Automaton $M(2, R(\texttt{FIB}, \mathbb{N}), 3)$.

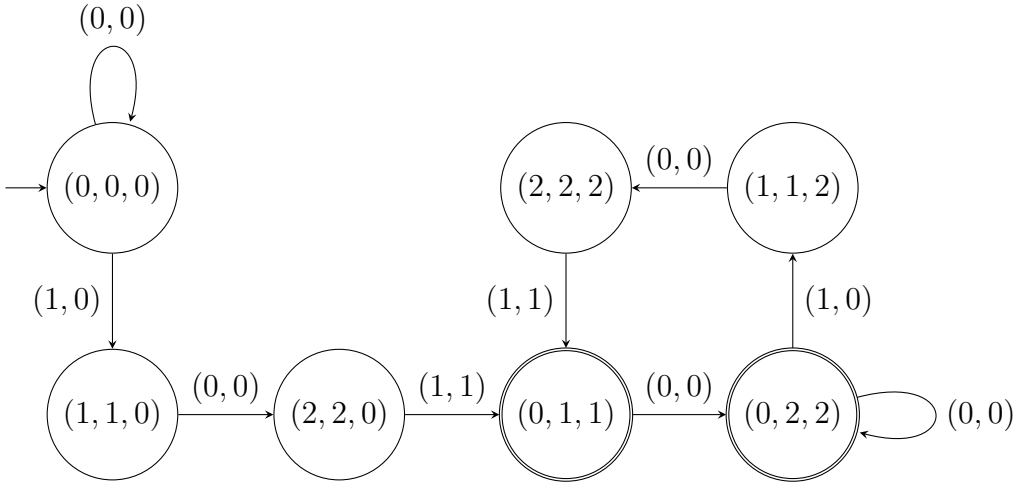


Figure 2.13: Automaton $M(2, R(\texttt{FIB}), 5)$.

3. A natural number $n \in \mathbb{N}$ has the property $n \in R(\texttt{FIB})$ if and only if $2n \in R(\texttt{FIB})$.

4. For $n \in \mathbb{N}$, if $n \equiv 3 \pmod 4$, then $n \notin R(\texttt{FIB})$.

Observation 39.1 shows that FIB has many gaps in the natural numbers, which limits the effectiveness of the $(R)$-density theorems presented in Section 1.2. However, Observation 39.3 is useful, as it means we only need to characterize the odd natural numbers in $R(\texttt{FIB})$ to understand all the natural numbers in $R(\texttt{FIB})$.

All the raw data discussed in this section and the code used to generate it is available on GitHub. Before we examine $R(\texttt{FIB})$, we briefly note that the data computed for $R(\texttt{FIB}, \mathbb{N})$ suggests the following conjecture:

**Conjecture 40.** For all $n \in \mathbb{N}$, we have that $n \in R(\texttt{FIB}, \mathbb{N})$.

The data for $R(\texttt{FIB})$ was more complicated and included many natural numbers $n$ where $n \notin R(\texttt{FIB})$. However, those natural numbers followed a particular pattern and led to the following conjecture:

**Conjecture 41.** Every natural number in the ratio set $R(\texttt{FIB})$ has a base-2 representation of one of the following forms:

1. $10^*$
2. $11(0,1)^*010^*$
3. $10(0,1)^*010^*$ where the factor 11 does not appear before the factor 100 reading left to right.

This is exactly every natural number $n$ in the interval

$$\bigcup_{i=0}^{\infty} \left( \frac{3}{4} \cdot 2^i, \frac{4}{3} \cdot 2^i \right)$$

except for natural numbers $n = 2^j \cdot m$ for $m \equiv 3 \pmod 4$.

Upon hearing this conjecture, Kevin Hare suggested an algorithm that approximates real numbers

$$r \in \bigcup_{i=-\infty}^{\infty} \left( \frac{3}{4} \cdot 2^i, \frac{4}{3} \cdot 2^i \right)$$

with elements of $R(\texttt{FIB})$ [15]. We use that algorithm to prove the following theorem:

**Theorem 42.** *For each natural number*

$$n \in \mathbb{N} \cap \bigcup_{j=0}^{\infty} \left( \frac{3}{4} \cdot 2^j, \frac{4}{3} \cdot 2^j \right)$$

*and positive real number $\epsilon > 0$, there exists $a/b \in R(\texttt{FIB})$ such that $|n - a/b| < \epsilon$.*

*Proof.* Let

$$n \in \mathbb{N} \cap \bigcup_{j=0}^{\infty} \left( \frac{3}{4} \cdot 2^j, \frac{4}{3} \cdot 2^j \right)$$

be the natural number we want to approximate. Define an *$f$-prefix* $x$ of a natural number $a \in \mathbb{N}$ as a prefix $x = \langle a \rangle_2[0..(\ell - 1)]$ of $\langle a \rangle_2$ with associated word $y$, such that either $x = y001^*$ or $x = y$ for some $[y]_2 \in \texttt{FIB}$, or $x = 1^*$, in which case we define $y = \epsilon$.

We build a sequence

$$\frac{a_1}{b_1}, \frac{a_2}{b_2}, \ldots, \frac{a_i}{b_i}, \ldots \tag{2.6}$$

such that $a_i/b_i = n$, $b_{i+1} > b_i$, $b_i \in \texttt{FIB}$, and $a_i$ has longest $f$-prefix $x_i$ such that $|\langle a_i \rangle_2| - |x_i| \leq |\langle n \rangle_2|$. We can use $a_i/b_i$ to get an approximation $a_i'/b_i \in R(\texttt{FIB})$ to $n$ of arbitrary precision. Assume Sequence (2.6) exists. Let $x_i$ be the longest $f$-prefix of $a_i$ with associated word $y_i$.

If $x_i = y_i$, then $x_i = y_i = \langle a_i \rangle_2$ and we have found an exact solution. Otherwise, we would have that $x_i a[|x_i|]$ is a longer $f$-prefix of $a$. In the case that $x_i = y_i$, we have that $a_i/b_i = n \in R(\texttt{FIB})$ and $|n - a_i/b_i| = 0 < \epsilon$ for any $\epsilon > 0$.

So instead, assume that $x_i = y001^*$ or $x_i = 1^*$. In either case, we have $a_i = x_i z_i$ for some $z_i \in \Sigma_2^*$ such that $|z_i| \leq |\langle n \rangle_2|$. As well, we have that $[x_i]_2 + 1 \in \texttt{FIB}$ in either case, since adding 1 removes the $1^*$ suffix of $x_i$ and the zeroes in front of the $1^*$ suffix ensures that the addition does create a new set of repeated 1 symbols. Let $a_i' = ([x_i]_2 + 1) \cdot 2^{|z_i|}$.

We get that

$$\begin{aligned}
\left| n - \frac{a'_i}{b_i} \right| &= \left| \frac{a_i}{b_i} - \frac{a'_i}{b_i} \right| \\
&= \left| \frac{a_i - a'_i}{b_i} \right| \\
&= \left| \frac{([x_i]_2 \cdot 2^{|z_i|} + [z_i]_2) - (([x_i]_2 + 1) \cdot 2^{|z_i|})}{b_i} \right| \\
&= \left| \frac{[z_i]_2 - 2^{|z_i|}}{b_i} \right| \\
&\leq \frac{2^{|z_i|}}{b_i} \\
&\leq \frac{2^{|\langle n \rangle_2|}}{b_i}.
\end{aligned}$$

However, we know that $2^{|\langle n \rangle_2|}$ is constant, whereas $b_i \to \infty$ as $i \to \infty$. Therefore, we have that $\left| n - \frac{a'_i}{b_i} \right| \to 0$ as $i \to \infty$ and can find an approximation for any $\epsilon > 0$.

All that remains is to prove that Sequence (2.6) exists in all cases. We compute a sequence $a_i$ and $b_i$ iteratively, beginning with $a_1 = n$ and $b_1 = 1$. At each step, we choose a natural number $m_i \in \mathbb{N}$, and then use $m_i$ to compute $a_{i+1} = a_i \cdot 2^{m_i} + n$ and $b_{i+1} = b_i \cdot 2^{m_i} + 1$. Clearly this generates a sequence $a_i / b_i$ with the property that $b_i \cdot n = a_i$. The properties $b_{i+1} > b_i$ and $b_i \in \texttt{FIB}$ both hold for the sequence we compute as long as $m_i \geq 2$ for all $i$.

To aid our proof, we reframe our understanding of the $m_i$ through a visual aid:

$$\begin{array}{rl}
\phantom{+}\ 1\,1\,1\,0\,0\,1\,0\,0\,0 & (a_1 \cdot 2^3) \\
+\qquad 1\,1\,1\,0\,0\,1 & (n) \\
\hline
\ 1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0 & (a_2 \cdot 2^4) \\
+\qquad\qquad\qquad 1\,1\,1\,0\,0\,1 & (n) \\
\hline
\ 1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,1 & (a_3) \\
\hline
\quad \alpha \quad \beta \quad\ \ \gamma &
\end{array}$$

Figure 2.14: The calculation $([111001]_2 \cdot 2^3 + [111001]_2) \cdot 2^4 + [111001]_2$.

We can view the operation $a_i \mapsto a_i \cdot 2^{m_i} + n$ as adding $n$ to $a_i$ with the leading digit of $n$ at a specific index. We can then simply track the changes to $b_i$ and otherwise ignore it,

instead focusing on where precisely we add $n$ to $a_i$. Under this interpretation, the chosen $m_i$ at each step is the difference between the indices where we add the leading digit of $n$ in step $i-1$ and $i$. For example, Figure 2.14 depicts a few iterations of the algorithm with $n = [111001]_2$, $a_1 = [111001]_2$, $b_1 = 1$, $m_1 = 3$, $m_2 = 4$. We view the index of the $\alpha$ column as the index of the leading digit of $n$ added into $a_i$ for an implicit step 0. We observe that $m_1$ is the difference in index between the $\beta$ column and $\alpha$ column and that $m_2$ is the difference in index between the $\gamma$ column and $\beta$ column. Therefore, we proceed by describing where the algorithm adds $n$ to $a_i$ at each step and the difference between that index and where $n$ is added in the next step. If this is always at least two symbols later, then we have that $m_i \leq 2$ for all $i$.

Additionally, we must verify that the factor of each $a_i$ not contained in its longest $f$-prefix $x_i$ has length at most $|\langle n \rangle_2|$. When we add $n$ into $a_i$, we always have some symbols of $n$ that trail past the least significant digit of $a_i$. Therefore, if the $f$-prefix $x_{i+1}$ of $a_{i+1}$ includes symbols up until the index of the leading digit of the $n$ that we add, then the number of symbols of $\langle a_{i+1} \rangle_2$ not included in the $f$-prefix $x_{i+1}$ must be less than the length of $\langle n \rangle_2$. This is the property we desire, so we show for each case that the $f$-prefix is extended past where we add in $n$.

We now analyze the two broad cases of $n$. We know that

$$n \in \mathbb{N} \cap \bigcup_{j=0}^{\infty} \left( \frac{3}{4} \cdot 2^j, \frac{4}{3} \cdot 2^j \right).$$

If

$$n \in \left( \frac{3}{4} \cdot 2^j, 2^j \right)$$

for some $j$, then $\langle n \rangle_2 = 11z$ for some $z \in \Sigma_2^*$. We refer to this as the (11) case. We note that in this case, we have that $n$ itself has an $f$-prefix of length at least 2. This ensures that our required properties are satisfied for $i = 1$. Otherwise, we have that

$$n \in \left[ 2^j, \frac{4}{3} \cdot 2^j \right)$$

for some $j$. In this case, our natural number $n$ has binary representation $\langle n \rangle_2 = 10z$ for some $z \in \Sigma_2^*$ with the restriction that the factor 11 does not appear before the factor 100 when reading $\langle n \rangle_2$ left to right. We refer to this as the (10) case. In this case, we have that $n$ has an $f$-prefix of length at least 3. Therefore, the Sequence (2.6) properties hold for $i = 1$ in all cases.

We begin by describing the (11) case. Let $\langle n \rangle_2 = 11 n_2 n_3 \cdots n_\ell$. At each step in this case, we add $n$ into $a_i$ immediately after the $f$-prefix $x_i$ of $a_i$. Let $a_i = x_i z_i$, $x_i = y_i 001^k$, and $z_i = z_0 z_1 \cdots z_j$. Consider the addition $a_i \cdot 2^{m_i} + n$, which we draw in Figure 2.15.

$$\begin{array}{rccccccccc} & y_i & 0 & 0 & 1^k & 0 & z_1 & z_2 & z_3 & \cdots \\ + & & & & & 1 & 1 & n_2 & n_3 & \cdots \\ \hline & y_i & 0 & \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \cdots \end{array}$$

Figure 2.15: The addition $a_i \cdot 2^{m_i} + n$ for the (11) case.

Recall that we need to verify that the next place we add $n$ is at least two indices later than where we add it here, which is the $\alpha_2$ column. As well, we need to ensure that the longest $f$-prefix includes at least up to the $\alpha_3$ column. Since we always add $n$ immediately after the $f$-prefix, showing that the $f$-prefix $x_i$ of $a_{i+1}$ includes up to $\alpha_3$ suffices to prove both conditions. If there is no carry from the addition in the $\alpha_3$ column, then $a_{i+1}$ has prefix $y'001^k11$, which is a valid $f$-prefix. So assume that there is a carry from the $\alpha_3$ column. In this case, we have that $a_{i+1}$ has prefix $y'010^k0\alpha_3$, which is an $f$-prefix regardless of $\alpha_3 \in \Sigma_2$. Therefore, in the (11) case, we can build a valid Sequence (2.6) and find an arbitrarily good approximation for $n$.

We now turn to the (10) case. This case is divided in to two subcases, which depend on the two last symbols of the $f$-prefix $x_i$. Since $x_i$ is the longest $f$-prefix of $a_i$, the next symbol after $x_i$ is always zero. We refer to the case where the $f$-prefix $x_i$ of $a_i$ ends with the factor 11 as the $(10-11)$ case. In this case, we add $n$ to $a_i$ by adding the leading digit of $n$ to the column of the last symbol of $x_i$. This is one symbol earlier than the other cases, and is what complicates the analysis of the (10) case. We refer to the case where the $f$-prefix $x_i$ of $a_i$ ends with the factor 01 as the $(10-01)$ case. In this case, we add $n$ to $a_i$ by adding the leading digit of $n$ to the column directly after the $f$-prefix of $x_i$. This is the same reletive index wherein we add $n$ in the (11) case. An important observation from the two subcases is that we exclusively add $n$ to an index which has preceeding digit 1.

We first consider the $(10-11)$ case. Let $\langle n \rangle_2 = 10 n_2 n_3 \cdots n_\ell$. Let $a_i = x_i z_i$, $x_i = y_i 001^k$, $k \geq 2$, and $z_i = 00 z_2 z_3 \cdots z_j$. Consider the addition $a_i \cdot 2^{m_i} + n$, which we draw in Figure 2.16.

$$\begin{array}{rccccccccc} & y_i & 0 & 0 & 1^{k-1} & 1 & 0 & z_1 & z_2 & \cdots \\ + & & & & & 1 & 0 & n_2 & n_3 & \cdots \\ \hline & y_i & 0 & 1 & 0^{k-1} & 0 & \alpha_3 & \alpha_4 & \alpha_5 & \cdots \end{array}$$

Figure 2.16: The addition $a_i \cdot 2^{m_i} + n$ for the $(10-11)$ case.

The $f$-prefix condition is always satisfied, since $y_i 010^{k-1} 0\alpha_3$ is a valid $f$-prefix for any $\alpha_3 \in \Sigma_2$. We must ensure that the next index where we add $n$ is at least the index of the column of $\alpha_4$. As we previously observered, in the (10) case, we never add $n$ to an index preceeded by a zero. Therefore, the earliest in $a_{i+1}$ we could add $n$ is $\alpha_4$, which means that the $(10-11)$ case also generates a valid Sequence (2.6).

Lastly, we consider the $(10-01)$ case, which we draw in Figure 2.17. As before, let $\langle n \rangle_2 = 10 n_2 n_3 \cdots n_\ell$. Let $a_i = x_i z_i$, $x_i = y_i 001$. We note that $z_i = 01 z_2 z_3 \cdots z_j$, since any other two initial symbols for $z_i$ would contradict the fact that $x_i$ is the longest $f$-prefix of $a_i$.

$$\begin{array}{rcccccccc} & y_i & 0 & 0 & 1 & 0 & 1 & z_2 & z_3 \cdots \\ + & & & & & 1 & 0 & n_2 & n_3 \cdots \\ \hline & y_i & 0 & \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 \cdots \end{array}$$

Figure 2.17: The addition $a_i \cdot 2^{m_i} + n$ for the $(10-01)$ case.

This case is determined by the $\alpha_4$ column. If the $\alpha_4$ column addition does not have a carry, then $a_{i+1}$ has $f$-prefix $y_i 001^k 11$ which has increased past the $\alpha_2$ column, which is the column where we add $n$. This means that the $f$-prefix property is satisfied when the $\alpha_4$ column does not have a carry. We must verify that the next location where we add $n$ is at least $\alpha_4$, since the leading digit of $n$ is in the $\alpha_2$ column. If the $\alpha_4$ column does not have a carry, then the only time we would add $n$ before $\alpha_4$ is if $\alpha_4 \alpha_5 = 00$, in which case we would implement the $(10-11)$ case procedure and add $n$ to $a_{i+1}$ at $\alpha_3$. However, if $\alpha_4 \alpha_5 = 00$ and there was not a carry from the $\alpha_4$ column, then $z_2, z_3, n_2, n_3 = 0$ as well. This is a contradiction, because if $z_2$ and $z_3$ are both zero, then $y_i 0010100 z_4$ is a longer $f$-prefix of $a_i$ than $x_i$. Lastly, we consider what happens if the $\alpha_4$ column addition does result in a carry. In this case, we have that $a_{i+1}$ has the $f$-prefix $y_i 01000 \alpha_4$, which is sufficient to satisfy the $f$-carry property. We once again invoke the property of the (10) case, and note that we would only add $n$ into $a_{i+1}$ at the $\alpha_5$ column or later, which satisfies the minimum distance between indicies where we add $n$ that is required to keep $b_i \in \texttt{FIB}$.

Therefore, in all cases, we can build the Sequence (2.6) and find arbitrarily close approximations to $n$.

□

Furthermore, we get the following corollary:

**Corollary 43.** *For each real number*

$$r \in \bigcup_{i=-\infty}^{\infty} \left( \frac{3}{4} \cdot 2^i, \frac{4}{3} \cdot 2^i \right) = R'$$

*and $\epsilon > 0$, there exists $a/b \in R(\texttt{FIB})$ such that $|r - a/b| < \epsilon$.*

*Proof.* Let $r \in R'$ be the real number we wish to approximate. Choose $i$ such that $2^{-i+1} < \epsilon$. Let $r' = \lfloor r \cdot 2^i \rfloor$. We note that $\left| r - \frac{r'}{2^i} \right| < \frac{\epsilon}{2}$.

If $r' = 0$ then $0 < r < k^{-i} < \epsilon$. We have that $2^{-i} \in R(\texttt{FIB})$, as $1, 2^i \in \texttt{FIB}$ and $2^{-i} = 1/2^i$. We then observe that $|r - 2^{-i}| < \epsilon$, as required. Therefore, we can assume without loss of generality that $r' > 0$.

We note that

$$r' \in \mathbb{N} \cap \bigcup_{j=0}^{\infty} \left( \frac{3}{4} \cdot 2^j, \frac{4}{3} \cdot 2^j \right),$$

given the restrictions on $r$. We can now invoke Theorem 42, and get an approximation $a/b \in R(\texttt{FIB})$ such that $|r' - a/b| \leq \frac{\epsilon}{2}$. We note that $b \cdot 2^i \in \texttt{FIB}$ if and only if $b \in \texttt{FIB}$. Therefore, we have that

$$\left| r - \frac{a}{b \cdot 2^i} \right| \leq \left| \frac{r'}{2^i} + \frac{\epsilon}{2} - \frac{a}{b \cdot 2^i} \right| \leq \epsilon$$

and $a/(b \cdot 2^i) \in R(\texttt{FIB})$. □

## 2.4 Open Problems

We conclude this chapter by briefly noting several open problems related to the content of this chapter.

1. Develop a heuristic search for finding accepting inputs, but not necessarily shortest inputs, more efficiently in $M(k, R(A, B), p/q)$.
2. Characterize $\mathbb{N} \cap R(A_{k,i})$ for $k \geq 3$ and $2 \leq i \leq k - 1$
3. Characterize $\mathbb{Q}^+ \cap R(A_{k,i})$ for $k \geq 3$ and $2 \leq i \leq k - 1$.
4. Characterize $\mathbb{Q}^+ \cap R(\texttt{FIB}, \mathbb{N})$.

5. Prove or disprove Conjecture 40.
6. Characterize $\mathbb{Q}^+ \cap R(\texttt{FIB})$.
7. Prove or disprove Conjecture 41.

# Chapter 3

# Ratio Sets of Palindromes and Antipalindromes

In this chapter, we examine some ratio sets that we are not able to utilize the `REGULAR RATIO SET MEMBERSHIP` algorithm to compute. We say that a word $x$ is a palindrome if $x = x^R$. If the word $x \in \Sigma_k$ for some $k \geq 2$, then we say that $x$ is an antipalindrome if $x = \overline{x}^R$. Recall that $\overline{x}$ is the complement, where $\overline{\sigma} = k - 1 - \sigma$ for $\sigma \in \Sigma_k$. We can extend this notion to a natural number $n \in \mathbb{N}$ by examining the base-$k$ representation of $n$. We say that a natural number $n$ is a *palindrome in base* $k$, or simply base-$k$ palindrome, if

$$\langle n \rangle_k = (\langle n \rangle_k)^R.$$

We also note that the base-$k$ representation of a palindrome in base $k$ must not end in zero, since the leading digit of a base-$k$ representation is always non-zero. We say that a natural number $n$ is an *antipalindrome in base* $k$, or simply base-$k$ antipalindrome, if

$$\langle n \rangle_k = \overline{(\langle n \rangle_k)^R}.$$

Analogously, the base-$k$ representation of an antipalindrome in base $k$ must not end with the symbol $k-1$, since the leading digit is also always non-zero. We note that it is impossible to determine if word is a palindrome or antipalindrome using a finite automaton. Therefore, it is also impossible to determine if a natural number $n$ is a palindrome in base $k$ or an antipalindrome in base $k$ using a finite automaton.

We begin with Section 3.1, wherein we compare several different algorithms for determining the smallest multiple of a natural number $n$ that is either a palindrome or

antipalindrome in base $k$. Section 3.2 focuses on reformulating the automata developed in Section 2.1 to decide membership in ratio sets entirely comprised of sets of palindromes in base $k$ or sets of antipalindromes in base $k$.

## 3.1 Palindromic and Antipalindromic Multiples

The results and algorithms in this section are based on my joint paper in preparation [8].

### 3.1.1 Palindromes

We denote the set of palindromes in base $k$ by $\texttt{PAL}_k \subseteq \mathbb{N}$. For example, the set

$$\texttt{PAL}_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44, 55, 66, 77, 88, 99, 101, \ldots\}$$

is the set of palindromes in base 10 and is sequence A002113 in the OEIS. Similarly, the set

$$\begin{aligned}\texttt{PAL}_2 = \{&0 = [\epsilon]_2, 1 = [1]_2, 3 = [11]_2, 5 = [101]_2, 7 = [111]_2, 9 = [1001]_2, 15 = [1111]_2,\\ &17 = [10001]_2, 21 = [10101]_2, 27 = [11011]_2, \ldots\}\end{aligned}$$

is the set of palindromes in base 2 and is sequence A006995 in the OEIS. Let $\varphi$ be the Euler phi-function, where $\varphi(n) = |\{m \mid m \in \mathbb{N}, 0 < m < n, \gcd(m, n) = 1\}|$. We now state a result of Jeffrey Shallit [8]:

**Theorem 44.** *Let $n \geq 1$ and $k \geq 2$. Then there exists $b \geq 1$ such that $bn$ is a palindrome in base $k$ if and only if $k \nmid n$.*

*Proof.* One direction is easy: if $k \mid n$, then $\langle n \rangle_k$ ends in a 0 and hence a multiple cannot be a palindrome.

For the other direction, assume $k \nmid n$, and let $S$ be the set of primes that divide both $k$ and $n$:

$$S = \{p \ : \ p \text{ prime and } p \mid k \text{ and } p \mid n\}.$$

For $j \in \mathbb{N}$ and prime $p$, let $\nu_p(j)$ be the largest power of $p$ that divides $j$.

Split $n = ms$ into a product of parts not relatively prime to $k$ and relatively prime to $k$, as follows:

$$m := \prod_{p \in S} p^{\nu_p(n)}$$
$$s := \prod_{q \notin S} q^{\nu_q(n)}.$$

We note that $\gcd(s, k) = 1$.

Let $i$ be the smallest natural number such that $m \mid k^i$. Write $x = \langle m \rangle_k$. Since $m \leq k^i$, we have $|x| \leq i$ except if $m = k^i$. In this latter case $k \mid n$, a contradiction. So $|x| \leq i$.

Let $b = \lceil \frac{i+|x|}{\varphi(s)} \rceil$, where $\varphi$ is the Euler phi-function. Note that $b\varphi(s) \geq i + |x|$. Set $\ell = b\varphi(s) - 2|x|$; then from the previous paragraph we have $\ell \geq i - |x| \geq 0$. Define $y = x^R(0^\ell)x$, so that $|y| = b\varphi(s)$. Clearly $y$ is a palindrome, and since $k \nmid m$, the string $x$ does not end in a 0 and so $x^R$ does not begin with a 0. Set $u = [y]_k$. Then $u = h \cdot k^{\ell+|x|} + m \equiv 0 \pmod{m}$, where $h = [x^R]_k$.

Now let $t = \frac{k^{bs\varphi(s)}-1}{k^{b\varphi(s)}-1}$. Since

$$t = k^{(s-1)b\varphi(s)} + \cdots + k^{2b\varphi(s)} + k^{b\varphi(s)} + 1,$$

the base-$k$ representation of $t$ is $(10^{b\varphi(s)-1})^{s-1}1$. Since $\gcd(k, s) = 1$, by Euler's theorem we know that $k^{\varphi(s)} \equiv 1 \pmod{s}$. Hence $k^{i\varphi(s)} \equiv 1 \pmod{s}$ for all $i \geq 0$ and hence $t \equiv 0 \pmod{s}$.

Finally, consider $tu$. We have $\langle tu \rangle_k = y^s$. This is a palindrome since $y$ is a palindrome. On the other hand, $t \equiv 0 \pmod{s}$ and $u \equiv 0 \pmod{m}$. So $tu \equiv 0 \pmod{n}$, since $n = ms$.

□

We can use Theorem 44 to acquire a bound on $\underline{b}(n)$ for $R(\mathtt{PAL}_k, \mathbb{N})$:

**Corollary 45.** *Let $n \geq 1$ and $k \geq 2$ be such that $k \nmid n$. Let $b \geq 1$ be the smallest natural number such that $\langle bn \rangle_k$ is a palindrome. Then $b < k^{n^2+2n\log_2(n)-n+1}$.*

*Proof.* Let $tu$ be constructed as in the previous proof. Since $\langle tu \rangle_k = y^s$, we have that $|\langle tu \rangle_k| = |y| \cdot s$. Note that $y = x^R(0^\ell)x$ and $\ell = b\varphi(s) - 2|x|$. Hence

$$|y| = \ell + 2|x| = b\varphi(s) - 2|x| + 2|x| = b\varphi(s).$$

Additionally, we have that

$$b = \left\lceil \frac{i + |x|}{\varphi(s)} \right\rceil \leq \frac{i + |x|}{\varphi(s)} + 1.$$

Multiplying both sides by $\varphi(s)$ shows us that $b\varphi(s) \leq i + |x| + \varphi(s) \leq 2i + \varphi(s)$ since $|x| \leq i$.

An alternative characterization of $i$ is that $i$ is the largest $\left\lceil \frac{\nu_p(m)}{\nu_p(k)} \right\rceil$ for $p \in S$. The largest possible case for $i$ is when $\nu_p(k) = 1$ and $\nu_p(m)$ is maximized. Note that $\nu_p(m)$ is maximized when $p = 2$ and $\nu_2(m) = \log_2(m)$ so we have that $i \leq \log_2(m)$.

Finally, $\varphi(s) \leq s - 1$, $m \leq n$, and $s \leq n$. Therefore,

$$\begin{aligned} |\langle tu \rangle_k| &= |y| \cdot s \\ &= b\varphi(s) \cdot s \\ &\leq (2i + \varphi(s)) \cdot s \\ &\leq (2\log_2(m) + s - 1) \cdot s \\ &\leq 2n\log_2(n) + n^2 - n. \end{aligned}$$

Hence $tu < k^{n^2 + 2n\log_2(n) - n + 1}$. □

When $\gcd(k, n) = 1$, we have a significantly better bound:

**Corollary 46.** *Let $n \geq 1$ and $k \geq 2$ be such that $\gcd(k, n) = 1$. Let $b \geq 1$ be the smallest natural number such that $\langle bn \rangle_k$ is a palindrome. Then $b \leq \frac{k^{\varphi(n)} - 1}{n}$, where $\varphi$ is the Euler phi-function.*

*Proof.* If $b = \frac{k^{\varphi(n)} - 1}{n}$, then $b \cdot n = k^{\varphi(n)} - 1$, which is a palindrome in base $k$. We have that $b \in \mathbb{N}$ from Euler's theorem, since

$$k^{\varphi(n)} - 1 \equiv 1 - 1 \equiv 0 \pmod{n}.$$

□

### 3.1.2 Palindrome Algorithms

We briefly present four algorithms for determining inclusion in $R(\mathtt{PAL}_2, \mathbb{N})$, and compare their effectiveness. To simplify our time complexity analysis, we assume that $n \cdot m$ and $n/m$ have time complexity $O((\log n)(\log m))$. Let $b = \underline{b}(n)$.

**Algorithm 1-P**

The simplest algorithm is to search for $b$ by brute force. We simply enumerate each natural number $m$ in ascending order and see if $\langle n \cdot m \rangle_2$ is a palindrome. This takes $O(b \cdot (\log b) \cdot (\log n))$ time.

**Algorithm 2-P**

An alternative method is to instead enumerate base-2 palindromes $m$, and check if $n \mid m$. Once we find such an $m$, we get that $b = m/n$. There are $O(\sqrt{2^{\log(b \cdot n)}})$ palindromes smaller than $b \cdot n$, so this algorithm takes $O(\sqrt{2^{\log(b \cdot n)}} \cdot (\log b) \cdot (\log n)) = O(\sqrt{n \cdot b} \cdot (\log b) \cdot (\log n))$ time.

**Algorithm 3-P**

A more complicated solution is a dynamic programming approach. We repeat the following procedure for $\ell \geq \lfloor \log n \rfloor$, until we find $b$:

Let

$$s_i = 2^{\ell-1-i} + 2^i$$

for $0 \leq i \leq \lfloor \ell/2 \rfloor - 1$. If $\ell$ is odd, then let

$$s_i = 2^{\ell-1-i}$$

for $i = \lfloor \ell/2 \rfloor$. Let $j = \lfloor (\ell/2) - 1 \rfloor$. For every subset $S \subseteq s_1, s_2, \ldots, s_j$,

$$s_0 + \sum_{s \in S} s$$

is a palindrome in base 2, with base-2 representation of length $\ell$. Let

$$s_S \equiv s_0 + \sum_{s \in S} s \pmod{n}.$$

If we can find a subset $S$ such that

$$s_S \equiv 0 \pmod{n},$$

then we have found a multiple of $n$ that is a palindrome in base 2.

To find the desired $S$, we begin with a list $L$ containing only $s_\emptyset$. For each $1 \leq i \leq j$ in descending order, we compute

$$s_{S\cup\{i\}} \equiv s_i + s_S \pmod{n}$$

for each $s_S$ in $L$. If $s_{S\cup\{i\}}$ is not already in the list $L$, then we append the natural number $s_{S\cup\{i\}}$ to the list and note that $S \cup \{i\}$ is the set that corresponds to the new addition to the list. If $s_{S\cup\{i\}} \equiv 0 \pmod{n}$, then we return

$$b = \frac{s_0 + \sum_{s\in S\cup\{i\}} s}{n}.$$

If we go through all of $1 \leq i \leq j$ without finding an $s_S$ equivalent to 0 modulo $n$, then we repeat the process for $\ell + 1$.

Each iteration takes $O(j \cdot n)$ time, since we can at most have $n$ natural numbers $0 \leq s_S < n$ in the list $L$. We must iterate until $\ell = \lfloor \log(b \cdot n) \rfloor$, so finding $b$ with this algorithm takes $O(n \cdot \log(b \cdot n))$ time.

**Algorithm 4-P**

The last approach is based on automata. Consider the automaton $M = (Q, \Sigma_2, \delta, q_0, F)$ where

- $Q = \{q_0\} \cup \{(i, j) \mid 0 \leq i < n, 0 \leq j < n\}$,
- $\delta((i, j), \sigma) = (i', j')$ for $i \equiv 2 \cdot i + (j + 1) \cdot \sigma \pmod{n}$ and $j' \equiv 4 \cdot j \pmod{n}$, and
- $F = \{(0, j) \mid 0 \leq j < n\}$.

The component $i$ of each state pair $(i, j)$ represents that the input word $x$ that lead to this state is such that $[x^R x]_2 \equiv i \pmod{n}$. We take everything modulo $n$ because we only want it to be some multiple of $n$, and it puts a limit on the number of states. Note that this means that each new symbol being read is appended to each side of the word $x^R x$. We use $j$ to track how much a new 1 symbol being appended to the front of $x^R x$ affects the string when interpreted as a number in base 2. We need to account for the unknown middle symbol $\sigma$, so we use the special start state $q_0$ to jump to each possibility. We have an $\epsilon$ transition from $q_0$ to each of $(0, 2), (0, 4)$, and $(1, 4)$ which corresponds to middle symbol $\epsilon$, 0, and 1 respectively.

| $i$ | Runtime of Algorithm $i$-P | Optimal range $b = \underline{b}(n)$ |
|---|---|---|
| 1 | $O(b \cdot (\log b) \cdot (\log n))$ | $b \leq n$ |
| 2 | $O(\sqrt{n \cdot b} \cdot (\log b) \cdot (\log n))$ | $n \leq b \leq n^3$ |
| 3 | $O(n \cdot \log(b \cdot n))$ | $n^3 \leq b \leq n \cdot 2^n$ |
| 4 | $O(n^2)$ | $n \cdot 2^n \leq b$ |

Table 3.1: Comparison of optimal algorithm to find $\underline{b}(n)$ for $R(\texttt{PAL}_2, \mathbb{N})$ by the ratio of $\underline{b}(n)$ to $n$.

| $i$ | $N_i \cap [1, 10000000)$ | $\lvert N_i \cap [1, 10000000)\rvert$ |
|---|---|---|
| 1 | $\{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 25, 27, 29, 33, 35, 37, \ldots\}$ | 4999526 |
| 2 | $\{23, 31, 49, 59, 63, 69, 71, 81, 83, 85, 87, 93, 123, 125, 127, \ldots\}$ | 427 |
| 3 | $\{47, 179, 329, 379, 453, 573, 1095, 3813, 5223, 5461, 178101, \ldots\}$ | 33 |
| 4 | $\{3937, 11811, 14329, 19685, 23715, 27559, 40005, 43307, \ldots\}$ | 14 |

Table 3.2: Comparison of effectiveness of algorithms that find $\underline{b}(n)$ for $R(\texttt{PAL}_2, \mathbb{N})$.

We find our base-2 palindrome $b = [x^R \sigma x]_2$ that is a multiple of $n$ by doing a breadth first search through $M$. We do need that the leading digit of $x^R \sigma x$ is non-zero, so we search for an accepting state of $M$ that we reached by taking an edge labeled with 1. This takes $O(n^2)$ time since there are $O(n^2)$ states in $M$.

### Combined Algorithm and Comparison

The Table 3.1 presents the values of $b$ in terms of $n$ where each algorithm is asymptotically the fastest. Since we do not know the size of $b$ for a given $n$, each algorithm presented in this subsection could be advantageous. Therefore, our full algorithm runs all 4 of the presented algorithms in parallel. Once one algorithm terminates, we returns the answer it computed and note which algorithm finished first. I implemented these algorithms in Python. The data examined in this subsection and code to generate it is available on GitHub. Let $N_i$ be the set of natural numbers $n$ where Algorithm $i$-P was the fastest algorithm to find $\underline{b}(n)$. We compare $N_i$ for $i \in \{1, 2, 3, 4\}$ in Table 3.2. The vast majority of $n \leq 10000000$ use Algorithm 1-P. However, the remaining algorithms are still beneficial in some circumstances.

### 3.1.3 Antipalindromes

We denote the set of antipalindromes in base $k$ by $\texttt{APAL}_k \subseteq \mathbb{N}$. If $n$ is an antipalindrome and has odd length in base $k$, then $\sigma = \overline{\sigma}$ where $\sigma$ is the middle symbol of the base-$k$ representation of $n$. We note that $\sigma = \overline{\sigma}$ if and only if $\sigma = \frac{k-1}{2}$. This means that $\sigma = \overline{\sigma}$ can only occur when $k$ is odd, as when $k$ is even $\frac{k-1}{2}$ is not a natural number. Therefore, all antipalindromes in even bases have representations that are of even length. For example, the set

$$\texttt{APAL}_{10} = \{0, 18, 27, 36, 45, 54, 63, 72, 81, 90, 1098, 1188, 1278, \ldots\}$$

is the set of antipalindromes in base 10. Note that 0 is an antipalindrome in base $k$ for all $k$, because $\langle 0 \rangle_k = \epsilon$. The set

$$\begin{aligned}\texttt{APAL}_3 = \{&0 = [\epsilon]_3, 1 = [1]_3, 4 = [11]_3, 6 = [20]_3, 13 = [111]_3, 21 = [210]_3, 34 = [1021]_3,\\ &40 = [1111]_3, 46 = [1201]_3, 60 = [2020]_3, 66 = [2110]_3, 72 = [2200]_3, 97 = [10121]_3\}\end{aligned}$$

is the set of antipalindromes in base 3. This set contains natural numbers that have base-$k$ representations of odd length. In this subsection, we mainly examine $R(\texttt{APAL}_2, \mathbb{N})$. However, we first prove a general result about $R(\texttt{APAL}_k, \mathbb{N})$.

**Theorem 47.** *Let $n \geq 1$ and $k \geq 1$. Then there exists $b \geq 1$ such that $\langle bn \rangle_k$ is an antipalindrome.*

*Proof.* Split $n = ms$ as in previous theorem. We once again have $\gcd(s, k) = 1$.

Let $i$ be the smallest natural number such that $m \mid k^i$. Let $b = \lceil \frac{i}{\varphi(s)} \rceil$, where $\varphi$ is the Euler phi-function. Note that $b\varphi(s) \geq i$. Define $y = (\langle k-1 \rangle_k)^{b\varphi(s)} 0^{b\varphi(s)}$. This $y$ has that $|y| = 2b\varphi(s)$. Set $u = [y]_k$. Then $u = (k^{b\varphi(s)} - 1) \cdot k^{b\varphi(s)} \equiv 0 \pmod{m}$ since $m \mid k^{b\varphi(s)}$.

Now let $t = \frac{k^{2bs\varphi(s)}-1}{k^{2b\varphi(s)}-1}$. Since

$$t = k^{2(s-1)b\varphi(s)} + \cdots + k^{4b\varphi(s)} + k^{2b\varphi(s)} + 1,$$

the base-$k$ representation of $t$ is $1(0^{2b\varphi(s)-1})^{s-1}1$. Since $\gcd(k, s) = 1$, by Euler's theorem we know that $k^{\varphi(s)} \equiv 1 \pmod{s}$. Hence $k^{i\varphi(s)} \equiv 1 \pmod{s}$ for all $i \geq 0$ and hence $t \equiv 0 \pmod{s}$.

Finally, consider $tu$. We have $\langle tu \rangle_k = y^s$. This is an antipalindrome since $y$ is an antipalindrome. On the other hand, $t \equiv 0 \pmod{s}$ and $u \equiv 0 \pmod{m}$. So $tu \equiv 0 \pmod{n}$, since $n = ms$. □

### 3.1.4 Antipalindrome Algorithms

We present four algorithms analogous to the four algorithms in Subsection 3.1.2.

#### Algorithm 1-A

Algorithm 1-A is completely identical to Algorithm 1-P, except for the obvious change that we check if $n \cdot m$ is an antipalindrome in base 2.

#### Algorithm 2-A

Similarly, Algorithm 2-A is identical to Algorithm 2-P, except that we list antipalindromes, of which there are the asymptotically the same number compared as palindromes.

#### Algorithm 3-A

Algorithm 3-A is the most significantly modified algorithm compared to palindromes. We still preform the algorithm in iterations where we choose a length $\ell$. Antipalindromes in base 2 have binary representations of even length, so we only consider even $\ell$.

Instead of each step potentially adding $s_i = 2^{\ell-1-i} + 2^i$ for $0 \leq i \leq \lfloor \ell/2 \rfloor - 1$, we have to decide between adding $2^{\ell-1-i}$ or adding $2^i$. An antipalindrome has exactly one of those two powers of two for each $i$. This is a bit different, but can reduce this problem back to a standard subset sum by adjusting the goal value $g$. For antipalindromes, we define $s_i = 2^{\ell-1-i} - 2^i$ for $0 \leq i \leq (\ell/2) - 1$. We define $s_S$ for $S \subseteq \{s_1, s_2, \ldots, s_j\}$ analogously to the palindrome case. For palindromes, we wanted our subset sum to be equivalent to zero modulo $n$. For antipalindromes, our goal is to find $S$ such that

$$s_S \equiv g \equiv \sum_{i=0}^{(\ell/2)} 2^i \pmod{n}.$$

If we have such an $S$, then we can derive an antipalindrome in base 2 that is a multiple of $n$. If $s_i$, then we have implicitly added $2^{\ell-1-i}$ since the $-2^i$ part of $s_i$ cancelling out the $2^i$ in $g$. If we exclude $s_i$, then $g$ implicitly includes the $2^i$ that is not cancelled out.

This algorithm takes the same asymptotic quantity of iterations and the same amount of time on each iteration. Therefore, Algorithm 3-A also achieves the same asymptotic runtime bound as Algorithm 3-P.

| $i$ | $N_i' \cap [1, 2727120)$ | $\lvert N_i' \cap [1, 2727120)\rvert$ |
|---|---|---|
| 1 | $\{3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, \ldots\}$ | 1631979 |
| 2 | $\{45, 85, 90, 120, 135, 173, 176, 180, 191, 225, 235, \ldots\}$ | 111350 |
| 3 | $\{189, 198, 297, 360, 378, 387, 396, 409, 424, 429, 453, \ldots\}$ | 983772 |
| 4 | $\{99, 2993, 8750, 21705, 25226, 356075, 589815, 907729, \ldots\}$ | 17 |

Table 3.3: Comparison of algorithms that find $\underline{b}(n)$ for $R(\texttt{APAL}_2, \mathbb{N})$.

### Algorithm 4-A

The construction of $M$ for Algorithm 4-A is even easier in comparison to Algorithm 4-P, though largely identical. We have no valid middle symbol, so we have the unique starting state $q_0 = (0, 2)$, which corresponds to the central symbol $\epsilon$ starting point for palindromes. The only additional change is that the transition function $\delta$ is instead defined as

$$\delta((i, j)\sigma) = (i', j')$$

for $i' \equiv 2 \cdot i + \sigma \cdot j + (1 - \sigma) \pmod{n}$ and $j' \equiv 4 \cdot j \pmod{n}$.

Since the set of states is the same, we end up with the same asymptotic runtime bounds once again.

### Combined Algorithm and Comparison

We note that Table 3.1 is also correct for the antipalindrome algorithms. Similarly to palindromes, we can combine all four algorithms into a single algorithm by running them in parallel. As before, once one algorithm terminates, we returns the answer it computed and note which algorithm finished first. I also implemented these algorithms in Python. The data used in this subsection and code to generate it is available on GitHub. Let $N_i'$ be the set of natural numbers $n$ where Algorithm $i$-A was the fastest algorithm to find $\underline{b}(n)$. We compare $N_i'$ for $i \in \{1, 2, 3, 4\}$ in Table 3.3. In comparison to palindromes, far fewer $n \leq 2727120$ use Algorithm 1-A. Algorithm 1-A is the fastest algorithm for 59.84% of $n \leq 2727120$. This difference is still striking compared to palindromes in binary where 99.99% of odd $n \leq 10000000$ used Algorithm 1-P. However, this is not unexpected, as there are many more palindromes in binary compared to antipalindromes in binary since palindromes in binary have less restrictions on the middle symbol. Antipalindromes in base $k$ for even $k$ are also all of even length, which further restricts the possibilities for $\underline{b}(n)$.

## 3.2 Ratio Sets of Palindromes and Antipalindromes

The results in this section are based on results from my joint paper [3]. Some of the text is taken almost verbatim or verbatim.

In this section, we build a specialized version of the `REGULAR RATIO SET MEMBERSHIP` algorithm designed specifically for deciding inclusion in $R(\texttt{PAL}_k)$. Similarly to Section 2.1, we begin by building an automaton for a given natural number $n$ and base $k$ that accepts some encoding of natural numbers $a$ and $b$ that are palindromes in base $k$. Afterwards, we extend the construction of the automata to rational numbers $p/q$. We then further extend the automaton construction that verifies $p/q \in R(A, B)$ to accommodate $A = \texttt{APAL}_k$ or $B = \texttt{APAL}_k$. We denote the automaton we construct for a given $n$ by $M(k, A, B, n)$ where $A, B \in \{\texttt{PAL}_k, \texttt{APAL}_k\}$.

Since the base-$k$ representations of $\texttt{PAL}_k$ and $\texttt{APAL}_k$ are not regular languages, the automata $M(k, A, B, p/q)$ that we construct here are significantly different from the usual `REGULAR RATIO SET MEMBERSHIP` automata. Recall the carry equation (2.2) and rational carry equation (2.5) from Section 2.1. Our new automaton $M(k, A, B, n)$ still implements the carry equations but needs several modifications in order to ensure that the carry equations are being satisfied by palindromes. The main difference in comparison to multiplication automata is that $M(k, A, B, n)$ checks the carry equations from the least significant digit carry equation and the most significant digit carry equation simultaneously and then moving inward from both directions at once. In this sense, $M(k, A, B, n)$ is simultaneously simulating a least-significant-digit-first multiplication automaton and a most-significant-digit-first multiplication automaton. We begin the path to the construction of $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ by defining an input encoding for palindromes in base $k$ that the automata designed in this section can understand.

### 3.2.1 Input Encoding

There are two main challenges regarding the input specification when trying to design an automaton that ensures that the inputs are palindromes. The first challenge is, of course, that it is impossible to recognize a palindrome with a finite automaton. To remedy this issue we take, as input, *half* of a palindrome and implicitly determine the other half. We build a new encoding $\langle \alpha, \beta \rangle$ of base-$k$ palindromes $a$ and $b$. We note that the $\alpha$ and $\beta$ in this encoding are words and the encoding $\langle \alpha, \beta \rangle$ in this section is distinct from the encoding $\langle a, b \rangle_k$ defined in previous sections. A naive approach is to interpret the input pair $\langle \alpha, \beta \rangle$ as referring to $a = [\alpha\alpha^R]$ and $b = [\beta\beta^R]$ and to the multiplication $[\alpha\alpha^R]_k = [\beta\beta^R]_k \cdot n$.

This approach has the property that all even-length palindromes in base $k$ have an associated encoding that is a valid input to our automaton. However, this does not cover the case of odd-length palindromes. Therefore, on input $\langle \alpha, \beta \rangle$, the automaton we construct simultaneously checks each multiplication $[\alpha\sigma_\alpha\alpha^R]_k = [\beta\sigma_\beta\beta^R]_k \cdot n$ where $\sigma_a, \sigma_b \in \{\epsilon\} \cup \Sigma_k$. If any of the multiplications are valid, then the automaton accepts the input. We accomplish this using nondeterminism, which is a major distinction between the automata constructed in this section and the automata constructed in Section 2.1. This makes discussions of the relationship between $\langle \alpha, \beta \rangle$ and $a$ and $b$ difficult, as each $a$ and $b$ pair has a specific associated encoding $\langle \alpha, \beta \rangle$ but that encoding does not refer to a unique $a$ and $b$. For this reason, we primarily discuss $\alpha$ and $\beta$ and leave $a$ and $b$ largely ambiguous while we discuss the construction of $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$. However, we can still talk about the specific carry equations where a given $\alpha_\ell$ appears. The only concern is around the middle indices that we have to implicitly guess. We handle this with nondeterminism, which we describe in the next subsections.

Let $a$ and $b$ be palindromes in base $k$ such that $b \cdot n = a$. The second challenge is that the strings $\langle a \rangle_k$ and $\langle b \rangle_k$ have, in general, different lengths. Furthermore, as noted in Subsection 2.1.2, the difference in length between them could be either the floor or the ceiling of $\log_k n$. This is a more serious concern for $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ since it must also keep track of the alignment of $\langle a \rangle_k$ and $\langle b \rangle_k$ that it is implicitly constructing off of $\alpha$ and $\beta$. To accommodate both possibilities of $\log_k n$, the automaton $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ begins by nondeterministically guessing the difference in length between $\langle a \rangle_k$ and $\langle b \rangle_k$. We assume $|\langle a \rangle_k| > |\langle b \rangle_k|$ and discuss the $\langle a \rangle_k = \langle b \rangle_k$ case briefly later, as it skips a few steps described in the next subsection. Given our assumption, it follows that $|\alpha| \geq |\beta|$. It is possible that there is a satisfying $\alpha$ and $\beta$ where $|\alpha| = |\beta|$ even if $\langle a \rangle_k \neq \langle b \rangle_k$. However, in general we need to pad $\beta$ to provide it as input to the automaton simultaneously with $\alpha$. Previously, we padded the input with zeroes when there was a mismatch in size between two input natural numbers. Here, we use $X$ as a padding character to indicate the end of input for $\beta$. A unique character is necessary, as palindromes in base $k$ require a more complicated alignment procedure compared to multiplication automata. The automata in this section need to know when they have examined the entirety of the input $\beta$ so they can use them with the correct symbols of $\alpha$, whereas standard multiplication automata can simply read zeros and process them without having to know if the representation of $b$ has been completely input.

Let $\alpha = \alpha_0\alpha_1 \cdots \alpha_i$ and $\beta = \beta_0\beta_1 \cdots \beta_j$ be words in the language $\Sigma_k^*$. We formally define the encoding $\langle \alpha, \beta \rangle = (\alpha_0, \beta_0)(\alpha_1, \beta_1) \cdots (\alpha_i, \beta_i)$ where $\beta_\ell = X$ for $j < \ell \leq i$. This means that $\langle \alpha, \beta \rangle$ is a word over the alphabet $\Sigma_k \times (\Sigma_k \cup \{X\})$. The automaton $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ immediately rejects the input if $\alpha_0 = 0$ or $\beta_0 = 0$. An input that had

a zero as either component of the first symbol would be a representation that was not canonical, which could create a false positive acceptance. We do however, permit the first symbol to be $(\alpha_0, X)$ for $\alpha_0 \in \Sigma_k$, as this is the situation where $|\langle b \rangle_k| = 1$. As well, the automaton $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ rejects any input not of the form $xy$ where $x \in (\Sigma_k \times \Sigma_k)^*$ and $y \in (\Sigma_k \times \{X\})^*$. We do not require $x$ nor $y$ to be non-empty. Every $\langle \alpha, \beta \rangle$ is of valid form $xy$, which means that every accepted input is some encoding $\langle \alpha, \beta \rangle$. It is meaningful to consider the input $\langle \alpha, \beta \rangle$ as being split into the first component $x$ and second component $y$, as the automaton acts very differently between the two stages of input. We describe the processing of each component $x$ and $y$ in the next two subsections respectively.

### 3.2.2 The First Component of the Input

Let $\langle \alpha, \beta \rangle = (\alpha_0, \beta_0)(\alpha_1, \beta_1) \cdots (\alpha_i, \beta_i)$. Note $\beta = \beta_0 \beta_1 \cdots \beta_j$ for some $j \leq i$ and $\beta_\ell = X$ for $j < \ell \leq i$. Let $\langle a \rangle_k = a_{i'} a_{i'-1} \cdots a_0 = \alpha \sigma_\alpha \alpha^R$ for some $\sigma_\alpha \in \Sigma_k$. Let $\langle b \rangle_k = b_{j'} b_{j'-1} \cdots b_0 = \beta \sigma_\beta \beta^R$ for some $\sigma_\beta \in \Sigma_k$. We define $b_\ell = 0$ for $j' < \ell \leq i'$ as the usual padding for the carry equations. In this subsection, we describe the process of $M(k, \texttt{PAL}_k, \texttt{PAL}_k, p/q)$ reading the component $x \in (\Sigma_k \times \Sigma_k)^*$ of the input $\langle a, b \rangle = xy$. Since the input $\langle \alpha, \beta \rangle$ is providing digits on the most significant side and least significant side of $a$ and $b$ simultaneously, we need to simultaneously check two carry equations after each input symbol.

The automaton is able to directly check the equations and compute the carries for the least signifiant digit carry equations. Since $a_0 = \alpha_0$ and $b_0 = \beta_0$, the first input symbol $(\alpha_0, \beta_0)$ has all the information required to check

$$a_0 \equiv n \cdot b_0 + c_0 \pmod{k}$$

and then compute the first carry equation

$$n \cdot b_0 + c_0 = c_1 \cdot k + a_0.$$

The carry $c_0 = 0$ as usual and afterwards, the automaton saves the carry $c_1$ for the next equation. On receiving each input $(\alpha_\ell, \beta_\ell) = (a_\ell, b_\ell)$ for $\ell \leq j$, the automaton is able to check the equation

$$a_\ell \equiv n \cdot b_\ell + c_\ell \pmod{k}$$

and compute the carry equation

$$n \cdot b_\ell + c_\ell = c_{\ell+1} \cdot k + a_\ell.$$

This is exactly the process implemented by the least-significant-digit-first base-$k$ multiplication automata in Subsection 2.1.3. The only information that $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ must

preserve between states in order to verify these equations is the current value of the carry. Since we need to simultaneously check multiple carry equations, we call this saved carry the *right carry* to disambiguate the multiple values that the automaton saves after each input.

The most significant digit carry equations require more careful handling to compute. We begin with the equations that include $b_\ell$ where $\ell > j'$. Since $\langle a \rangle_k$ is a palindrome and there is a difference in length between it and $\langle b \rangle_k$, we have that $a_{i'} = \alpha_0$ and $b_{i'} = 0$. Therefore, the first input symbol $(\alpha_0, \beta_0) = (a_{i'}, b_{j'})$. As defined in the extended long multiplication algorithm, the carry $c_{i'+1} = 0$. Hence, the automaton can compute the carry equation

$$c_{i'} = c_{i'+1} \cdot k + a_{i'} - n \cdot b_{i'} = 0 \cdot k + a_{i'} - n \cdot 0 = a_{i'}.$$

This carry $c_{i'}$ we denote as the *left carry*. After the next inputs $(\alpha_\ell, \beta_\ell) = (a_{i'-\ell}, b_{j'-\ell})$ for $1 \leq \ell < i' - j'$, the automaton proceeds with calculating $c_{i'-\ell}$ with the equation

$$c_{i'-\ell} = k \cdot c_{i'-\ell+1} + a_{i'-\ell} - n \cdot b_{i'-\ell} = k \cdot c_{i'-\ell+1} + a_{i'-\ell}$$

since $b_{j'-\ell} = 0$. This is exactly the process implemented by the most-significant-digit-first base-$k$ multiplication automata in Subsection 2.1.3. The carry equation using $\alpha_\ell = a_{i'-\ell}$ to compute the left carry is computed concurrently with the corresponding least significant digit carry equation that includes $\alpha_\ell = a_\ell$ to compute the right carry. We call this portion of the input processing the *loading phase*.

Once we reach $\ell = i' - j'$ the handling becomes more complicated. At this step, $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ needs $b_{j'} = \beta_0$ along with $a_{j'} = \alpha_{i'-j'}$ to compute the most significant digit carry equation

$$c_{j'} = k \cdot c_{j'+1} + a_{j'} - n \cdot b_{j'}.$$

In order to compute an equation requiring information contained in different input symbols, the automaton must save some additional information beyond the two carries after each step. After the first input symbol $(\alpha_0, \beta_0)$ is read and the right and left carries are computed, the automaton $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ preserves $\beta_0$ to be used in the computation of $c_{j'}$, which occurs $m$ steps later. After the automaton has computes $c_{j'}$ it discards the saved symbol $\beta_0$, as it is not needed for any other calculations. Similarly, to compute $c_{j'-\ell}$, the automaton needs $\beta_\ell$, which gets preserved after reading input symbol $(\alpha_\ell, \beta_\ell)$ and discarded after computing $c_{j'-\ell}$. We note that at the step where the most significant digit carry equations are computing $c_{j'-\ell}$, the least significant carry equations are computing $c_{\ell+1}$ using input symbol $(\alpha_{\ell+1}, \beta_{\ell+1})$ as usual, since they require no additional information. This process of using, discarding, and then subsequently replacing a saved symbol continues while the input symbols are of the form $(\alpha_\ell, \beta_\ell)$ for $\beta_\ell \neq X$. (This means that this

phase continues until we have seen all of $\beta$.) We call the section of computation where $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ consumes and discards saved symbols while still saving new ones the *shifting phase*.

The number of symbols of $\beta$ that need to be saved at any given time is at most the difference in length between $\langle a \rangle_k$ and $\langle b \rangle_k$. The maximum number of symbols saved is achieved at the end of the loading phase. The number of symbols saved stays constant for the shifting phase. As stated previously, this difference can vary between the floor and ceiling of $\log_k n$. To accommodate both possibilities, $M(k, \texttt{PAL}_2, \texttt{PAL}, n)$ nondeterministically assumes that the difference is a fixed value $m$ and the loading phase saves that many symbols of $\beta$ before starting to consume and replace them in the shifting phase. We call the currently saved section of $\beta$ the *queue of saved symbols* since at each step we read the first-in symbol of the queue for the most significant digit carry equation.

Each state of $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ is therefore identified by the $\leq m$ symbols saved, the number $m$ itself, the left and right carries, and what phase the automaton is in. The automaton also has a special start state, with an $\epsilon$ transition to the two states with no symbols saved, left and right carries set to 0, and each possibility for $m$. For all other states, the automaton has a transition to a new state corresponding to the updated carries and queue of saved symbols for each possible input, as long as the least significant digit carry equation modular equivalence is verified and correct for that input. If the associated equation of the least significant digit carry equations for a given input symbol is not verified, or the most significant carry equation results in a carry larger than $n$, then the transition is omitted. These are the conditions on transitions such that the equations verified and calculated by most-significant-digit-first base-$k$ multiplication automata and least-significant-digit-first base-$k$ multiplication automata are effectuated simultaneously. The loading stage is characterized by having less than $m$ saved symbols and the shifting phase having exactly $m$ saved symbols that it cycles through.

### 3.2.3 The Second Component of the Input

Once $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ has seen all of the input $\beta$, the input symbols change to being of the form $(\alpha_\ell, X)$. We call this final section of processing the *unloading phase*. Any transition with an input of the form $(\alpha_\ell, X)$ pushes the automaton directly into the unloading phase. This can lead to not having $m$ saved symbols in the queue of saved symbols despite having read all of $\beta$. This can occur when $\beta$ is shorter than the difference in length between $a$ and $b$. If this occurs, $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ implicitly pads the front of the queue of saved symbols with enough zeroes to have $m$ saved symbols. At this point the automaton has all

of the digits of $\langle b \rangle_k$, except for the central symbol $\sigma_b$, and has yet to examine the middle section of $\langle a \rangle_k$ that corresponds to the remainder of $\alpha$. This middle section of $\langle a \rangle_k$ lines up with the queue of saved symbols to supply the $\beta$ symbols that are no longer coming from the input. When the automaton reads a symbol $(\alpha_\ell, X)$, we use $\alpha_\ell = a_\ell = a_{i'-\ell}$ for both the most significant digit and least significant digit carry equations as usual.

The automaton must now contend with the possibility that $\langle b \rangle_k$ has odd length and that there may be a symbol of $\langle b \rangle_k$ not given in $\beta$. It nondeterministically decides what the central symbol $\sigma_\beta \in \Sigma_k \cup \{\epsilon\}$ is for $\langle b \rangle_k$. If $\sigma_b \neq \epsilon$, then the automaton proceeds using the input $\alpha_\ell$ as the symbol of $\langle a \rangle_k$ for both the least significant digit and most significant digit carry equations. However, the automaton uses the chosen symbol $\sigma_b$ as the symbol of $\langle b \rangle_k$ for the least significant digit carry equations since we have already processed the entire least significant half of $\langle b \rangle_k$ that corresponds to $\beta^R$. Since the automaton chooses $\sigma_b$ nondeterministically, the states where the choice of $\sigma_b$ must be made have transitions labeled $(\alpha_\ell, X)$ to each state that is the result of some choice of $\sigma_b$. The most significant digit carry equation uses the first-in symbol of the queue of saved symbols as usual, but nothing is added to the queue since there is no new $\beta_\ell$. The left and right carries are updated as usual and the automaton continues with the queue of saved symbols reduced by one. If $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ nondeterministically chose that $\sigma_\beta = \epsilon$, then it skips the step described in this paragraph and proceeds directly with the subsequent steps.

At this point, the automaton $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ consumes both ends of the queue of saved symbols to compute the left carry and right carry for the most significant digit carry and least significant digit equations respectively. The first-in symbol is used along with $\alpha_\ell$ to compute the most significant digit carry equation as usual. However, the least significant carry equation instead takes from the last-in end of the queue since the least significant digit carry equations have already used all of $\beta$ on their side and instead start using the first half of $\langle b \rangle_k$ that was saved in the queue. This proceeds, consuming two symbols from the queue of saved symbols each time. Once the automaton has less than two symbols left, there are two remaining cases. If there are 0 saved symbols remaining and the left and right carries are equal, then the automaton accepts the input. In this case, the entire series of carry equations are satisfied and the input represents a valid $a/b = n$ with $\langle a \rangle_k$ and $\langle b \rangle_k$ palindromes. Alternatively, if the automaton has one saved symbol left, then this is the case where $\langle a \rangle_k$ has an odd number of symbols. If there is an assignment for the middle symbol $\sigma_\alpha$ that results in the carries being equal after computing one final, central, carry equation, then the automaton accepts the input.

### 3.2.4 Extension to Rational Numbers

We can extend the definition of $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ to automata $M(k, \texttt{PAL}_k, \texttt{PAL}_k, p/q)$ that accept an encoding of base-$k$ palindromes $a$ and $b$ such that $a/b = p/q$. We once again assume that $p > q$, as we can use the fact that $p/q = a/b$ if and only if $q/p = b/a$ in order to decide inclusion in the ratio for $p/q$ where $q < p$. We replace the carry equations computed by $M(k, \texttt{PAL}_k, \texttt{PAL}_k, n)$ with rational carry equations. We do this analogously to the modification we presented to multiplication automata in Subsection 2.1.4. We directly defined how the rational carry equations can be implemented for a most-significant-digit-first base-$k$ multiplication automata. We use that process directly for the most significant digit carry equations in $M(k, \texttt{PAL}_k, \texttt{PAL}_k, p/q)$ and this requires the same left carry and queue of saved symbols that are used in the natural number case.

We did not define least-significant-digit-first multiplication automata for rational numbers, so we briefly explain how it works here. We define the unified carry $c_\ell = c_{b,\ell} - c_{a,\ell}$, analogously to how we previously defined the unified carry for the rational multiplication automata. So upon reading input symbol $(\alpha_\ell, \beta_\ell)$, our automaton verifies the equation

$$p \cdot b_\ell + c_\ell \equiv q \cdot a_j \pmod{k}$$

and then computes

$$c_{\ell+1} = \frac{p \cdot b_\ell + c_\ell - q \cdot a_\ell}{k}$$

to ensure that the least significant digit carry equations hold. Therefore, the automaton $M(k, \texttt{PAL}_k, \texttt{PAL}_k, p/q)$ still has a single right carry.

The rational carry equation carries $c_\ell$ have the bound $-q < c_\ell < p$ for the given $p/q$ so the left and right carry each have the same bound $-q < c_\ell < p$. The queue of saved symbols required has length either the floor or ceiling of $\log_k(p/q)$, which can be zero. We still require $p > q$ but $p/q < k$ is permitted. The construction as presented still works in this case, the automaton constructed simply has no loading or unloading phase. It simply remains in the shifting phase until the input is complete, and has acceptance conditions analogous to the loading phase at the point when the queue of saved symbols is depleted.

The Bai et al. paper [3], where the description of $M(k, \texttt{PAL}_k, \texttt{PAL}_k, p/q)$ first appears, has a more complicated description of the automata $M(k, \texttt{PAL}_k, \texttt{PAL}_k, p/q)$. In the paper [3], the automaton $M(k, \texttt{PAL}_k, \texttt{PAL}_k, p/q)$ tracks two distinct right carries in addition to the unified left carry. This resulted in a slightly worse complexity bound than what appears in this thesis.

### 3.2.5 Extension to Antipalindromes

Once we have constructed $M(k, \texttt{PAL}_k, \texttt{PAL}_k, p/q)$, it is relatively simple to generalize the construction to antipalindromes in base $k$. We denote the automaton accepting pairs of base-$k$ antipalindromes $a$ and $b$ such that $a/b = p/q$ as $M(k, \texttt{APAL}_k, \texttt{APAL}_k, p/q)$. If $A = \texttt{APAL}_k$, then the $\alpha$ component of $\langle \alpha, \beta \rangle$ is interpreted as $a = [\alpha \sigma_\alpha \overline{\alpha}^R]_k$ where $\sigma_\alpha \in \{\epsilon, (k-1)/2\} \cap \Sigma_k$. Similarly, if $B = \texttt{APAL}_k$, then the $\beta$ component of $\langle \alpha, \beta \rangle$ is interpreted as $b = [\beta \sigma_\beta \overline{\beta}^R]_k$ where $\sigma_\beta \in \{\epsilon, (k-1)/2\} \cap \Sigma_k$. This mainly changes the handling of the least significant digit carry equations, wherein $\alpha_\ell$ or $\beta_\ell$ is interpreted as the complement. The only other difference is that the nondeterministic choice of central digit of $\langle a \rangle_k$ and $\langle b \rangle_k$ is restricted to $\{\epsilon, (k-1)/2\} \cap \Sigma_k$. Otherwise, the construction is identical and achieves all associated complexity bounds.

### 3.2.6 Implementation and Complexity

I implemented this algorithm in Python. The algorithm and all data computed is available on GitHub. Similarly to the `REGULAR RATIO SET MEMBERSHIP` algorithm, we can perform a graph search on the automaton $M(k, A, B, p/q)$ in order to decide inclusion in $R(A, B)$ for $A, B \in \{\texttt{PAL}_k, \texttt{APAL}_k\}$. The first question in determining the time complexity of our new algorithm is to find a bound on the number of states in $M(k, A, B, p/q)$. As we discussed, each state in $M(k, A, B, p/q)$ needs to track the left and right carries $-q < c_l < p$ and $-q < c_r < p$, as well as the queue of saved states, which is $m$ symbols in $\Sigma_k$ where $m \in \{\lfloor \log_k(p/q) \rfloor, \lceil \log_k(p/q) \rceil\}$. Therefore, tracking the basic information required needs at most

$$(p + q - 1)^2 \cdot k^{\lceil \log_k(p/q) \rceil}$$

states. However, we noted that the automaton must track of which of the three phases it is currently executing. As well, at the beginning of the execution the automaton nondeterministically chooses between $m = \lfloor \log_k(p/q) \rfloor$ and $m = \lceil \log_k(p/q) \rceil$. Hence, the total number of states is

$$|M(k, A, B, p/q)| \leq 3 \cdot 2 \cdot (p + q - 1)^2 \cdot k^{\lceil \log_k(p/q) \rceil} \leq 6 \cdot (p + q - 1)^2 \cdot k \cdot \frac{p}{q} \in O(kp^3/q)$$

as $p > q$.

If we simply want to decide if $p/q \in R(A, B)$ then we can accomplish this with a simple breath first search of $M(k, A, B, p/q)$ for an accepting state, which takes $O((k^2 \cdot (kp^3/q)) + (kp^3/q)) = O(k^3 p^3/q)$ time since each state has at most $|\Sigma_k \times \Sigma_k|$ transitions out of it.

However, given the nondeterministic choices that the automaton can make, this breadth first search is insufficient for finding the smallest $a$ and $b$ such that $a/b = p/q$. Instead, we use a modified Dijkstra's algorithm to find shortest paths, in terms of the implicit $a$ and $b$ instead of simply $\langle \alpha, \beta \rangle$, to each state with some careful processing of the possible middle symbols. We then find the shortest path to any accepting state. Therefore, our implementation for finding the smallest $a$ and $b$ instead takes

$$O(((k^2 \cdot (kp^3/q)) + (kp^3/q)) \cdot \log(kp^3/q)) = O((k^3p^3/q) \cdot \log(kp^3/q))$$

time.

We can put a slightly better bound on the worst case size of the smallest $a$ for a given $p/q$.

**Theorem 48.** *Let $k \geq 2$ be the base and let $A, B \in \{\mathtt{PAL}_k, \mathtt{APAL}_k\}$. If there exists $a$ and $b$ such that $a/b = p/q$, $a \in A$, and $b \in B$, then,*

$$a \leq k^{2 \cdot \left( \lceil \log_k(p/q) \rceil + (p+q-1)^2 \cdot k^{\lceil \log_k(p/q) \rceil} + \left\lceil \frac{\log_k(p/q)}{2} \right\rceil \right) + 1}.$$

*Proof.* We can get this bound through analysis of each phase of processing in the automaton $M(k, A, B, p/q)$ and determining the maximum size of $\alpha$ for input $\langle \alpha, \beta \rangle$.

The algorithm begins by nondeterministically choosing $m \in \{\lfloor \log_k(p/q) \rfloor, \lceil \log_k(p/q) \rceil\}$ and then loading at most $m$ symbols into the queue of saved symbols. The worst case is when $m = \lceil \log_k(p/q) \rceil$. Therefore, the loading phase is at most $m$ symbols $(\alpha_\ell, \beta_\ell)$.

The unloading phase consumes two symbols from the queue of saved symbols each time an input symbol $(\alpha_\ell, X)$ is read. Therefore, we read at most

$$\left\lceil \frac{\lceil \log_k(p/q) \rceil}{2} \right\rceil = \left\lceil \frac{\log_k(p/q)}{2} \right\rceil$$

symbols of $\alpha$ in this phase. However, we also accommodate the possibility of an implicit central digit of $\langle a \rangle_k$ in this phase, so our final count on the digits of $a$ can include one additional digit past the digits in $\alpha$.

It is theoretically possible that the shortest accepting path for any input goes through every single possible state in the shifting phase. This case would mean reading $(p + q - 1)^2 \cdot k^{\lceil \log_k(p/q) \rceil}$ symbols $(\alpha_\ell, \beta_\ell)$.

Taken together, we get that

$$|\alpha| \leq \lceil \log_k(p/q) \rceil + (p + q - 1)^2 \cdot k^{\lceil \log_k(p/q) \rceil} + \left\lceil \frac{\log_k(p/q)}{2} \right\rceil$$

and

$$|\langle a\rangle_k| \leq 2 \cdot \left( \lceil \log_k(p/q) \rceil + (p+q-1)^2 \cdot k^{\lceil \log_k(p/q) \rceil} + \left\lceil \frac{\log_k(p/q)}{2} \right\rceil \right) + 1.$$

□

### 3.2.7 Binary Palindromes

In this subsection, we examine the ratio set $R(\texttt{PAL}_2)$. The natural numbers

$$\mathbb{N} \cap R(\texttt{PAL}_2) = \{0, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 27, \ldots\}$$

form sequence A305468 in the OEIS. We have the following theorem on the $(R)$-density of $R(\texttt{PAL}_2)$.

**Theorem 49.** *The set* $\texttt{PAL}_2$ *is* $(R)$*-dense.*

*Proof.* Let $\texttt{PAL}_2 = \{a_0 < a_1 < \cdots\}$ be the set of palindromes in base 2 listed in increasing order. Consider the base-2 palindrome $\langle a_i\rangle_k = \alpha\sigma_\alpha\alpha^R$. We get that

$$a_i' = [\langle [\alpha]_2 + 1\rangle_k \sigma_\alpha (\langle [\alpha]_2 + 1\rangle_k)^R]_2 \leq a_i + 2^{1+|\sigma_\alpha\alpha^R|}$$

is also a palindrome in base 2. Let $a_{i+1}$ be the smallest palindrome in base 2 larger than $a_i$. We have that

$$\frac{a_{i+1}}{a_i} \leq \frac{a_i + 2^{1+|\sigma_\alpha\alpha^R|}}{a_i} = 1 + \frac{2^{1+|\sigma_\alpha\alpha^R|}}{a_i} \leq 1 + \frac{2^{1+j+|\sigma_\alpha|}}{2^{2\cdot i+|\sigma_\alpha|-1}} = 1 + 2^{1+j+|\sigma_\alpha|-2\cdot j-|\sigma_\alpha|+1} = 1 + 2^{-j+2}$$

where $j = |\alpha|$. However, we get that $j \to \infty$ as $i \to \infty$ since we consider progressively larger palindromes, which in turn have longer base-2 representations. Therefore,

$$\lim_{i\to\infty} \frac{a_{i+1}}{a_i} = 1$$

since $2^{-j+2} \to 0$ as $i \to \infty$. This means we can use Theorem 5, which gives us that $\texttt{PAL}_2$ has the property $(S)$ described in Section 1.2 and $\texttt{PAL}_2$ is $(R)$-dense. □

We note that since $\texttt{APAL}_2$ only contains even numbers and $\texttt{PAL}_2$ only contains odd numbers, the ratio sets $R(\texttt{PAL}_2, \texttt{APAL}_2)$ and $R(\texttt{APAL}_2, \texttt{PAL}_2)$ contain no natural numbers. However, we have the following corollary:

**Corollary 50.** *The ratio sets* $R(\texttt{PAL}_2, \texttt{APAL}_2)$ *and* $R(\texttt{APAL}_2, \texttt{PAL}_2)$ *are dense in* $\mathbb{R}^+$.

*Proof.* We have from Theorem 49 that $\texttt{PAL}_2$ has the property $(S)$. Theorem 6 gives us that $R(\texttt{PAL}_2, B)$ is dense in $\mathbb{R}^+$ for all infinite sets $B \subseteq \mathbb{N}$. Therefore $R(\texttt{PAL}_2, \texttt{APAL}_2)$ is dense in $\mathbb{R}^+$. This gives us that the ratio set $R(\texttt{APAL}_2, \texttt{PAL}_2)$ is also dense in $\mathbb{R}^+$ since $R(A, B)$ is dense in $\mathbb{R}^+$ if and only if $R(B, A)$ is dense in $\mathbb{R}^+$. □

A natural question is to first ask how many natural numbers are in $R(\texttt{PAL}_2)$. Table 3.4 contains the amount of natural numbers with binary representation of each length $i$ that are in $R(\texttt{PAL}_2)$. The trend appears to be exponential but we have no proof of this beyond the numerical data we computed with the algorithm presented in this section. We also examined the $n \in R(\texttt{PAL}_2)$ that set new records for $\underline{b}(n)$. We include the data in Table 3.5. We also examined the rational numbers $p/q \in R(\texttt{PAL}_2)$. These were difficult to explore, as $\underline{a}(p/q)$ and $\underline{b}(p/q)$ were frequently extremely large. For example, the smallest base-2 palindromes $a$ and $b$ such that $a/b = 979/765$ are

$$a = 435964577851526887677597179561025269848009167916543881959761365529045212378773108135544954987$$

and

$$b = 340666907105636434191380840004274087266319727738668099794910566526782009672892163149838499045.$$

The data did suggest the following conjecture:

**Conjecture 51.** For all odd numbers $p > 1$, $p \neq 23$, there exists an odd number $q < p$ such that there exists base-2 palindromes $a$ and $b$ where $p/q = a/b$.

We experimentally showed that this conjecture holds up for $p < 1000$ but further computation for this conjecture is computationally expensive.

The ratio set $R(\texttt{PAL}_2)$ is further examined by Bai et al. [3], which lead to a few more notable conclusions:

**Theorem 52.**

1. *There are* $\Omega(2i/2)$ $i$*-bit natural numbers representable as the quotient of palindromic numbers. [3, Thm. 6]*
2. *The lower asymptotic density of natural numbers not contained in* $R(\texttt{PAL}_2)$ *is* $\underline{d}(\mathbb{N} \setminus R(\texttt{PAL}_2)) > 1/40$. *[3, Cor. 11]*
3. *If there exists base-2 palindromes* $a$ *and* $b$ *such that* $n = a/b$*, then there are infinitely many pairs of base-2 palindromes* $a'$ *and* $b'$ *such that* $n = a'/b'$. *[3, Thm. 13]*

| $i$ | $\lvert\mathbb{N} \cap R(\mathtt{PAL}_2) \cap [2^{i-1}, 2^i)\rvert$ |
|---|---|
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | 4 |
| 5 | 5 |
| 6 | 10 |
| 7 | 17 |
| 8 | 33 |
| 9 | 55 |
| 10 | 98 |
| 11 | 165 |
| 12 | 309 |
| 13 | 571 |
| 14 | 985 |

Table 3.4: Number of $i$-bit natural numbers in the ratio set of palindromes in base 2.

| $N$ | $A$ | $B$ |
|---|---|---|
| 1 | 1 | 1 |
| 11 | 33 | 3 |
| 13 | 65 | 5 |
| 19 | 513 | 27 |
| 53 | 3339 | 63 |
| 71 | 54315 | 765 |
| 79 | 888987 | 11253 |
| 149 | 5887437 | 39513 |
| 319 | 224725611 | 704469 |
| 575 | 147606740625 | 256707375 |
| 1823 | 394070635302093 | 216166009491 |
| 2597 | 96342506397593044197 | 37097615093412801 |
| 5155 | 324903223321029232798074465 | 63026813447338357477803 |
| 10627 | 930075382452907131236047024606890 3 | 875200322247960036921094405389 |
| 22331 | 7937744489597569305570866473462312986756397 5 | 3554585325152285748766677029001080554725 |

Table 3.5: Natural numbers $n \in R(\mathtt{PAL}_2)$ such that $\underline{b}(n) > \underline{b}(n')$ for $n' < n$.

| $i$ | $\lvert\mathbb{N} \cap R(\texttt{APAL}_2) \cap [2^{i-1}, 2^i)\rvert$ |
|---|---|
| 1 | 1 |
| 2 | 0 |
| 3 | 2 |
| 4 | 1 |
| 5 | 8 |
| 6 | 4 |
| 7 | 24 |
| 8 | 17 |
| 9 | 75 |
| 10 | 50 |
| 11 | 247 |
| 12 | 165 |
| 13 | 903 |

Table 3.6: The amount of natural numbers with binary representation of length $i$ in $R(\texttt{PAL}_2)$.

### 3.2.8 Binary Antipalindromes

In this subsection, we investigate the ratio set $R(\texttt{APAL}_2)$. The natural numbers

$$\mathbb{N} \cap R(\texttt{APAL}_2) = \{0, 1, 5, 6, 15, 17, 18, 19, 20, 21, \ldots\}$$

form sequence A351172 in the OEIS Table 3.6 contains the amount of natural numbers of binary representation of each length $i$ that are elements of $R(\texttt{PAL}_2)$. Similarly to $\texttt{PAL}_2$, this trend appears to be exponential but we also have no proof of this pattern beyond the numerical results. We also examined the $n \in R(\texttt{APAL}_2)$ that set new records for $\underline{b}(n)$. We include the data in Table 3.7. We note that the numbers in Table 3.7 are even larger than in Table 3.5.

**Theorem 53.** *There are infinitely many natural numbers in* $R(\texttt{APAL}_2)$.

*Proof.* Natural numbers of the form $n = 2^{2i+1} - 2^i = [1^{i+1}0^i]_2$ are elements of $R(\texttt{APAL}_2)$. We have that $a/b = n$ for $a = 2^{2i+2} - 2^{i+1} = [1^{i+1}0^{i+1}]_2$ and $b = 2 = [10]_2$, which are both antipalindromes. □

**Theorem 54.** *There are infinitely many natural numbers* $n \in \mathbb{N}$ *such that there are infinitely many base-2 antipalindromes* $a$ *and* $b$ *where* $n = a/b$.

| $N$ | $A$ | $B$ |
|---|---|---|
| 5 | 10 | 2 |
| 15 | 150 | 10 |
| 18 | 936 | 52 |
| 59 | 52140188 | 883732 |
| 66 | 65099232 | 986352 |
| 83 | 206712630902722 | 2490513625334 |
| 343 | 841469573210301602 | 2453264061837614 |
| 835 | 180616526119856633856230 | 216307216910007944738 |
| 991 | 2004287797608707007280062973725 50 | 2022490209494154396851728530 50 |
| 1268 | 7554776151776056927908760805826890 4 | 595802535629026571601637287525 78 |
| 1290 | 43959239407961251665818031144043012938376675325 40 | 340769297736133733843550629023589247584315312 6 |
| 1952 | 158668199276265902297399644779200695547126001790447385315654 4 | 81284938153824745029405555727049536653240779605761980182 2 |
| 4091 | 10223272491989051875528852806818198915974054413770448081896281 6 | 24989666321166100893495118080709359364395146452628814670976 |
| 4460 | 38898710433553477152076407181322465555429871822889997891243000 0 | 87216839537115419623489702200274586447152178975089681370500 |
| 4640 | 85112365674283227507265261996365447811182320230460498 83220363630941564530051147747252794193043200 | 18343182257388626617945099568182208579996189704840624 74831974920461544079752402531735515989880 |
| 4848 | 16307148112492799707206815760673202828585190069605262924 53647949068964670350753495389037684933163550313918407048323128697 6 | 33636856667683167712885346040992580091966151133674222204 07689663921131745773006384878926915208985880840329704424 1590612 |
| 5840 | 43493875233140378950672024766781801439773086758844870 87734362685028948495519190020221259652712554180379503 20370614437165579602331 20 | 74475813755377361216904151997914043561255285545967244 65298566241487925506026010308598047350535195514348464 389907781458314719218 |
| 6624 | 33301854653004018709445764603598238453897624842252171 14065184395426890419279201949902950413217648251363098 49496826284424060000589698848419903839832345753230313 68320074437113766562324271243044637263162663379437241 6 | 50274539029293506505805804051325843076536269387457987 83310966780535764521858698595867980696282681538893566 56849073497016998793160777247010724395882164482533686 11594315273420541307856689678509416158156194715334 |

Table 3.7: Natural numbers $n \in R(\texttt{APAL}_2)$ such that $\underline{b}(n) > \underline{b}(n')$ for $n' < n$.

*Proof.* Let $n = 2^{2\ell+1} - 2^\ell$ for some $\ell \geq 1$. Define

$$b_i = [1(0^{\ell+2}1^{\ell+2})^i 0]_2 = 2^{2\ell i+4i+1} + (2^{\ell+2}-1) \cdot \sum_{j=0}^{i-1} 2^{2\ell j+4j+1}$$

for $i \geq 0$. Clearly $b_i$ is an antipalindrome in base 2. We now compute $a_i = n \cdot b_i$.

$$\begin{aligned}
n \cdot b_i &= (2^{2\ell+1} - 2^\ell) \cdot \left( 2^{2\ell i+4i+1} + (2^{\ell+2}-1) \cdot \sum_{j=0}^{i-1} 2^{2\ell j+4j+1} \right) \\
&= (2^{2\ell i+4i+2\ell+2} - 2^{2\ell i+4i+\ell+1}) + (2^{3\ell+3} - 2^{2\ell+2} - 2^{\ell+1} + 2^\ell) \cdot \left( \sum_{j=0}^{i-1} 2^{2\ell j+4j+1} \right) \\
&= (2^{2\ell i+4i+2\ell+2} - 2^{2\ell i+4i+\ell+1}) + (2^{2\ell+3} - 2^{\ell+2} - 2^{\ell+1} + 1) \cdot 2^\ell \cdot \left( \sum_{j=0}^{i-1} 2^{2\ell j+4j+1} \right) \\
&= (2^{2\ell i+4i+2\ell+2} - 2^{2\ell i+4i+\ell+1}) + (2^{2\ell+3} - 2^{\ell+2} - 2^{\ell+1} + 1) \cdot \left( \sum_{j=0}^{i-1} 2^{2\ell j+4j+\ell+1} \right) \\
&= [1^{\ell+1}0^{2\ell i+4i+\ell+1}]_2 + [1^\ell 010^\ell 1]_2 \cdot [(10^{2\ell+3})^{i-1}10^{\ell+1}]_2 \\
&= [1^{\ell+1}0^{2\ell i+4i+\ell+1}]_2 + [(1^\ell 010^\ell 10)^{i-1}1^\ell 010^\ell 10^{\ell+1}]_2 \\
&= [1^{\ell+1}0(1^\ell 010^\ell 10)^{i-1}1^\ell 010^\ell 10^{\ell+1}]_2 \\
&= [1^{\ell+1}(01^\ell 010^\ell 1)^i 0^{\ell+1}]_2 \\
&= a_i.
\end{aligned}$$

Thus $a_i$ is also an antipalindrome in base 2 for each $i \geq 0$. Therefore, we have an infinite set of base-2 antipalindromes $a_i$ and $b_i$ such that $a_i/b_i = n$ for each $n = 2^{2\ell+1} - 2^\ell$. □

**Theorem 55.** *There are exactly $2^{i-1}$ pairs of base-2 antipalindromes $a$ and $b$ such that $n = a/b$ for $n = 4^i + 1$.*

*Proof.* Let $n = 4^i + 1 = [10^{2i-1}1]_2$. Consider a base-2 antipalindrome $b$. Let $\langle b \rangle_k = \beta = \beta_1\beta_2\cdots\beta_\ell$ and note $|\beta| = \ell$.

If $\beta$ has length $\ell < 2i$, then $\langle bn \rangle_k = \langle a \rangle_k = \beta 0^{2i-\ell}\beta$. Since antipalindromes in base 2 have binary representations of even length, if the center of $\langle a \rangle_k$ is at least two zeros, then $a$ is not an antipalindrome in base 2. Therefore, our $a$ here is not an antipalindrome in base 2.

If $\beta$ has length $\ell = 2i$, then $\langle bn \rangle_k = \langle a \rangle_k = \beta\beta$. Here, $a$ is an antipalindrome since $\overline{(\beta\beta)^R} = \overline{\beta}^R\overline{\beta}^R = \beta\beta$.

If $\beta$ has length $\ell > 2i$, then $\langle bn \rangle_k$ can be viewed as the binary addition of $[\beta 0^{2i}]_2 + [\beta]_2$. Since $\beta$ was sufficiently long, there is some non-trivial overlap in the addition. Let $j = 2i - \ell$. The overlap has length $\ell - j$ and there are $j$ symbols of $\beta$ on each side of the overlap.

$$\begin{array}{cc|c|c} & \beta_1\beta_2\cdots\beta_j & \beta_{j+1}\beta_{j+2}\cdots\beta_\ell & 0^j \\ + & 0^j & \beta_1\beta_2\cdots\beta_{\ell-j} & \beta\ell - j + 1\beta_{\ell-j+2}\cdots\beta_\ell \end{array}$$

Figure 3.1: Piecewise addition of $[\beta 0^{2i}]_2 + [\beta]_2$.

Since $b$ is an antipalindrome in base 2, we get that $\beta_1\beta_2\cdots\beta_j = \overline{\beta_{\ell-j+1}\beta_{\ell-j+1}\cdots\beta_\ell}^R$. This means that for $[\beta 0^{2i}]_2 + [\beta]_2$ to be an antipalindrome the overlap region must not overflow to the left. We have additional information that further constrains this addition. We know that $\beta_1 = 1$, which implies that $\beta_\ell = \overline{\beta_1} = 0$. Additionally, we know that the overlap region cannot overflow so $\beta_{j+1} = 0$, which subsequently implies that $\beta_{\ell-j} = \overline{\beta_{j+1}} = 1$. As well, the remaining addition $[\beta j + 2\beta j + 3\cdots\beta_{\ell-1}]_2 + [\beta_2\beta_3\cdots\beta_{\ell-j-1}]_2$ must not overflow either.

$$\begin{array}{cc|c|c|c|c} & \beta_1\beta_2\cdots\beta_j & 0 & \beta_{j+2}\beta_{j+3}\cdots\beta_{\ell-1} & 0 & 0^j \\ + & 0^j & 1 & \beta_2\beta_3\cdots\beta_{\ell-j-1} & 1 & \beta_{\ell-j+1}\beta\ell - j + 2\cdots\beta_\ell \\ \hline = & \beta_1\beta_2\cdots\beta_j & 1 & \beta' & 1 & \beta_{\ell-j+1}\beta_{\ell+j+1}\cdots\beta_\ell \end{array}$$

Figure 3.2: Piecewise addition of $[\beta 0^{2i}]_2 + [\beta]_2$ with constraints.

From the result of the addition we see that we have a 1 at $j+1$ symbols from the front and a 1 at $j+1$ symbols from the back. Therefore, this cannot be an antipalindrome in base 2.

Overall, given a base-2 antipalindrome $b$, we have that $bn$ is an antipalindrome in base 2 if and only if $\langle b \rangle_k$ has length $2i$. There are $2^{i-1}$ antipalindromes in base 2 of length $2i$, so for $n = 4^i + 1$ there are exactly $2^{i-1}$ solutions to $n = a/b$ for base-2 antipalindromes $a$ and $b$. □

As with $R(\texttt{PAL}_2)$ before, the ratio set $R(\texttt{APAL}_2)$ is further examined by Bai et al. [3]:

**Theorem 56.**

1. *The set* $\texttt{APAL}_2$ *is* $(R)$*-dense. [3, Thm. 14]*

2. *The lower asymptotic density of natural numbers not contained in* $R(\texttt{APAL}_2)$ *is* $\underline{d}(\mathbb{N} \setminus R(\texttt{APAL}_2)) > 1/60$*. [3, Thm. 20]*

As well, examining rational numbers $p/q \in R(\texttt{APAL}_2)$ lead to a similar conjecture to palindromes:

**Conjecture 57.** For all $p \geq 4$ there exists $q < p$ such that $p/q = A/B$ has a solution in antipalindromes.

The smallest base-$k$ antipalindromes $\underline{a}(p/q)$ and $\underline{b}(p/q)$ are generally even larger than for palindromes in base $k$. For example, the smallest antipalindromes in binary $a$ and $b$ such that $a/b = 960/527$ are $a =$

123488335521399097520446714068347599479933500362668242775693013065831705778455411015978753726653857443627332547988390098721673963103239979035640547077917392804795250182028753800174169116477800361082899344465944556084111470545477090239447028941702755740595022368518275171007572436760482385904809838780735014863686241818215607795947411080913498008442825679592833678865846036391335428845975712764583827139150178213891564696471882542693026228872977592848186347465518430085971658311548426349712629617061002461937088916568789455331781860009277363002444938372376426409349549969820438753161560915890436797199051312068851515357512387981254460480906980717773805815538001443554183199390918213670460282463422656834514445716194836822256690771708798244010820952165632924869863611983142620371328966957512364597567158981492432747694245025717455343991855418326593897404081449362927535384737555977627483829984300836874384257902399933566997414686571563690971632075913517295268137127611381422913678220794954727600533534516312312331038829749723349859042215544591191317981860065085279274232029170938239774166430904765407576433808705730785028275096490777190553086332250642184307631986194351365337324601401527659584251780426995592541451396343086191791838699791485099128013340230974422429588804353687565086020814966547965068536407399756886018154816109644210404200564689981839524385856174094456288001040420056468998183952438585617409445628800

and $b =$

$$
\begin{aligned}
&6778995085393471290966189407710331763117182780325642077373981029759719\\
&6817964585005646670014527690492491254429989459981277418935995216113491\\
&4401753229817354251323925478428679715539449212331258232194666193057841\\
&4693367369268486086099602977526278890862009747582105117814075103195226\\
&3306476428994567747340992534544426498124609696316964207959805677551426\\
&1803598159882940633970606601781269054173197246634399293165820008902031\\
&6737718749919252355839499107395229699409188818261152492727710488156099\\
&5633532446143167547769824741711416509416900926219064883835960669142414\\
&2991800355160116905376485444523543667957292098544632797848010713188761\\
&4653483122795652791215082138204245109848549897281104617975922731639599\\
&1446992596286123963884662538219309036035106918532592241048352211994912\\
&6676413441308193843918155394716492151167271196532589094780898788622973\\
&5220310826244887897319042827891322083355175414416846514690916719157767\\
&1630197716289103982514651189635525006691265214904444011664593620321273\\
&2905636890057095548855172797900598575813585472663700495749995394006004\\
&5859822910643491695768029630454269344696542851020081314290408346219781\\
&3516511082895230704684475092115760543809087940801596635484311046954792\\
&6048836302361221555675894508400240357281195730340075421489898976286673\\
&1290968738999306958368017654934455999074863197882487388704957092685676\\
&9669805934991271280655574318962237269233 10.
\end{aligned}
$$

### 3.2.9 Ternary Palindromes and Antipalindromes

While the majority of our investigation concerns $\texttt{PAL}_2$ and $\texttt{APAL}_2$, our approach is functional in any base $k$. To give a brief example of the extensibility to other bases and future examination possible, we compare the ratio sets $R(A, B)$ for $A, B \in \{\texttt{PAL}_3, \texttt{APAL}_3\}$. In base 3 we have that $\mathbb{N} \cap R(\texttt{PAL}_3, \texttt{APAL}_3) \neq \emptyset$. This is in contrast to $R(\texttt{PAL}_2, \texttt{APAL}_2)$, which contains no natural numbers. As well, the set $\texttt{APAL}_3$ has antipalindromes in base 3 $n$ with base-3 representations $\langle n \rangle_3$ of odd length since the middle symbol can be non-empty in an antipalindrome in an odd base. Table 3.8 contains the number of natural numbers with base-3 representations of length $i$ in the ratio sets $R(A, B)$ for $A, B \in \{\texttt{PAL}_3, \texttt{APAL}_3\}$.

| $i$ | $\lvert\mathbb{N} \cap R(\mathtt{PAL}_3, \mathtt{PAL}_3) \cap [3^{i-1}, 3^i)\rvert$ | $\lvert\mathbb{N} \cap R(\mathtt{PAL}_3, \mathtt{APAL}_3) \cap [3^{i-1}, 3^i)\rvert$ |
|---|---|---|
| 0 | 2 | 2 |
| 1 | 4 | 4 |
| 2 | 11 | 9 |
| 3 | 25 | 23 |
| 4 | 67 | 62 |
| 5 | 176 | 129 |
| 6 | 450 | 348 |
| 7 | 1072 | 848 |

| $i$ | $\lvert\mathbb{N} \cap R(\mathtt{APAL}_3, \mathtt{PAL}_3) \cap [3^{i-1}, 3^i)\rvert$ | $\lvert\mathbb{N} \cap R(\mathtt{APAL}_3, \mathtt{APAL}_3) \cap [3^{i-1}, 3^i)\rvert$ |
|---|---|---|
| 0 | 2 | 1 |
| 1 | 5 | 3 |
| 2 | 16 | 9 |
| 3 | 40 | 21 |
| 4 | 108 | 58 |
| 5 | 240 | 122 |
| 6 | 682 | 368 |
| 7 | 1637 | 803 |

Table 3.8: The amount of natural numbers with base-3 representation of length $i$ in the ratio sets of palindromes in base 3 and antipalindromes in base 3.

## 3.3 Open Problems

We conclude by briefly noting several open problems related to the content of this chapter.

1. Derive sub-exponential bounds on $\underline{b}(p/q)$ for $R(\mathtt{PAL}_k, \mathbb{N})$ and $R(\mathtt{APAL}_k, \mathbb{N})$, or prove that none exist.
2. Improve the bound in Theorem 48.
3. Design multiplication automata analogous to $M(k, \mathtt{PAL}_k, \mathtt{PAL}_k, p/q)$ for squares, words $x$ that can be written as $x = yy$ for some word $y$, or antisquares, words $x$ that can be written as $x = y\overline{y}$ for some word $y$.
4. Design multiplication automata analogous to $M(k, \mathtt{PAL}_k, \mathtt{PAL}_k, p/q)$ for a context-sensitive, but not context-free, language.
5. Prove or disprove Conjecture 51.
6. Prove or disprove $\underline{d}(\mathbb{N} \cap R(\mathtt{APAL}_2)) > 0$.
7. Prove or disprove Conjecture 57.
8. Examine $\underline{d}(\mathbb{N} \cap R(A, B))$ and $\underline{d}(\mathbb{N} \setminus R(A, B))$ for $k \geq 3$ and $A, B \in \{\mathtt{PAL}_k, \mathtt{APAL}_k\}$.

# Bibliography


[1] D. Antony and R. Barman. $p$-adic quotient sets: cubic forms. *Arch. Math. (Basel)* 118.2 (2022), 143–149.

[2] D. Antony, R. Barman, and J. Chattopadhyay. On denseness of certain direction and generalized direction sets. ArXiv preprint, available at https://arxiv.org/abs/2206.00413. 2022.

[3] J. H. Bai, J. Meleshko, S. Riasat, and J. Shallit. Quotients of palindromic and antipalindromic numbers. *Integers* 22 (2022).

[4] B. Brown, M. Dairyko, S. R. Garcia, B. Lutz, and M. Someck. Four quotient set gems. *Amer. Math. Monthly* 121 (2014), 590–599.

[5] J. Bukor, P. Erdős, T. Šalát, and J. T. Tóth. Remarks on the $(R)$-density of sets of numbers. II. *Math. Slovaca* 47 (1997), 517–526.

[6] H. V. Chu. On sets with more products than quotients. *Rocky Mountain J. Math.* 50.2 (2020), 499–512.

[7] J. Cilleruelo and J. Guijarro-Ordóñez. Ratio sets of random sets. *Ramanujan J.* 43.2 (2015), 327–345.

[8] T. Clokie, J. Meleshko, and J. Shallit. Palindromic multiples. In preparation.

[9] C. Donnay, S. R. Garcia, and J. Rouse. $p$-adic quotient sets II: quadratic forms. *J. Number Theory* 201 (2019), 23–39.

[10] P. Erdős and E. Szemerédi. "On sums and products of integers". In: *Studies in Pure Mathematics*. Birkhäuser Basel, 1983, pp. 213–218.

[11] F. Filip, L. Mišík, and J. T. Tóth. Dispersion of ratio block sequences and asymptotic density. *Acta Arith.* 131.2 (2008), 183–191.

[12] S. R. Garcia, Y. X. Hong, F. Luca, E. Pinsker, C. Sanna, E. Schechter, and A. Starr. $p$-adic quotient sets. *Acta Arith.* 179.2 (2017), 163–184.

[13] S. R. Garcia and F. Luca. Quotients of Fibonacci numbers. *Amer. Math. Monthly* 123.10 (2016), 1039–1044.

[14] S. R. Garcia, D. E. Poore, V. Selhorst-Jones, and N. Simon. Quotient sets and Diophantine equations. *Amer. Math. Monthly* 118 (2011), 704–711.

[15] K. Hare. *private communication (July 2022).*

[16] S. Hedman and D. Rose. Light subsets of $\mathbb{N}$ with dense quotient sets. *Amer. Math. Monthly* 116.7 (2009), 635–641.

[17] C. G. Lekkerkerker. Voorstelling van natuurlijke getallen door een som van getallen van Fibonacci. *Simon Stevin* 29 (1952), 190–195.

[18] P. Leonetti and C. Sanna. Directions sets: a generalization of ratio sets. *Bull. Aust. Math. Soc.* 101.3 (2020), 389–395.

[19] J. H. Loxton and A. J. van der Poorten. An awful problem about integers in base four. *Acta Arith.* 49 (1987), 193–203.

[20] L. Mišík. Sets of positive integers with prescribed values of densities. *Math. Slovaca* 52 (2002), 289–296.

[21] L. Mišík and J. T. Tóth. Logarithmic density of a sequence of integers and density of its ratio set. *J. Théorie Nombres Bordeaux* 15 (2003), 309–318.

[22] P. Miska, N. Murru, and C. Sanna. On the $p$-adic denseness of the quotient set of a polynomial image. *J. Number Theory* 197 (2019), 218–227.

[23] P. Miska and C. Sanna. $p$-adic denseness of members of partitions of $\mathbb{N}$ and their ratio sets. *Bull. Malays. Math. Sci. Soc* 43.2 (2020), 1127–1133.

[24] W. Narkiewicz and T. Šalát. A theorem of H. Steinhaus and $(R)$-dense sets of positive integers. *Czech. Math. J.* 34 (1984), 355–361.

[25] O. Roche-Newton. If $(A+A)/(A+A)$ is small, then the ratio set is large. *J. London Math. Soc.* 93.1 (2016), 83–100.

[26] T. Šalát. On ratio sets of sets of natural numbers. *Acta Arith.* 15 (1969). Corrigendum, **16** (1969), 103, 273–278.

[27] T. Šalát. Remarks on Steinhaus' property and ratio sets of positive integers. *Czech. Math. J.* 50 (2000), 175–183.

[28] C. Sanna. Membership in random ratio sets. *Indag. Math.* (2022). DOI: https://doi.org/10.1016/j.indag.2022.08.007. URL: https://www.sciencedirect.com/science/article/pii/S0019357722000672.

[29] C. Sanna. The quotient set of $k$-generalized Fibonacci numbers is dense in $\mathbb{Q}_p$. *Bull. Aust. Math. Soc.* 96.1 (2017), 24–29.

[30] I. D. Shkredov. Some remarks on sets with small quotient set. *Mat. Sbornik* 208.12 (2017), 1854–1868.

[31] M. Sipser. *Introduction to the Theory of Computation*. Third edition. Boston, MA, USA: Cengage Learning, 2013.

[32] S. C. Sisneros-Thiry. "Combinatorial number theory through diagramming and gesture". PhD Dissertation. University of Illinois at Urbana-Champaign, 2020. URL: https://hdl.handle.net/2142/108500.

[33] O. Strauch and J. T. Tóth. Asymptotic density of $A \subset \mathbb{N}$ and density of the ratio set $R(A)$. *Acta Arith.* 87 (1998). Corrigendum, **103** (2002), 191–200, 67–78.

[34] J. T. Tóth, L. Mišík, and F. Filip. On some properties of dispersion of block sequences of positive integers. *Math. Slovaca* 54.5 (2004), 453–464.

[35] J. T. Tóth and L. Szilinszky. On density of ratio sets of powers of primes. *Nieuw Archief voor Wiskunde* 13 (1995), 205–208.

[36] E. Zeckendorf. Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas. *Bull. Soc. Roy. Liége* 41 (1972), 179–182.