Decision Algorithms for Ostrowski-Automatic Sequences

by

Aseem Raj Baranwal

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, 2020

Abstract

In this thesis, we extend the notion of k-automatic sequences to Ostrowski-automatic sequences. We develop a procedure for computationally deciding certain combinatorial and enumeration questions about such sequences that can be expressed as predicates in first-order logic.

In Chapter 1, we begin with topics and ideas that are preliminary to this work, including a small introduction to non-standard positional numeration systems and the relationship between words and automata. We also provide a brief history of previous work in this area that concerns decidability and automatic theorem-proving. In Chapter 2, we define the theoretical foundations for recognizing addition in a generalized Ostrowski numeration system and formalize the general theory that develops our decision procedure. Next, in Chapter 3, we show how to implement these ideas in practice, and provide the implementation as an integration to the automatic theorem-proving software package – `Walnut`.

Further, we provide some applications of our work in Chapter 4. These applications span several topics in combinatorics on words, including repetitions, pattern-avoidance, critical exponents of special classes of words, properties of Lucas words, and so forth. In Chapter 5, we close with open problems on decidability and higher-order numeration systems, and discuss future directions for research.

Acknowledgements

I thank my supervisor Professor Jeffrey Shallit for his assistance, encouragement and supervision of the research presented in this thesis. For the many insightful discussions that helped shape this thesis, I thank Luke Schaeffer, Hamoon Mousavi, Narad Rampersad and Lucas Mol. I also thank my parents for their everlasting support and encouragement. Finally, I thank Professors Jason Bell and Eric Blais for serving as readers for this thesis.

Dedication

This is dedicated to Joe Petrik, who consistently puts appropriate comic strips from several amazing sources in every email he sends out. Thanks, Joe for keeping us happy! Your emails always bring a smile to me. I wish the best for you.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The goal of this thesis is to develop and explain a procedure for mechanically deciding certain combinatorial and enumerative properties of sequences that are automatic in some numeration system, that is, there exists a finite-state machine that, on input n expressed in that numeration system, computes the n^{th} symbol in the sequence. Our procedure works for properties that can be expressed as predicates in first-order logic, and, in principle, is applicable to any Ostrowski numeration system, which are a generalization of the standard base-k number systems (defined in Section 2.1). We provide the fundamental decidability result in Theorem 1.3. This is not a new result, but we apply it to a broader class of automatic sequences, the Ostrowski-automatic sequences.

The primary contributions of this thesis are Theorems 2.1 and 2.2, where we provide the main result about recognizing Ostrowski addition. This interesting result has led to the application of these decision procedures to several problems in combinatorics on words. For example, it enables us to decide certain combinatorial questions about a broad class of sequences that was not possible computationally before this development. The second contribution of this thesis is Section 3.2, where we provide an algorithm to implement the decision procedure for the case of quadratic irrational numbers, as part of the software package `Walnut`.

The third contribution is Chapter 4, where we discuss four different applications of the developed procedures. First, we resolve a conjecture by Rampersad et al. [46] about the repetition threshold for balanced words, for alphabets of size ≤ 8. Second, we make the first progress on the problem of determining the repetition threshold for infinite palindrome-rich words. Third, we show the existence of an infinite binary word that avoids large antisquares. Fourth, we determine certain properties of Lucas words, which is not a new result, but we show that our procedure can retrieve several such results in a few milliseconds, purely with machine computation.

The implementation, along with the command files required for all proofs and examples in Chapters 3 and 4 are available on GitHub. Some of the text in Chapters 2 and 5 and Sections 4.1 and 4.2 is taken verbatim from previous papers that have contributions by the author [4, 5], and a paper that will be submitted to the journal *Theoretical Computer Science*.

1.1 Preliminaries

We begin with some preliminary definitions and ideas essential to the work in this thesis.

1.1.1 Numeration Systems

A numeration system (or a numeral system) is a method for expressing elements of $\mathbb{N}$ using a consistent set of symbols.

Definition 1.1. A numeration system is a triple $\mathcal{N} = (\Sigma, L, f_{\mathcal{N}})$ consisting of

- a finite alphabet, Σ;
- a language $L \subseteq \Sigma^*$; and
- an onto function $f_{\mathcal{N}} : \Sigma^* \to \mathbb{N}$.

The elements of L are called *representations*, and we say $w \in L$ is a representation for an integer $N \geq 0$ if $f_{\mathcal{N}}(w) = N$. For convenience, we write $f_{\mathcal{N}}(w)$ as $\langle w \rangle_{\mathcal{N}}$, or sometimes as $[w]_{\mathcal{N}}$.

Remark. Every integer $n \in \mathbb{N}$ has at least one representation $\in L$, and can have more than one representation.

Below we list some of the most popular numeration systems in computer science and mathematics.

- Base-b representation for $b \geq 2$ is the most well known. It is defined by the triple $(\Sigma_b, \Sigma_b^*, f_b)$ where
$$\Sigma_b = \{0, 1, \ldots, b-1\}, \text{ and}$$
$$\langle a_{n-1}a_{n-2}\cdots a_0 \rangle_b = \sum_{0 \leq i < n} a_i b^i.$$

For example, in base-3, $\langle 1002\rangle_3 = 29$. This is the most significant digit (MSD) first notation.

- Bijective base-b representation for $b \geq 1$ is the triple $(\{1,\ldots,b\}, \{1,\ldots,b\}^*, f_b)$ where

$$\langle a_{n-1}a_{n-2}\cdots a_0\rangle_b = \sum_{0\leq i<n} a_i b^i.$$

  It is known that $f_b : \{1,\ldots,b\}^* \to \mathbb{N}$ is a bijection [27].

- Fibonacci (Zeckendorf) representation is based on Fibonacci numbers [59]. It is defined by the triple $(\{0,1\}, L_{\mathcal{F}}, f_{\mathcal{F}})$, where $L_{\mathcal{F}} \subset \{0,1\}^*$ is the set of words that do not contain two consecutive 1's, and

$$\langle a_{n-1}a_{n-2}\cdots a_0\rangle_{\mathcal{F}} = \sum_{0\leq i<n} a_i F_{i+2},$$

  where $(F_i)_{i\geq 0}$ is the Fibonacci sequence.

The Zeckendorf representation is a special case of a more general class of numeration systems, called the Ostrowski numeration systems, which we introduce in Section 2.1.

1.1.2 Continued Fraction Expansion

Definition 1.2. We say that a sequence of integers $[d_0; d_1, d_2, \ldots]$ with $d_i \geq 1$ for $i \geq 1$ is the simple continued fraction (or just *continued fraction*) expansion of a real number α if

$$\alpha = d_0 + \cfrac{1}{d_1 + \cfrac{1}{d_2 + \cdots}}.$$

Here are some examples of the continued fraction expansion of real numbers. As usual, an overline or vinculum denotes a repeating block.

- The golden ratio $\varphi = [1; \overline{1}]$.
- $\sqrt{2} = [1; \overline{2}]$.
- The base of the natural logarithm, e has a noticeable pattern in its continued fraction expansion: $e = [2; 1, 2, 1, 1, 4, 1, 1, 6, \ldots]$ (A003417).

- The continued fraction of π has no known simple pattern: $\pi = [3; 7, 15, 1, 292, \dots]$ (A001203).

It is known that a real number α is rational if and only if its continued fraction is finite. Furthermore, α is an irrational root of a quadratic equation, or simply a *quadratic irrational*, if its continued fraction is ultimately periodic. For example,

$$\frac{63 - \sqrt{10}}{107} = 0.55923 \cdots = [0; 1, 1, 3, \overline{1, 2, 1}].$$

Here we call the sequence $0, 1, 1, 3$ the *preperiod*, and $1, 2, 1$ the *period*.

For a real number $\alpha = [d_0; d_1, d_2, \dots]$, we say that $p_i/q_i = [d_0; d_1, d_2, \dots, d_i]$ is a *convergent* of α where the p_i and q_i satisfy the following relations:

$$p_0 = d_0, \quad p_1 = d_1 d_0 + 1, \qquad p_i = d_i p_{i-1} + p_{i-2}, \quad \text{and} \tag{1.1}$$

$$q_0 = 1, \quad q_1 = d_1, \qquad q_i = d_i q_{i-1} + q_{i-2}. \tag{1.2}$$

1.1.3 Automatic Sequences

In this section, we formalize the definition of an automatic sequence and state some characterizations. First, we require some preliminary definitions.

Definition 1.3. A deterministic finite automaton with output or DFAO is defined by a 6-tuple $(Q, \Sigma, \delta, q_0, \Delta, \tau)$, where

- Q is a nonempty set of states,
- Σ is the input alphabet,
- $\delta : Q \times \Sigma \to Q$ is the transition function,
- $q_0 \in Q$ is the initial state,
- Δ is the output alphabet, and
- $\tau : Q \to \Delta$ is the output mapping.

The domain of δ is extended to $Q \times \Sigma^*$ in the usual way, that is, for a word $w = xa$ where $w, x \in \Sigma^*$ and $a \in \Sigma$ we define $\delta(q_0, w) = \delta(\delta(q_0, x), a)$. On input w, the DFAO outputs $\tau(\delta(q_0, w))$. When drawing a DFAO, we label the states q/b to denote that the state q has output $\tau(q) = b$ associated with it. The function $f : \Sigma^* \to \Delta$ is called a *finite-state function* if it is computed by some DFAO $M = (Q, \Sigma, \Delta, \delta, q_0, \tau)$.

Theorem 1.1. *([1], Theorem 4.3.1) Let $M = (Q, \Sigma, \delta, q_0, \Delta, \tau)$ be a DFAO. Then the set*

$$I_d(M) = \{w \in \Sigma^* : \tau(\delta(q_0, w)) = d\}$$

is a regular language for all $d \in \Delta$.

Definition 1.4. A k-automatic sequence is an infinite sequence $\mathbf{a} = (a_n)_{n\geq 0}$ over a finite alphabet such that there exists a DFAO that reaches some state q with output a_n after completely processing the input n expressed in base k [1].

Automatic sequences appear in the literature as early as 1960 by Büchi [14], and in 1969 by Cobham where he referred to them as *uniform tag sequences* [17, 18]. Eilenberg also discussed these sequences in [26], where they are called *k-recognizable*. Since then, there have been several characterizations of these sequences, and researchers have worked on many problems associated with them [1]. In a review of Büchi's paper [35], McNaughton pointed out that Theorem 4 in [14] is incorrect. Specifically, Büchi falsely stated that the system containing $=, +$, and the constant monadic predicate "is a power of 2" forms a first-order arithmetic that is intertranslatable with weak second-order arithmetic. A version of this claim is true, but relies instead on the function $V_k(n)$: the highest power of k dividing n, instead of $v_k(n)$: the exponent of the highest power of k dividing n.

When talking about the automatic sequence corresponding to the DFAO M, the set $I_d(M)$ is sometimes also called a *fiber*. Thus, Theorem 1.1 shows that DFAOs, DFAs and regular languages are closely related, and gives another characterization of automatic sequences. We also note a crucial property of Definition 1.4: the property of being k-automatic does not depend on whether the input is read MSD first or LSD first, due to the following theorem.

Theorem 1.2. *([1, Theorem 4.3.3]) Let $f : \Sigma^* \to \Delta$ be a finite state function. Then so is the function f^R defined by $f^R(w) = f(w^R)$.*

Cobham also described another characterization of k-automatic sequences, as images of fixed points of k-uniform homomorphisms, i.e., morphisms h that map every symbol to a word of k letters. In [16], Christol showed that for a sequence $(a_n)_{n\geq 0}$ over the alphabet Σ, it is k-automatic if and only if the formal power series $\sum_{n\geq 0} g(a_n)X^n$ is algebraic over $GF(k)[X]$ for an injective function $g : \Sigma \to GF(k)$. Following this result, Christol et al. gave another characterization of these sequences for $k = p^n$, a prime power.

Example 1.1. The Thue-Morse sequence $\mathbf{t} = t_0t_1t_2 \cdots = \texttt{01101001} \cdots$, named after Axel Thue [55], is given by the DFAO in Figure 1.1, where t_n is the output associated with the state reached on completely reading n in base 2.

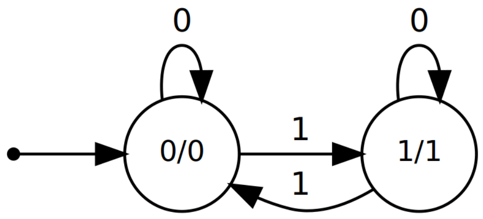

Figure 1.1: DFAO computing the Thue-Morse sequence $\mathtt{t}$.

1.1.4 Decision Procedures

The logical theory $\mathsf{Th}(\mathbb{N}, +)$, called Presburger arithmetic, is decidable [44]. Büchi [14] showed that the resulting theory from adding the function $V_k(n) = k^e$, for some fixed integer $k \geq 2$, where $e = \max\{i : k^i | n\}$ is also decidable. For more details on this topic, see [12, 11, 53]. Therefore, we have the following theorem (see [53]).

Theorem 1.3. *There exists an algorithm that, given a proposition $\mathcal{P}$ phrased using only the universal and existential quantifiers, addition, subtraction, logical operations, comparisons, and indexing into one or more automatic sequences, will decide the truth of $\mathcal{P}$.*

1.2 Previous Work

In this section, we provide a survey of previous work related to decidability and decision procedures for automatic sequences.

1.2.1 Procedures for k-Automatic Sequences

It is known that several questions about an automatic sequence can be answered by performing transformations on the corresponding DFAO. In his master's thesis [50], Schaeffer used a logical theory based on the first-order theory of $\mathbb{N}$, by representing a sequence $w \in \Sigma^\omega$ as a finite set of unary predicates $\{P_a\}_{a\in\Sigma}$, such that $P_a(i)$ is true if $w_i = a$. These predicates enable indexing into the sequence w, and the logical theory is $\mathsf{Th}(\mathbb{N}, \{P_a\}_{a\in\Sigma})$. Thus, if w is automatic in a numeration system then this theory is decidable.

Schaeffer also gave several applications of this result. These applications include computing the supremum of a set of rational numbers described by an automaton, and thus, deciding the critical exponent and related properties of k-automatic words (see [51]). We discuss such properties in more detail for a different class of sequences in Chapter 4.

1.2.2 Procedures for Fibonacci-Automatic Sequences

The notion of k-automatic sequence extends naturally to other numeration systems. We say an infinite sequence $\mathbf{a} = (a_n)_{n \geq 0}$ is $\mathcal{N}$-automatic if it is computable by a DFAO that reads $(n)_{\mathcal{N}}$: the representation of integer n in the numeration system $\mathcal{N}$, and produces output a_n associated with the last state reached.

Example 1.2. The infinite Fibonacci word is given by $\mathbf{f} = h^{\omega}(0) = \texttt{01001010}\cdots$, where the morphism h is defined as follows:

$$h\colon \quad \begin{aligned} 0 &\mapsto 01 \\ 1 &\mapsto 0. \end{aligned}$$

Here, $h^{\omega}(0)$ refers to the fixed point of the morphism h at 0, and is given by $\lim_{n\to\infty} h^n(0)$. The word $\mathbf{f}$ is Fibonacci-automatic, meaning that there exists a DFAO that reads in an integer n in the Fibonacci representation and produces $\mathbf{f}[n]$ as the output associated with the last state reached.

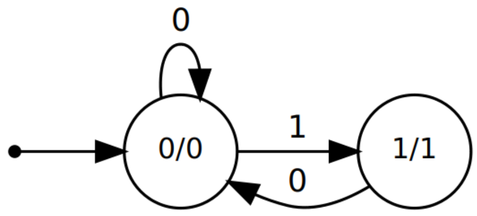

Figure 1.2: DFAO computing the infinite Fibonacci word $\mathbf{f}$.

Note that the automaton only reads canonical (Zeckendorf) representations that do not have two consecutive 1's. In [37], the authors implement a procedure for deciding properties of Fibonacci-automatic words. They also use the procedure to recover and improve several results about the Fibonacci word from the literature, such as the existence of repetitions and patterns, that is, squares, cubes, palindromes, and so on.

We can naturally extend Theorem 1.3 for k-automatic sequences to a broader class, consisting of sequences that are automatic in a generalized numeration system, such that addition in that numeration system can be performed by an automaton. It is known that this is possible for the Fibonacci representation, and the authors in [37] provide an adder automaton for the same. In Chapter 2, we provide an adder framework for the class of Ostrowski numeration systems, which is, in a way, a generalization of the Fibonacci system. Then, in Chapter 3, we implement the adder for the quadratic case.

1.2.3 Applications to Combinatorics on Words

The decision procedures developed for both k-automatic and Fibonacci-automatic sequences have unlocked answers to several combinatorial questions on these sequences. In [23], Du et al. show that there exists an aperiodic infinite binary word avoiding the pattern xxx^R, where x^R denotes the reversal of the word x. This is the first result on pattern avoidance involving a nonuniform morphism to be proved purely by machine computation. In another paper by the same authors [24], they examine enumeration questions such as the number of distinct squares that occur in an infinite word, and abelian properties of Fibonacci-automatic sequences.

With the right set of procedures, a wide variety of combinatorial questions can be answered. For example, using only machine computation, we can comment on the existence of certain kinds of repetitions and patterns in a given automatic sequence, such as palindromes, reversals, squares, overlaps, and so forth. In [52], Schaeffer and Shallit proved some results about closed, palindromic, rich, privileged, trapezoidal, and balanced words in automatic sequences. In [38], Mousavi and Shallit studied properties of the tribonacci word. Currie et al. have recently solved three open problems concerning squarefree arithmetic progressions in infinite words [19]. In [54], the authors proved various results about the largest exponent of a repetition in a circularly considered factor of the Thue-Morse word. They also generalized some previous results in this area. In [39], Ng et al. presented some pattern-avoidance results concerning squares and antisquares. They also proved the existence of an infinite binary word simultaneously avoiding all occurrences of $xh(x)$ for every nonerasing morphism h and all sufficiently large words x. Bonardo, Frid and Shallit [9] showed several recurrence relations and an explicit formula for the number of factorizations of the length-n prefix of the Fibonacci word into a decreasing sequence of standard Fibonacci words. Another interesting application involves the class of balanced words, which are discussed further in Section 4.1. In [46], Rampersad et al. studied the critical exponent of infinite balanced words. They also explored a computational approach to the problem using the procedure developed by Du et al. in [37].

Similar procedures have been applied to problems in number theory as well. In [6], Bell et al. proved some new theorems in additive number theory. Specifically, they showed that every natural number greater than 25 is the sum of at most three natural numbers whose base-2 representation has an equal number of 0's and 1's.

Chapter 2

Theoretical Framework

In this chapter, we provide the formal definition of the fundamental framework that develops the decision procedure using automata theory. An implementation of the developed procedure can then computationally decide certain questions about sequences automatic in the corresponding numeration system. We start with a description of the Ostrowski numeration system.

2.1 Ostrowski Numeration System

Named after the mathematician Alexander Markowich Ostrowski [40], the Ostrowski numeration system is a representation system for integers, based on the continued fraction of an irrational number α. Recall from Section 1.1.2 that the continued fraction of an irrational number is infinite.

Definition 2.1. Let $\alpha = [d_0; d_1, d_2, \ldots]$, and let $(p_i)_{i\geq 0}$ and $(q_i)_{i\geq 0}$ be the sequences denoting respectively, the numerator and denominator of the convergents of α. Then, the *Ostrowski-α numeration system* is defined as a representation system for non-negative integers, such that any $N \geq 0$ can be uniquely represented in MSD-first notation as

$$N = [a_{n-1}a_{n-2}\cdots a_0]_\alpha = \sum_{0\leq i<n} a_i q_i,$$

where

1. $0 \leq a_0 < d_1$,

2. $0 \leq a_i \leq d_{i+1}$, for $i \geq 1$, and
3. for all $i \geq 1$, if $a_i = d_{i+1}$ then $a_{i-1} = 0$.

We call such a unique representation a *canonical* representation of N in the corresponding Ostrowski α-numeration system and write it as $N = [a_{k-1}a_{k-2} \cdots a_0]_\alpha$. We observe that d_0 plays no role in the representation. Therefore, for all practical purposes we can assume $d_0 = 0$, and hence $0 < \alpha < 1$. Furthermore, we observe that if $d_1 = 1$ then $q_0 = q_1 = 1$, implying that for every non-negative integer N, the least significant digit in the representation, a_0, is always 0. Thus, we can further assume that $d_1 \geq 2$, so that $0 < \alpha < 1/2$.

Example 2.1. Consider $\alpha = 1/\phi^2 = [0; 2, \overline{1}]$, where ϕ is the golden ratio. The corresponding Ostrowski numeration system is the Zeckendorf representation that we saw in Section 1.1.1 defined by the Fibonacci sequence $(q_n)_{n \geq 0} = 1, 2, 3, 5, 8, \ldots$. For example, the representation for $N = 23$ is $[1000010]_\alpha$.

Example 2.2. Consider $\beta = 1/\delta_S = \sqrt{2} - 1 = [0; \overline{2}]$, where $\delta_S = \sqrt{2} + 1$ is the silver ratio. The corresponding Ostrowski numeration system is based on the Pell sequence $(q_n)_{n \geq 0} = 1, 2, 5, 12, \ldots$. For example, the representation for $N = 27$ is $[2011]_\beta$.

2.2 Decidability

In Section 1.2 we discussed previous work on procedures developed for deciding properties of k-automatic and Fibonacci-automatic sequences. We extend these results to a generalized Ostrowski numeration system by showing that addition in any Ostrowski numeration system can be performed using an automaton.

Hieronymi and Terry [31] recently showed that addition for an Ostrowski-α numeration system is recognizable when α is a quadratic irrational number. In Chapter 3, we provide an implementation for this case. Frougny and Solomyak [28] have shown that addition is computable in any of the Pisot numeration systems, and hence, a theorem analogous to Theorem 1.3 holds for these systems as well.

2.3 Constructing the Framework

For a generalized Ostrowski-α numeration system with $\alpha = [0; d_1, d_2, \ldots]$, let $\mathcal{L}$ denote the language of canonical representations. To define the framework, we require two automata: first, an automaton that accepts all canonical representations of non-negative integers and rejects representations that do not obey the canonical rules, and second, an automaton that recognizes the addition relation in this numeration system.

2.3.1 Recognizing the Canonical Representation

Recall the canonical rules for an Ostrowski-α numeration system. In this section, we provide an automaton that recognizes the language $\mathcal{L}$ of valid canonical representations. This automaton reads two inputs in parallel, in MSD-first notation — a representation $x = x_{k-1}x_{k-2}\cdots x_0$, and the sequence of the continued fraction $d = d_k d_{k-1}\cdots d_1$ — and accepts if and only if the representation is canonical. For details on how two inputs are read in parallel, see [37, Section 2]. Informally, to read m inputs over an alphabet Σ, we project the i^{th} symbols of all the inputs into an m-tuple, thereby extending the alphabet to $\underbrace{\Sigma \times \cdots \times \Sigma}_{m\text{-times}}$.

The input symbol being read by the automaton is denoted by the pair (d_i, x_{i-1}), where x_{i-1} is a symbol in x and d_i is the corresponding symbol in d (Figure 2.1). For example, consider the Fibonacci numeration system where $\alpha = [0; 2, \overline{1}]$, and a representation $x = 10010$. The tuples read in order are $(1,1), (0,1), (0,1), (1,1), (0,2)$.

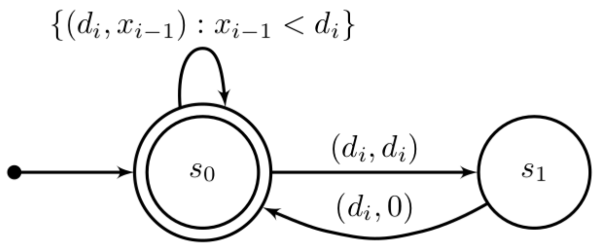

Figure 2.1: Automaton recognizing a canonical Ostrowski representation.

Theorem 2.1. *For $\alpha = [0; d_1, d_2, \ldots]$, the automaton in Figure 2.1 accepts an input (x, d) if and only if x is a canonical representation in the Ostrowski-α numeration system.*

Proof. Let Σ be the alphabet for all representations. According to the canonical rules, every symbol in x that equals the corresponding symbol in the sequence of continued fraction d must be followed by a 0. This rule is enforced by the transitions (d_i, d_i) for $s_0 \to s_1$ and $(d_i, 0)$ for $s_1 \to s_0$. Furthermore, the least significant symbol in x must be less than d_1, which is enforced by the transition (d_i, d_i) ending in the non-accepting state s_1. $\square$

2.3.2 Recognizing the Addition Relation

In order to recognize the addition relation, we require an automaton that reads three integers represented in their canonical representation x, y, z, along with the sequence of continued fraction d in parallel, and accepts if $[x]_\alpha + [y]_\alpha = [z]_\alpha$. To construct this automaton, we use an idea given by Luke Schaeffer to encode information about the required difference in values within the states of the automaton. Recall that $(q_n)_{n\geq 0}$ is the sequence of denominators of the convergents of α, and is used as place values for the numeration system. To work on parallel inputs, we pad the smaller inputs with leading 0's so that $|x| = |y| = |z| = k$. We let

$$\begin{aligned} x &= x_{k-1}x_{k-2}\cdots x_0, \\ y &= y_{k-1}y_{k-2}\cdots y_0, \quad \text{and} \\ z &= z_{k-1}z_{k-2}\cdots z_0. \end{aligned}$$

We also define $w = w_{k-1}w_{k-2}\cdots w_0$ where $w_i = z_i - (x_i + y_i)$ for all $0 \leq i < k$. Note that the integer value represented by any sequence $u = u_{k-1}\cdots u_0$ is $[u]_\alpha = \sum_{0\leq i<k} q_i u_i$.

For ease of notation, we say that the automaton reads a pair of parallel inputs (d, w) instead of the 4-tuple (d, x, y, z). This does not affect the result as we are only concerned with the difference $z_i - (x_i + y_i)$ instead of their actual values.

2.3.3 States and Transitions of the Adder

We label the states with a pair of integers (r, s). We denote the adder automaton by the tuple $A = (Q, \Delta, (r_0, s_0), F, \delta)$, where Q is the set of states (r, s) for all $r, s \in \{-1, 0, 1\}$, $F = \{(r, s) \in Q : s = 0\}$ is the set of final states, (r_0, s_0) is the initial state, and δ is the transition function extended to the domain $Q \times \Delta^*$ as usual. We construct the automaton such that if we start from the state (r, s) and read the input $(d_i, w_{i-1})(d_{i-1}, w_{i-2})\cdots(d_1, w_0)$, then we reach an accepting state if and only if

$$\sum_{0\leq j<i} q_j w_j = rq_{i-1} + sq_i. \tag{2.1}$$

Intuitively, we model the states (r, s) such that the remaining sequence of input evaluates to the difference $rq_{i-1} + sq_i$. Before we proceed, there are two crucial observations. In the canonical representation, every non-negative integer has a unique representation. Therefore, a representation of the form $u = \mathtt{10}^k$ evaluates to $[u]_\alpha = q_k$, analogous to a representation in base b of the form $\mathtt{10}^*$ being a power of b. Hence, we have that $-2q_i + 2 \leq [w_{i-1}w_{i-2}\cdots w_0]_\alpha \leq q_i - 1$.

Furthermore, we note that $r_0 = s_0 = 0$, since we want A to accept if and only if $\sum_{0 \le i < k} q_i w_i = 0$. For the same reason, we must have $\delta((0,0),0) = (0,0)$, that is, in the initial state the transition on reading the pair $(d_i, 0)$ keeps the DFA in the same state. Note that although the input being read is a pair (d_{i+1}, w_i), yet to decide the transition we are only concerned with how w_i is related to d_{i+1}. Therefore, for ease of notation, we write $\delta((r,s), w_i)$ instead of $\delta((r,s),(d_{i+1}, w_i))$.

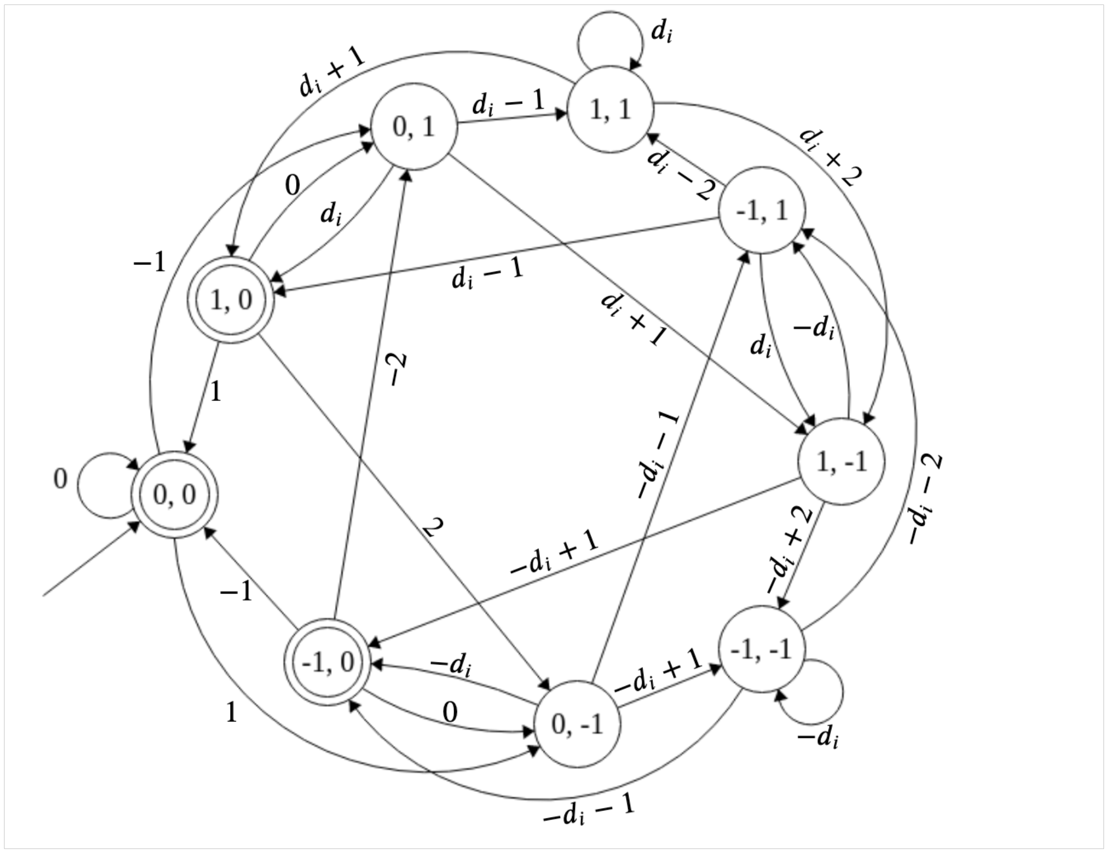

Figure 2.2: Automaton recognizing an Ostrowski addition relation.

Theorem 2.2. *If the automaton in Figure 2.2 is to process an input of length i: $w_{i-1} \cdots w_0$, starting in an arbitrary state (r, s) for $r, s \in \{-1, 0, 1\}$, then the transitions from state (r, s) that may lead to an accepting state are of the form $w_{i-1} = r + sd_i - t$ for $t \in \{-1, 0, 1\}$, and the destination state is (s, t).*

Proof. After reading in the first symbol w_{i-1}, the sum of the remaining input is bounded as follows:

$$-2q_{i-1} + 2 \leq \sum_{0 \leq j < i-1} q_j w_j \leq q_{i-1} - 1.$$

The upper bound is achieved when all x_j, y_j are 0 for $j < i-1$ and $[z_{i-2} \cdots z_0]_\alpha = q_{i-1} - 1$, while the lower bound is achieved when all z_j are 0 and the x_j's and y_j's both evaluate to $q_{i-1} - 1$. Combining these bounds with the definition of a state in Equation (2.1) and separating out the immediate next transition, we get

$$-2q_{i-1} + 2 \leq (rq_{i-1} + sq_i) - w_{i-1}q_{i-1} \leq q_{i-1} - 1.$$

Therefore, for the upper bound on w_{i-1} we have

$$\begin{aligned} w_{i-1}q_{i-1} &\leq rq_{i-1} + sq_i + 2q_{i-1} - 2 \\ &\leq rq_{i-1} + s(d_i q_{i-1} + q_{i-2}) + 2q_{i-1} - 2 \\ &\leq (r + sd_i + 2)q_{i-1} + sq_{i-2} - 2. \end{aligned}$$

For $s > 0$, we have

$$w_{i-1}q_{i-1} \leq (r + d_i + 2)q_{i-1} + q_{i-2} - 2.$$

Since $w_{i-1} = z_{i-1} - (x_{i-1} + y_{i-1})$, we also have that $w_{i-1} \leq d_i$. Hence, in this case we get

$$\begin{aligned} w_{i-1}q_{i-1} &\leq (r + sd_i + 1)q_{i-1} \\ \implies w_{i-1} &\leq (r + sd_i + 1). \end{aligned}$$

For $s \leq 0$, we have

$$\begin{aligned} w_{i-1}q_{i-1} &\leq (r + sd_i + 2)q_{i-1} + sq_{i-2} - 2 \\ &< (r + sd_i + 2)q_{i-1} \\ \implies w_{i-1} &\leq (r + sd_i + 1). \end{aligned}$$

Hence, we get the upper bound $w_{i-1} \leq (r + sd_i + 1)$. For the lower bound, consider

$$\begin{aligned} w_{i-1}q_{i-1} &\geq rq_{i-1} + sq_i - q_{i-1} + 1 \\ &\geq rq_{i-1} + sd_i q_{i-1} + sq_{i-2} - q_{i-1} + 1 \\ &\geq (r + sd_i - 1)q_{i-1} + sq_{i-2} + 1 \\ \implies w_{i-1} &\geq (r + sd_i - 1). \end{aligned}$$

Therefore, we have the following bounds on w_{i-1}.

$$r + sd_i - 1 \leq w_{i-1} \leq r + sd_i + 1. \tag{2.2}$$

Hence, all transitions are of the form $r + sd_i - t$ for some $t \in \{-1, 0, 1\}$. Furthermore, consider the transition $r+sd_i-t$ originating at state (r, s). If we separate out the immediate next transition, we have that

$$\begin{aligned}
\sum_{0 \leq j < i-1} q_j w_j + w_{i-1} q_{i-1} &= r q_{i-1} + s q_i \\
\sum_{0 \leq j < i-1} q_j w_j &= r q_{i-1} + s q_i - (r + sd_i - t) q_{i-1} \\
&= s(d_i q_{i-1} + q_{i-2}) - (sd_i - t) q_{i-1} \\
&= s q_{i-2} + t q_{i-1}.
\end{aligned}$$

This essentially gives us the transition function,

$$\delta((r, s), r + sd_i - t) = (s, t) \text{ for all } t \in \{-1, 0, 1\}. \tag{2.3}$$

□

The initial state is $(0, 0)$. From Theorem 2.2, it follows that there are only 9 states required, for $r, s \in \{-1, 0, 1\}$ and each of them has three transitions from it, of the form $r + sd_i - t$ for $t \in \{-1, 0, 1\}$. We provide this automaton in Figure 2.2, where states are labeled (r, s) and a transition from state (r, s) to state (s, t) is labeled $w_{i-1} = r + sd_i - t$. In the next section, we show that two of these states are never part of an accepting path, and hence, they can be removed.

2.3.4 Eliminating Redundant States

Observe that the bound in Equation (2.2) can be improved for specific values of r, s and t. First, we have that $w_{i-1} \leq d_i$. Therefore, all transitions labeled $d_i + 1$ and $d_i + 2$ can be removed. Next, consider the initial state $(0, 0)$. From this state we cannot have a transition with $w_{i-1} < 0$, because the maximum value that $z_{i-2} \cdots z_0$ can take is $q_{i-1} - 1$ and the sum required to close the gap will be q_{i-1}, implying that it will never lead to an accepting state. Hence, the transition -1 from $(0, 0)$ to $(0, 1)$ can also be removed.

After removing these transitions, the non-final state $(1, 1)$ does not have a transition to any other state, so it can be removed along with all its transitions. Since $w_{i-1} = 0$ implies that $z_{i-1} = d_i$ and $z_{i-2} = 0$, we now consider state $(0, 1)$. The only outgoing transition from this state is to state $(1, 0)$ labeled d_i. In any canonical representation $u = u_{i-1}u_{i-2} \cdots u_0$, we have $u_0 < d_1$; therefore, if we take the transition d_i from $(0, 1)$, there must exist another transition from $(1, 0)$ with $w_{i-2} \leq 0$. The only such transition is 0, which takes us back to $(0, 1)$. Therefore, there is no path from state $(0, 1)$ that leads to an accepting state, implying that the state can be removed. The minimized adder is given in Figure 2.3.

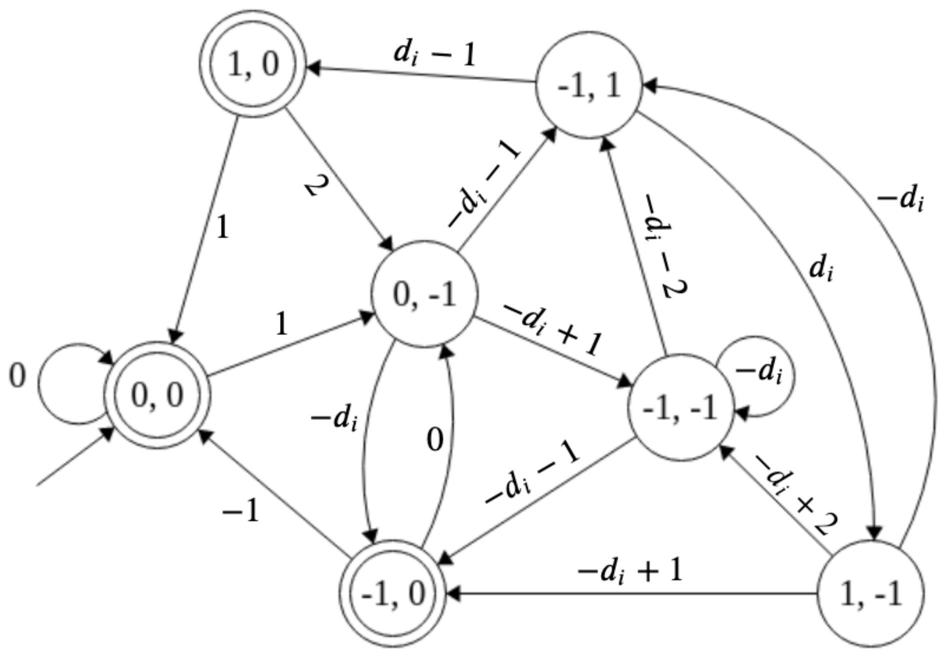

Figure 2.3: Minimized Ostrowski adder.

In the next chapter, we present an implementation of the Ostrowski adder and integrate it with `Walnut`, automatic theorem-proving software written by Hamoon Mousavi [36]. Our implementation considers the case where α is a quadratic irrational number. The more general implementation for an arbitrary real α requires encoding the terms of the continued fraction of α and reading it in parallel with the three inputs x, y, z of the addition relation. We discuss this further in Section 5.2.

Chapter 3

Implementation Details

To decide properties of Ostrowski-automatic sequences that are expressible in first-order logic, the adder must be expressed in a machine-readable format. `Walnut` has the feature to generate a base-b adder for any $b \geq 2$. In [37], Mousavi et al. extended `Walnut` by providing an implementation of a Fibonacci adder. In this chapter, we extend it further and generalize this feature to be able to generate any Ostrowski-α numeration system for quadratic irrationals α.

An adder in `Walnut` is an automaton that takes in 3 inputs x, y, z in parallel, and accepts if and only if $x + y = z$. Since the alphabet for the representations is finite, and transitions in the constructed adder (see Figure 2.3) depend on $w_i = z_i - x_i - y_i$, we model the transitions as triples (x_i, y_i, z_i). For example, for an alphabet $\{0, 1\}$ and $w_i = 0$, the transitions will be $(0, 0, 0)$, $(0, 1, 1)$ and $(1, 0, 1)$. Since the current implementation is restricted to a 3-input automaton, we implement the adder for the case where α is a quadratic irrational.

3.1 Working with Walnut

To understand the preliminaries about `Walnut` and how it works, see [36]. We recall that a custom numeration system can be introduced in `Walnut` by giving it the automata recognizing a canonical representation in that system, and the addition relation. In theory, the comparison automaton is also required, that is, recognizing $(x, y) : x < y$; however, if representations are canonical then a simple lexicographical comparison suffices.

In this chapter, we write a module that generates the representation and addition automata for any Ostrowski-α numeration system for quadratic irrational α, using 3 input parameters: a name for the numeration system, the *preperiod*, and the *period* of the continued fraction of α. The Walnut version equipped with this implementation supports a new command `ost`. To create a numeration system based on some quadratic irrational α, one can simply use the `ost` command with the desired parameters. An example is given below.

Example 3.1. To generate the Fibonacci numeration system, we have $\alpha = 1/\phi^2 = [0; 2, \overline{1}]$. So the preperiod is `[0 2]` and the period is `[1]`. Hence, we use the following command, which generates the required representation and addition automata.

```
ost fib [0 2] [1];
```

Now one can use this numeration system in predicates like usual:

```
eval test "?msd_fib <predicate>";
```

Variable	Initialized value	Description
q_0	$((0,0),0,0)$	Denotes a state of the NFA.
Q	$\{q_0\}$	Denotes the set of all states of the NFA.
F	$\{\}$	Denotes the set of final states of the NFA.
queue	$\{\}$	BFS queue used to explore new states.
δ	$\{\}$	Transition function for the NFA.
η	Specified by Figure 2.3	Transition function for the 4-input adder.
d	$[d_1, \ldots, d_m, d_{m+1}, \ldots, d_{m+n}]$	Continued fraction expansion of α.

Table 3.1: Global variables for the adder generation algorithm.

3.2 Generation Algorithm

To generate the 3-input adder for a given set of parameters: the preperiod and the period, we start simultaneously in all states in the 4-input automaton (see Figure 2.3) and traverse through the states that are reachable in a breadth-first fashion. This creates an NFA $M = (Q, \Sigma, \delta, q_0 \in Q, F \subseteq Q)$, which we later determinize and minimize to produce the final adder.

Let α be a quadratic irrational number, and let the continued fraction of α have preperiod $[d_1, \ldots, d_m]$ and period $[d_{m+1}, \ldots, d_{m+n}]$. A node (state) in the NFA is defined as a triplet (b, i_s, i_c), where $b = (r_b, s_b)$ represents one of the seven states in Figure 2.3, and $1 \leq i_s, i_c \leq m + n$ represent respectively the starting index and current index of the corresponding value in the continued fraction of α. We denote the transition function of the minimized 4-input adder by η, so that $\eta((r, s), r + sd_i - t) = (s, t)$. For ease of notation, we denote by $\eta((r, s), *)$ the set of all states (s, t) such that there exists a transition $(r, s) \rightarrow (s, t)$.

We present the algorithm as a set of three procedures. Informally, the algorithm constructs an NFA that accepts the desired continued fraction and intersects it with the automaton in Figure 2.3. The main procedure is Procedure 1 – GenerateStates, which calls two subroutines AddTransition and MarkFinalStates. The NFA is then minimized using Hopcroft's algorithm [32].

Remark. The initial values of the global variables that are accessible by all procedures are given in Table 3.1.

Procedure 1 – GenerateStates — starts with the initial state, takes ϵ transitions to all states where the input may start, and performs a breadth-first search over all possible expected inputs, creating all the required states and adding all the transitions.

Procedure 1 GenerateStates

```
1: for $i \leftarrow 1$ to $m + n$ do
2:     $q \leftarrow ((0, 0), i, i)$
3:     $\delta(q_0, \epsilon) \leftarrow \delta(q_0, \epsilon) \cup \{q\}$
4:     $Q \leftarrow Q \cup \{q\}$
5:     queue.push($q$)
6: end for
7: while queue $\neq \{\}$ do
8:     $q = ((r, s), i_s, i_c) \leftarrow$ queue.pop()
9:     for $(s, t) \in \eta((r, s), *)$ do
10:        if $i_c > 1$ then
11:            AddTransition($q, s, t, i_c - 1$)
12:        end if
13:        if $i_c = m + 1$ then
14:            AddTransition($q, s, t, m + n$)
15:        end if
16:    end for
17: end while
18: MarkFinalStates
```

Procedure 2 – AddTransition — adds a transition in the NFA. If the required destination state exists, then we only add the appropriate transition; otherwise, we also create the new state and push it to the queue. This ensures that we explore all possible paths from the new state in the breadth-first search.

Procedure 2 AddTransition($q, s, t, index$)

```
1: $u \leftarrow ((s,t), i_s, idx)$
2: $\delta(q, r + sd_{idx} - t) \leftarrow \delta(q, r + sd_{idx} - t) \cup \{u\}$
3: if $u \notin Q$ then
4:     $Q \leftarrow Q \cup \{u\}$
5:     queue.push($u$)
6: end if
```

Procedure 3 – MarkFinalStates adds the final states to F. Recall from Section 2.3.2 that a state with label (r, s) is final if $s = 0$. Therefore, in the NFA, we mark those states $q = ((r, s), i_s, i_c)$ as final, for which $s = 0$ and $i_c = 1$. The latter condition is required in order to make sure we completely read the input.

Procedure 3 MarkFinalStates

```
1: for $q \in Q$ do
2:     $q = ((r,s), i_s, i_c)$
3:     if $s = 0$ and $i_c = 1$ then
4:         $F \leftarrow F \cup \{q\}$
5:     end if
6: end for
```

3.3 Mechanical Verification of the Adder

In [37], the authors show that the generated adder can itself be verified mechanically. To show that an adder $\mathcal{A}$ specifies a function $A(x, y)$, we assert that there is exactly one sum of x and y using the predicates

$$\forall x \; \forall \; y \; \exists z \; \mathcal{A}(x, y, z), \text{ and} \tag{3.1}$$

$$\forall x \; \forall \; y \; \forall u \; \forall v \; (\mathcal{A}(x, y, u) \land \mathcal{A}(x, y, v)) \implies u = v. \tag{3.2}$$

To check associativity, we use the predicate

$$\forall x \; \forall y \; \forall z \; \forall w \; \forall r \; \forall s \; \forall t (\mathcal{A}(x, y, r) \land \mathcal{A}(r, z, t) \land \mathcal{A}(y, z, s)) \implies \mathcal{A}(x, s, t).$$

Next, we can show that the function $A(x, y)$ indeed performs addition using induction. First, we check that for all x, $A(x, 0) = x$. This translates to the predicate

$$\forall x\ \forall y\ \mathcal{A}(x, 0, y) \iff x = y.$$

We note that the *successor* can be defined in first-order logic. The following predicate asserts that y is a successor of x if $x < y$ and there do not exist any integers between x and y.

$$\mathrm{Succ}(x, y) := (x < y) \wedge (\forall z\ z \leq x \vee z \geq y).$$

Finally, we induct using the definition of the successor above.

$$\forall x\ \forall y\ \forall z\ \forall u\ \forall v(\mathrm{Succ}(y, u) \wedge \mathrm{Succ}(z, v)) \implies (\mathcal{A}(x, y, z) \iff \mathcal{A}(x, u, v)).$$

3.4 Examples

We show how the commands work using the Fibonacci numeration system as an example. We already saw how to generate this system in Example 3.1. Now we provide a more complete set of commands to create and verify an adder. We copy the adder automaton file `Custom Bases/msd_fib_addition.txt` in the `Automata Library` directory and name it `Adder.txt` for the below commands to work.

Generation	`ost fib [0 2] [1];`
Verification	`eval Function "?msd_fib (Ax,y Ez $Adder(x,y,z)) & (Ax,y,u,v ($Adder(x,y,u) & $Adder(x,y,v)) => u = v)";`
	`def Succ "?msd_fib x < y & (Az (z <= x) | (z >= y))";`
	`eval BaseCase "?msd_fib Ax,z ((x + 0 = z) <=> (x = z))";`
	`eval Induction "?msd_fib Ax,y,z,u,v ($Succ(y, u) & $Succ(z, v)) => ((x + y = z) <=> (x + u = v))";`

Table 3.2: `Walnut` commands to generate and verify an Ostrowski adder.

Both the predicates `BaseCase` and `Induction` evaluate to true, showing that the generated automaton indeed recognizes the addition relation.

Chapter 4

Applications

In this chapter, we discuss several applications of our work in the area of combinatorics on words.

4.1 Repetition Threshold for Balanced Words

In a paper by Rampersad et al. [46], the authors conjectured that the smallest possible critical exponent of an infinite balanced word over a k-letter alphabet is $(k-2)/(k-3)$ for all $k \geq 5$. We resolve this result for all $k \leq 8$, using a formulation of first-order logic and machine computation supported by the decision procedure developed in Chapter 2.

4.1.1 Definitions

We say that a word x is a *factor* of the word w if x appears as a contiguous subword inside w. Let $\mathrm{Fac}(w)$ denote the set of all factors of w. We begin with some definitions that are used throughout this chapter.

Definition 4.1. A word w over the alphabet Σ is *balanced* if for every symbol $a \in \Sigma$, and every pair of words u, $v \in \mathrm{Fac}(w)$ with $|u| = |v|$, we have $||u|_a - |v|_a| \leq 1$.

Definition 4.2. *Sturmian words*, denoted by $\mathbf{c}_{\alpha,\beta}$, can be defined in terms of two real parameters α, β with $0 \leq \alpha, \beta < 1$, and α irrational as follows:

$$\mathbf{c}_{\alpha,\beta}[n] := \lfloor \alpha(n+1) + \beta \rfloor - \lfloor \alpha n + \beta \rfloor.$$

A Sturmian word is called *characteristic* if $\beta = 0$, and is written as $\mathbf{c}_\alpha$. In this case, it is well-known that an alternative characterization for these words can be given in terms of the continued fraction expansion of $\alpha = [d_0, d_1, d_2, \ldots]$ where $d_i \in \mathbb{N}$ for $i \geq 0$ and $d_i \geq 1$ for $i \geq 1$. Then $\mathbf{c}_\alpha$ is produced as the limit of the sequence of *standard words* s_n defined as follows:

$$s_0 = 0, \quad s_1 = 0^{d_1 - 1}1, \quad s_n = s_{n-1}^{d_n} s_{n-2} \text{ for } n \geq 2.$$

The class of Sturmian words and the class of infinite aperiodic balanced words coincide over a binary alphabet. Vuillon [58] provides a survey on some previous work on balanced words, and Berstel et al. [7] provide a survey on Sturmian words.

Definition 4.3. Let $w = w_0 w_1 \cdots w_{n-1}$ be a finite word of length n. Then $p \in \mathbb{N}$ is a *period* of w if $w_i = w_{i+p}$ for all i with $0 \leq i < n - p$.

We say that a word u has *exponent* e and write $u = z^e$, where $e = |u|/p$ is a positive rational number, and z is the prefix of u of length p. A word may have multiple periods and exponents. We say u is *primitive* if its only integer exponent is 1. If u is a finite nonempty word, then u^ω denotes the infinite word $uuu\cdots$.

Example 4.1. The word $w =$ `alfalfa` has three periods: $p_1 = 3$, $p_2 = 6$, and $p_3 = 7$. The corresponding exponents are $e_1 = 7/3$, $e_2 = 7/6$, and $e_3 = 1$. In this example, w is a primitive word since its only integer exponent is 1.

Rampersad et al. [46] gave a method to construct infinite balanced words from binary Sturmian words, using a characterization of recurrent aperiodic balanced words given by Hubert [33]. Their method is based on the notion of the constant gap property.

Definition 4.4. An infinite word $\mathbf{w}$ has the *constant gap property* if, for each symbol a, there is a positive integer d such that the distance between successive occurrences of a in $\mathbf{w}$ is always d.

For example, $(0102)^\omega = 010201020102\cdots$ has the constant gap property because the distance between consecutive 0's is always 2, while the distance between consecutive 1's (resp., 2's) is always 4.

Definition 4.5. The *critical exponent* of an infinite word w is defined to be the supremum of the set of all rational numbers e such that there exists a finite nonempty factor of w with exponent e.

Definition 4.6. The *repetition threshold* on an alphabet of size k is the infimum of the set of exponents e such that there exists an infinite word that avoids greater than e-powers.

4.1.2 Constructing Balanced Words from Sturmian Words

The authors of [46] constructed certain infinite balanced words over the alphabet $\Sigma_k = \{0, \ldots, k-1\}$, denoted by $\mathbf{x}_k$, for $3 \le k \le 10$. Their construction uses characteristic Sturmian words $\mathbf{c}_\alpha$ and a pair of constant gap words $\mathbf{y}$ and $\mathbf{y}'$, where α, $\mathbf{y}$ and $\mathbf{y}'$ are carefully chosen (see Table 4.1). Here $\varphi = (1+\sqrt{5})/2$ is the golden ratio. The authors also proved that $E(\mathbf{x}_3) = 2 + \frac{\sqrt{2}}{2}$ and $E(\mathbf{x}_4) = 1 + \frac{\varphi}{2}$; furthermore, they showed that $E(\mathbf{x}_3)$ is the least possible critical exponent over an alphabet of 3 symbols. A proof that the critical exponent for $\mathbf{x}_4$ is actually minimal was given by Peltomäki.

k	c.f.	α	y	y'
3	$\sqrt{2}-1$	$[0;\overline{2}]$	$(01)^\omega$	2^ω
4	$1/\varphi^2$	$[0;2,\overline{1}]$	$(01)^\omega$	$(23)^\omega$
5	$\sqrt{2}-1$	$[0;\overline{2}]$	$(0102)^\omega$	$(34)^\omega$
6	$(78-2\sqrt{6})/101$	$[0;1,2,1,1,\overline{1,1,1,2}]$	0^ω	$(123415321435)^\omega$
7	$(63-\sqrt{10})/107$	$[0;1,1,3,\overline{1,2,1}]$	$(01)^\omega$	$(234526432546)^\omega$
8	$(23+\sqrt{2})/31$	$[0;1,3,1,\overline{2}]$	$(01)^\omega$	$(2345267325462375264325 76)^\omega$
9	$(23-\sqrt{2})/31$	$[0;1,2,3,\overline{2}]$	$(01)^\omega$	$(234567284365274863254768)^\omega$
10	$(109+\sqrt{13})/138$	$[0;1,4,2,\overline{3}]$	$(01)^\omega$	$(234567284963254768294365274869)^\omega$

Table 4.1: Parameters α, y, and y' for construction of balanced words $\mathbf{x}_k$.

To prove that these words $\mathbf{x}_k$ do indeed achieve the claimed critical exponent, we use the computational approach based on the methods of Du et al. [24, 37]. Using Theorems 4.1 and 4.2, we can construct a DFAO for each of the balanced words $\mathbf{x}_k$. Due to the absence of an implementation of numeration systems based on these irrational numbers, the authors in [46] left the proofs for $5 \le k \le 10$ as an open problem. We use our implementation given in Chapter 3 to prove the results for $5 \le k \le 8$. For $k = 9$ and 10, the memory requirements made the computation infeasible on a machine with 400GB of memory. In theory, the results might be provable on a machine with more memory.

Theorem 4.1. *[1, Theorem 9.1.15] Let $N \ge 1$ be an integer with Ostrowski-α representation $b_j \cdots b_0$. Then $\mathbf{c}_\alpha[N] = 1$ if and only if $b_j \cdots b_0$ ends with an odd number of 0's.*

Theorem 4.2. *[46, Theorem 12] Let α be a quadratic irrational and let $\mathbf{c}_\alpha$ be the characteristic Sturmian word with slope α. Let $\mathbf{x}$ be any word obtained by replacing the 0's in $\mathbf{c}_\alpha$ with a periodic sequence $\mathbf{y}$ and replacing the 1's with a periodic sequence $\mathbf{y}'$. Then $\mathbf{x}$ is Ostrowski α-automatic.*

Proof. Let p and p' be the periods of words y and y' respectively. For a positive integer n, the value of $\mathbf{x}[n]$ depends on

1. the value of $\mathbf{c}_\alpha[n]$,
2. the number of 0's modulo p in the length-n prefix of $\mathbf{c}_\alpha[n]$, and
3. the number of 1's modulo p' in the length-n prefix of $\mathbf{c}_\alpha[n]$.

Let $b_j b_{j-1} \cdots b_0$ be the Ostrowski-α representation of n. By [1, Lemma 9.1.9 and Theorem 9.1.13], we have

$$|\mathbf{c}_\alpha[1..n]|_0 = b_j(q_j - p_j) + b_{j-1}(q_{j-1} - p_{j-1}) + \cdots + b_0(q_0 - p_0), \tag{4.1}$$
$$|\mathbf{c}_\alpha[1..n]|_1 = b_j p_j + b_{j-1} p_{j-1} + \cdots + b_0 p_0. \tag{4.2}$$

Here p_i and q_i are the numerators and denominators of the convergents of α (see Equations (1.1) and (1.2)). Since α is a quadratic irrational, its continued fraction is ultimately periodic, implying that the sequences $((q_i - p_i) \bmod p)_{i \geq 0}$ and $(p_i \bmod p')_{i \geq 0}$ are ultimately periodic. Based on this periodicity, we can construct an automaton that computes the word $\mathbf{x}_k$. □

4.1.3 Determining the Critical Exponents of $\mathbf{x}_k$

For each of the words $\mathbf{x}_k$ constructed using the parameters in Table 4.1, we perform the following procedure to determine its critical exponent. Note that the values of the critical exponents were already conjectured by the authors in [46].

1. Create the required Ostrowski numeration system $\mathcal{N}$.
2. Construct the $\mathcal{N}$-automaton producing $\mathbf{x}_k$.
3. Assert with first-order predicates that the maximum possible exponent of a subword in $\mathbf{x}_k$ is $\frac{k-2}{k-3}$.

We observe that the DFAO for the words $\mathbf{x}_k$ for $6 \leq k \leq 10$ are smaller in size if we construct them in LSD-first notation instead of MSD-first. Therefore, all commands below follow the LSD-first notation.

We now present the commands that resolve the conjecture for $5 \leq k \leq 8$. The balanced word $\mathbf{x}_k$ is denoted by `Xk` in the following commands. First, we create the required numeration systems for $5 \leq k \leq 10$.

```
ost ns5 [0] [2];
ost ns6 [0 1 2 1 1] [1 1 1 2];
ost ns7 [0 1 1 3] [1 2 1];
ost ns8 [0 1 3 1] [2];
ost ns9 [0 1 2 3] [2];
ost ns10 [0 1 4 2] [3];
```

Next, we show the commands to verify the critical exponent for each $\mathbf{x}_k$. As an example, we show the commands for $\mathbf{x}_6$. For other words, the commands are similar and can be found on Github. The claimed value of the critical exponent is the rational number $(k-2)/(k-3)$, and so, we assert the following two statements using first-order predicates.

1. There exist integers $i, p \geq 1$ such that for all j with $(k-3)j < p$, we have $\mathbf{x}[i+j] = \mathbf{x}[i+j+p]$. In other words, there exists a subword that has exponent $(k-2)/(k-3)$. In `Walnut`, for $\mathbf{x}_6$ we write:

```
eval CritExp "?lsd_ns6 Ei,p (i >= 1) & (p >= 1) &
    (Aj (3*j < p) => X6[i + j] = X6[i + j + p])";
```

2. There do not exist integers $i, p \geq 1$ such that for all j with $(k-3)j \leq p$, we have $\mathbf{x}[i+j] = \mathbf{x}[i+j+p]$. In other words, there does not exist a subword that has exponent greater than $(k-2)/(k-3)$. For $\mathbf{x}_6$ we write the predicate:

```
eval CritExp "?lsd_ns6 ~Ei,p (i >= 1) & (p >= 1) &
    (Aj (3*j <= p) => X6[i + j] = X6[i + j + p])";
```

Both these predicates produce the `true` automaton for all $5 \leq k \leq 8$, proving the result. For reference, in Table 4.2 we provide the number of states in the DFAO, the approximate amount of memory consumed, and the time taken by the computation for each $\mathbf{x}_k$. We also provide below some specific factors of the words $\mathbf{x}_k$ for $6 \leq k \leq 8$ that realize the critical exponent $(k-2)/(k-3)$ as their exponent.

- The factor of $\mathbf{x}_6 = 1203410530214\cdots$, $\mathbf{x}_6[7..10] = 0530$, has exponent $4/3$.
- The factor of $\mathbf{x}_7 = 2031405216041\cdots$, $\mathbf{x}_7[2..6] = 03140$, has exponent $5/4$.
- The factor of $\mathbf{x}_8 = 2340526713254\cdots$, $\mathbf{x}_8[1..6] = 234052$, has exponent $6/5$.

k	States	Memory	Time
5	24	2 GB	30 seconds
6	210	40 GB	5 minutes
7	591	150 GB	45 minutes
8	781	360 GB	2 hours
9	780	—	—
10	1458	—	—

Table 4.2: Computational statistics for predicates involving $\mathbf{x}_k$.

We leave it as an open problem to compute the proofs for $\mathbf{x}_9$ and $\mathbf{x}_{10}$. Ideally, we require a result that will prove or disprove the conjecture in its entirety. This conjecture is analogous to Dejean's conjecture about the repetition threshold of infinite words over a k-letter alphabet [22], which was resolved in 2011 by Currie and Rampersad [20], and independently by Rao [47].

4.2 Critical Exponent of Rich Words

Palindromes are among the most widely studied repetitions in words. The class of rich words — those words that contain, as factors, the maximum possible number of distinct palindromes, was introduced in [10, 21, 29]. Since then, rich words have received much attention in the combinatorics on words literature; see, for example, [13, 30, 56]. Yet, there are still numerous interesting open problems concerning repetitions in rich words.

Example 4.2. The word 00010110 is rich because it contains 8 distinct nonempty palindromes: $0, 00, 000, 1, 010, 101, 11$, and 0110. The word 00101100 is not rich because it contains only 7 distinct palindromes.

In this section, we study lower bounds on the repetition threshold of infinite rich words over 2 and 3-letter alphabets and construct a candidate infinite rich word over the binary alphabet with a small critical exponent of $2 + \sqrt{2}/2$. This construction utilizes the framework developed in Chapter 2. Thus, our work is the first progress on an open problem of Vesti from 2017. Recently, Rampersad et al. [45] have proved that our candidate word indeed achieves the repetition threshold, and hence, the problem is completely resolved for the binary case. Before presenting our results, we provide a brief overview of previous research in this direction.

Definition 4.7. For a given alphabet Σ, a mapping φ on Σ^* is an *involutive antimorphism* if $\varphi(uv) = \varphi(v)\varphi(u)$, and $\varphi^2(u) = u$ for all $u, v \in \Sigma^*$.

Let the word w be the fixed point of a given involutive antimorphism Θ. We say w is a Θ-*palindrome* if $w = \Theta(w)$. The set of Θ-palindromic factors of a word w is denoted by $\operatorname{Pal}_\Theta(w)$. In 2013, Pelantová and Starosta introduced the idea of Θ-palindromic defect.

Definition 4.8. The Θ-*palindromic defect* of a finite word w is defined as

$$D_\Theta(w) = |w| + 1 - \gamma_\Theta(w) - |\operatorname{Pal}_\Theta(w)|,$$

where $\gamma_\Theta(w) = |\{\{a, \Theta(a)\} : a \in \Sigma, \text{a occurs in } w \text{ and } a \neq \Theta(a)\}|$. For an infinite word w, the Θ-palindromic defect is the supremum of the set of $D_\Theta(u)$, where u is a factor of w.

Pelantová and Starosta also proved that all recurrent words with a finite Θ-palindromic defect contain infinitely many overlapping factors [41]. This result leads to the following theorem.

Theorem 4.3. *All infinite rich words contain a square.*

Theorem 4.3 provides a trivial lower bound on the repetition threshold for infinite rich words over a k-letter alphabet; namely $RT(k) \geq 2$. In [57], Vesti gave both upper and lower bounds on the length of the longest square-free rich words, and proposed the open problem of determining the repetition threshold for infinite rich words.

4.2.1 Building the Candidate Rich Word r

We construct an infinite binary rich word and determine the value of its critical exponent. The word $\mathbf{r}$ is defined as the image of a fixed point, $\mathbf{r} = \tau(\varphi^\omega(0)) = 001001100100110\cdots$, where the morphisms φ and τ are given by

$$\begin{array}{llll} \varphi\colon & 0 \to 01 & \qquad \tau\colon & 0 \to 0 \\ & 1 \to 02 & & 1 \to 01 \\ & 2 \to 022, & & 2 \to 011. \end{array}$$

Observing the lengths $L_i = |\tau(\varphi^i(0))|$ for $i \geq 0$, we note that $L_0 = 1$, $L_1 = 3$, and $L_i = 2L_{i-1} + L_{i-2}$ for $i \geq 2$. This suggests that the word $\mathbf{r}$ might be *Pell-automatic*, that is, there exists a DFAO that takes as input an integer N represented in the Pell numeration system, and outputs the symbol in $\mathbf{r}$ at index N).

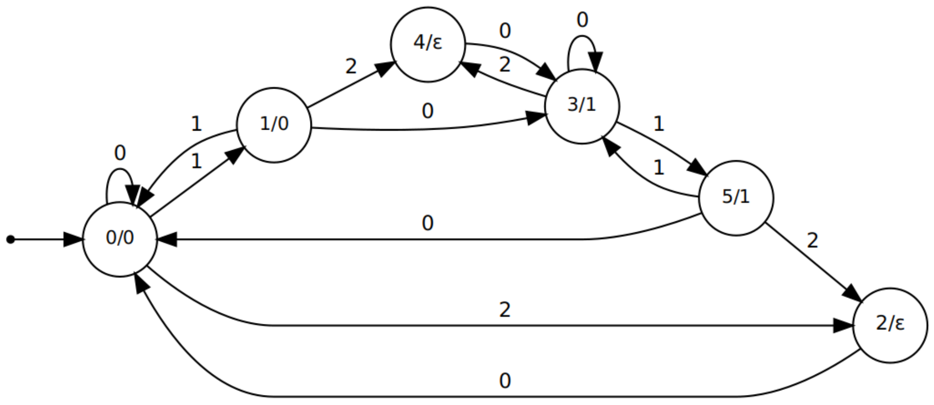

Figure 4.1: DFAO for the infinite rich word **r**.

We guess the DFAO for the word **r** using a combination of membership and equivalence queries as described in the $L*$ algorithm given by Angluin [2]. Figure 4.1 represents the constructed automaton. Before proceeding, we prove that this DFAO produces the same word as $\tau(\varphi^\omega(0))$.

Let f and g be the morphisms associated with the automaton in Figure 4.1. Then $g(f^\omega(0))$ denotes the infinite word produced. The morphisms f and g are given by

$$f\colon \begin{array}{l} 0 \to 012 \\ 1 \to 304 \\ 2 \to 0 \\ 3 \to 354 \\ 4 \to 3 \\ 5 \to 032, \end{array} \qquad g\colon \begin{array}{l} 0 \to 0 \\ 1 \to 0 \\ 2 \to \epsilon \\ 3 \to 1 \\ 4 \to \epsilon \\ 5 \to 1. \end{array}$$

Lemma 4.4. *For all $n \geq 2$, we have $g(f^n(0)) = g(f^{n-1}(0))g(f^{n-2}(3))g(f^{n-1}(0))$.*

Proof. We prove this by induction on n. For $n = 2$, we have that

$$g(f^2(0)) = g(f^1(0))g(3)g(f^1(0)) = 00100.$$

So the base case holds. Next, we construct the induction hypothesis,

$$H_1 : g(f^k(0)) = g(f^{k-1}(0))g(f^{k-2}(3))g(f^{k-1}(0)), \forall k \leq n.$$

For the inductive step, consider $g(f^{n+1}(0))$. Using the definition of the morphisms f and g, we have that,

$$\begin{aligned} g(f^{n+1}(0)) &= g(f^n(0))g(f^n(1))g(f^n(2)) \\ &= g(f^n(0))g(f^{n-1}(3))g(f^{n-1}(0))g(f^{n-1}(4))g(f^n(2)) \\ &= g(f^n(0))g(f^{n-1}(3))g(f^{n-1}(0))g(f^{n-2}(3))g(f^{n-1}(0)). \end{aligned} \tag{4.3}$$

Using the induction hypothesis H_1 in Equation (4.3), we get

$$g(f^{n+1}(0)) = g(f^n(0))g(f^{n-1}(3))g(f^n(0)).$$

□

Lemma 4.5. *For all $n \geq 2$, we have $g(f^n(3)) = g(f^{n-1}(3))g(f^{n-2}(0))g(f^{n-1}(3))$.*

Proof. The proof is similar to that of Lemma 4.4, by induction on n. For $n = 2$, we have

$$g(f^2(3)) = g(f^1(3))g(0)g(f^1(3)) = 11011.$$

So the base case holds. We have the induction hypothesis,

$$H_2 : g(f^k(3)) = g(f^{k-1}(3))g(f^{k-2}(0))g(f^{k-1}(3)), \forall k \leq n.$$

For the inductive step, consider $g(f^{n+1}(3))$. Using the definition of the morphisms f and g, we have that

$$\begin{aligned} g(f^{n+1}(3)) &= g(f^n(3))g(f^n(5))g(f^n(4)) \\ &= g(f^n(3))g(f^{n-1}(0))g(f^{n-1}(3))g(f^{n-1}(2))g(f^n(4)) \\ &= g(f^n(3))g(f^{n-1}(0))g(f^{n-1}(3))g(f^{n-2}(0))g(f^{n-1}(3)). \end{aligned} \tag{4.4}$$

Using the induction hypothesis H_2 in Equation (4.4), we get

$$g(f^{n+1}(3)) = g(f^n(3))g(f^{n-1}(0))g(f^n(3)).$$

□

The following theorem proves the desired equivalence.

Theorem 4.6. *The infinite words $\tau(\varphi^\omega(0))$ and $g(f^\omega(0))$ are equal.*

Proof. We prove this by a simultaneous induction on n with 3 hypotheses.

$$\tau(\varphi^k(0)) = g(f^k(0))g(f^{k-1}(3)) \tag{4.5}$$

$$\tau(\varphi^k(1)) = g(f^k(0))g(f^k(3)) \tag{4.6}$$

$$\tau(\varphi^k(2)) = g(f^k(0))g(f^{k+1}(3)) \tag{4.7}$$

The base case $k = 1$ can be checked by hand. Assume that the hypotheses hold for $k \leq n$. Next, we consider the following inductive steps using the definitions of φ and τ.

$$\begin{aligned}\tau(\varphi^{n+1}(0)) &= \tau(\varphi^n(0))\tau(\varphi^n(1)) \\ &= g(f^n(0))g(f^{n-1}(3))g(f^n(0))g(f^n(3)) && \text{using Equations (4.5) and (4.6)} \\ &= g(f^{n+1}(0))g(f^n(3)). && \text{using Lemma 4.4.}\end{aligned}$$

$$\begin{aligned}\tau(\varphi^{n+1}(1)) &= \tau(\varphi^n(0))\tau(\varphi^n(2)) \\ &= g(f^n(0))g(f^{n-1}(3))g(f^n(0))g(f^{n+1}(3)) && \text{using Equations (4.5) and (4.7)} \\ &= g(f^{n+1}(0))g(f^{n+1}(3)) && \text{using Lemma 4.4.}\end{aligned}$$

$$\begin{aligned}\tau(\varphi^{n+1}(2)) &= \tau(\varphi^n(0))\tau(\varphi^n(2))\tau(\varphi^n(2)) \\ &= g(f^n(0))g(f^{n-1}(3))g(f^n(0))g(f^{n+1}(3))g(f^n(0))g(f^{n+1}(3)) \\ &= g(f^{n+1}(0))g(f^{n+2}(3)) && \text{using Lemmas 4.4 and 4.5.}\end{aligned}$$

This proves that the hypotheses are true. From Equation (4.5), we have

$$\tau(\varphi^k(0)) = g(f^k(0))g(f^{k-1}(3)).$$

Letting $k \to \infty$, we get

$$\tau(\varphi^\omega(0)) = g(f^\omega(0)).$$

□

4.2.2 Proof of Palindromic Richness

We claim that the infinite word $\mathbf{r} = g(f^\omega(0)) = 001001100100110\cdots$ is rich[1]. The proof is carried out using `Walnut` by constructing a set of predicates based on a characterization

[1]We learned from Edita Pelantová that the word $\mathbf{r}$ is a complementary symmetric Rote word [48], and hence by [8, 42] it follows that $\mathbf{r}$ is rich. Yet, our approach is important because it lets us compute the critical exponent of the word.

of rich words given by Glen et al. in [29], that is also used by Schaeffer and Shallit in [52]. We say that a word w has a *unioccurrent* suffix s if s is not a factor of any proper prefix of w. The following theorem provides the characterization.

Theorem 4.7. *(Glen et al. [29]) A word w is rich if and only if every prefix of w has a unioccurrent palindromic suffix.*

All of the computations we describe were carried out in a few seconds on a Linux machine with an AMD Fx-8370e processor. In the following predicates, `R` denotes the automaton in Figure 4.1. First, we introduce the fundamental predicates that form the building blocks for verification of the richness property.

1. The predicate `FactorEq` takes 3 parameters i, j, n and evaluates to true if the length-n factors of $\mathbf{r}$ starting at indices i and j are equal.

```
def FactorEq "?msd_pell Ak (k < n) => (R[i + k] = R[j + k])";
```

2. The predicate `Occurs` takes 4 parameters i, j, m, n and evaluates to true if the length-m factor of $\mathbf{r}$ starting at index i occurs in the length-n factor starting at index j, i.e., $\texttt{R}[i..i+m-1]$ is a factor of $\texttt{R}[j..j+n-1]$.

```
def Occurs "?msd_pell (m <= n) &
    (Ek (k + m <= n) & $FactorEq(i, j + k, m))";
```

3. The predicate `Palindrome` takes 2 parameters i, n and evaluates to true if the length-n factor of $\mathbf{r}$ starting at index i is a palindrome.

```
def Palindrome "?msd_pell Aj,k ((k < n) & (j + k + 1 = n)) =>
    (R[i + k] = R[i + j])";
```

By Theorem 4.7, for any finite word to be rich, it is sufficient to check if all its prefixes have a unioccurrent palindromic suffix. We use this property to construct the predicate `RichFactor` which takes two parameters i, n, and evaluates to true if the length-n factor of $\mathbf{r}$ starting at index i is rich. Figure 4.2 shows the significance of the variables in this predicate.

```
def RichFactor "?msd_pell
    Am ((m >= 1) & (m <= n)) =>
        (Ej (i <= j) & (j < i + m) &
         $Palindrome(j, i + m - j) &
         ~$Occurs(j, i, i + m - j, m - 1))";
```

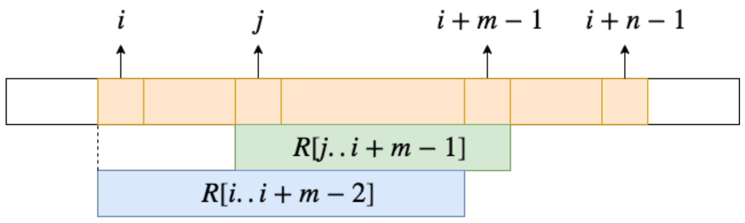

Figure 4.2: Significance of i, j, m, n in the predicate `RichFactor`.

Now, we simply check that all prefixes of **r** are rich to show that the infinite word **r** is rich. The following predicate, `R_Is_Rich` evaluates to true, which completes the proof.

```
eval R_Is_Rich "?msd_pell An $RichFactor(0, n)";
```

4.2.3 Determining the Critical Exponent

First, we observe that the critical exponent of **r** is < 3. This can be checked in `Walnut` with the following command.

```
eval CheckCritExp "?msd_pell ~(E i, p (p >= 1) &
    Aj (j < 2*p) => R[i + j] = R[i + j + p])";
```

Next, we compute the periods p such that a repetition with exponent $\geq 5/2$ and period p occurs in **r**. The language accepted by the produced automaton is 0^*1100^*, which is the Pell-base representation of numbers of the form $P_t + P_{t-1}$, for $t \geq 3$.

```
eval HighPowPeriods "?msd_pell (p >= 1) &
    (Ei Aj (2*j <= 3*p) => R[i + j] = R[i + j + p])";
```

Next, we compute pairs of integers (n, p) such that **r** has a factor of length $n + p$ with period p, and this factor cannot be extended to a longer factor beginning at the same position, of length $n + p + 1$ with the same period p.

```
def MaximalReps "?msd_pell Ei
    (Aj (j < n) => R[i + j] = R[i + j + p]) &
    (R[i + n] != R[i + n + p])";
```

Finally, we compute the pairs (n, p) where p matches the regular expression 0^*1100^* in the Pell-base representation, and $n + p$ is the maximum possible length of any factor with period p.

```
eval HighestPowersR "?msd_pell
    $HighPowPeriods(p) &
    $MaximalReps(n, p) &
    (Am $MaximalReps(m, p) => m <= n)";
```

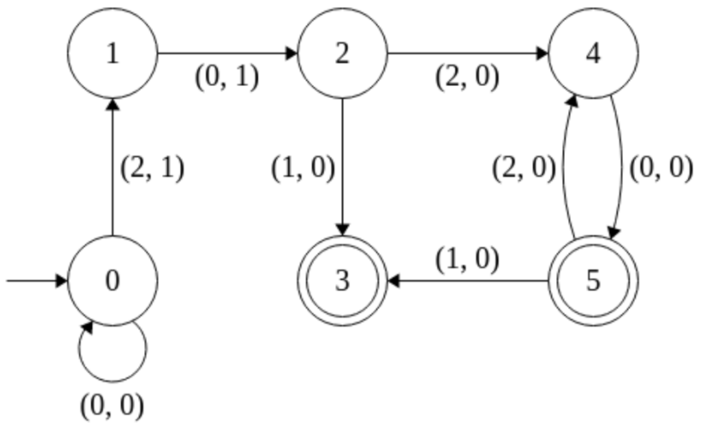

Figure 4.3: Pairs (n, p) satisfying the predicate `HighestPowersR`.

Figure 4.3 shows the automaton produced by the predicate `HighestPowersR`. It accepts pairs (n, p) of the following forms:

$$\binom{0}{0}^* \binom{2}{1} \binom{0}{1} \binom{1}{0}, \tag{4.8}$$

$$\binom{0}{0}^* \binom{2}{1} \binom{0}{1} \binom{2}{0} \binom{0}{0} \left\{ \binom{2}{0} \binom{0}{0} \right\}^*, \text{ or} \tag{4.9}$$

$$\binom{0}{0}^* \binom{2}{1} \binom{0}{1} \binom{2}{0} \binom{0}{0} \left\{ \binom{2}{0} \binom{0}{0} \right\}^* \binom{1}{0}. \tag{4.10}$$

Here, the length of the words is $n + p$ and the period is p. Equation (4.8) corresponds to $n = (201)_P = 11$ and $p = (110)_P = 7$. Thus we have

$$e = \frac{n + p}{p} = \frac{18}{7} \approx 2.57.$$

Equation (4.9) corresponds to

$$n = \sum_{1\le i\le k} 2P_{2i} = P_{2k+1} - 1, \quad p = P_{2k} + P_{2k-1}.$$

Equation (4.10) corresponds to

$$n = 1 + \sum_{1\le i\le k} 2P_{2i+1} = P_{2k+2} - 1, \quad p = P_{2k+1} + P_{2k}.$$

Putting $m = 2k-1$ for Equation (4.9), and $m = 2k$ for Equation (4.10), we notice that the expressions for n and p coincide.

$$\begin{aligned} e &= \frac{P_{m+2} + P_{m+1} + P_m - 1}{P_{m+1} + P_m} \\ &= 2 + \frac{P_{m+1} - 1}{P_{m+1} + P_m} \ . \end{aligned}$$

Since Pell numbers are the convergents of $\sqrt{2} - 1$, and the ratio P_{m+1}/P_m converges to $\sqrt{2} + 1$, we have that

$$\begin{aligned} e &= 2 + \frac{P_{m+1} - 1}{P_{m+1} + P_m} \\ &< 2 + \frac{\sqrt{2} + 1 + 1/P_m^2 - 1/P_m}{\sqrt{2} + 2 - 1/P_m^2} \ . \end{aligned} \tag{4.11}$$

For $m \ge 4$, as $m \to \infty$, the value in Equation (4.11) is increasing, and tends to $2+\sqrt{2}/2$. Thus, the critical exponent of the word $\mathbf{r}$ is $2 + \sqrt{2}/2$. All `Walnut` commands for verifying richness and computing the critical exponent are available on GitHub.

In [5], we conjectured using computational evidence that the word $\mathbf{r}$ achieves the repetition threshold for rich words over the binary alphabet. Very recently, Rampersad et al. [45] proved our conjecture by giving a lower bound of $2 + \sqrt{2}/2$ on the repetition threshold. We leave it as an open problem to determine the exact value of the threshold for larger alphabets and construct candidate words that achieve the same.

4.3 Infinite Binary Words Avoiding Antisquares

In this section, we explore another application of our work to pattern avoidance. We construct an infinite binary word with critical exponent $= 2 + \phi$, which avoids all antisquares other than 01 and 10. Here ϕ is the golden ratio. We also claim that this is the minimum possible critical exponent that can be achieved by any infinite binary word with this property.

An *antisquare* is defined as a finite word $w = xx'$ where x' is the binary complement of x. For example, 00101101 is an antisquare since 1101 is the binary complement of 0010. Naturally, studying antisquare avoidance is meaningful only for the binary alphabet.

4.3.1 Construction of the Candidate Word

Consider the morphisms below:

$$\begin{array}{llll} \varphi\colon & 0 \mapsto 001 & \tau\colon & 0 \mapsto 0001 \\ & 1 \mapsto 01 & & 1 \mapsto 01. \end{array}$$

We claim that the infinite word $\mathbf{w} = \tau(\varphi^\omega(0))$ does not have antisquares other than 01 and 10, and has the critical exponent $2+\phi$. To prove this, we follow an approach similar to that in Section 4.2. First, we build the DFAO producing this word (see Figure 4.4). Then, we show that the word avoids all antisquares other than 01 and 10 and also compute its critical exponent. Let f and g be the morphisms associated with this automaton. Then $g(f^\omega(0))$ denotes the infinite word produced. These morphisms are defined as follows:

$$\begin{array}{llll} f\colon & 0 \mapsto 01 & g\colon & 0 \mapsto 0 \\ & 1 \mapsto 2 & & 1 \mapsto 0 \\ & 2 \mapsto 34 & & 2 \mapsto 0 \\ & 3 \mapsto 56 & & 3 \mapsto 1 \\ & 4 \mapsto 3 & & 4 \mapsto 0 \\ & 5 \mapsto 27 & & 5 \mapsto 0 \\ & 6 \mapsto 8 & & 6 \mapsto 0 \\ & 7 \mapsto 5 & & 7 \mapsto 1 \\ & 8 \mapsto 9a & & 8 \mapsto 0 \\ & 9 \mapsto 81 & & 9 \mapsto 0 \\ & a \mapsto 2, & & a \mapsto 1. \end{array}$$

Before we proceed, we show that the set of morphisms (φ, τ) produce the same word as that produced by (f, g). First, we need a lemma.

Lemma 4.8. *For all $n \geq 0$, we have $g(f^{2n}(8)) = g(f^{2n}(0))$.*

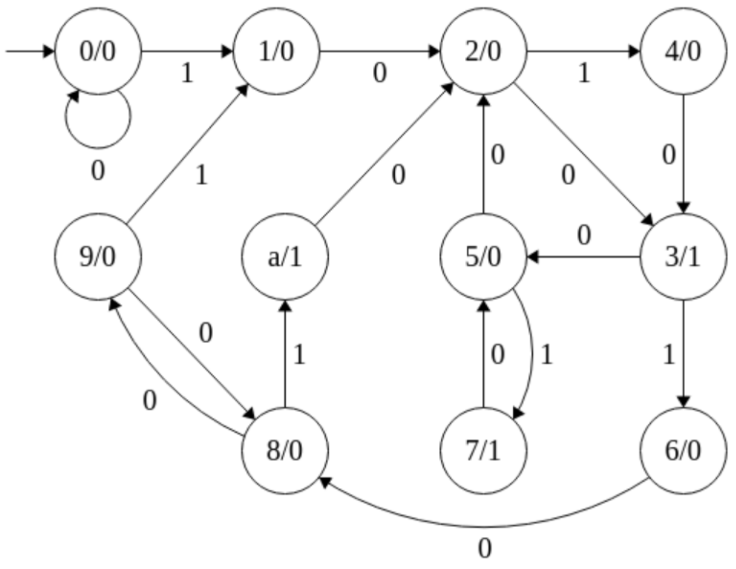

Figure 4.4: DFAO computing the infinite word **w** avoiding antisquares.

Proof. We prove this by induction on n. For the base case, we have $g(8) = g(0) = 0$. Assume that the hypothesis is true for all $k \le n$. Consider the inductive step:

$$\begin{aligned}
g(f^{2n+2}(8)) &= g(f^{2n+1}(9))g(f^{2n+1}(a)) \\
&= g(f^{2n+1}(9))g(f^{2n}(2)) \\
&= g(f^{2n}(8))g(f^{2n}(1))g(f^{2n}(2)) \\
&= g(f^{2n}(0))g(f^{2n}(1))g(f^{2n}(2)) \qquad \text{(using induction hypothesis)} \\
&= g(f^{2n+1}(0))g(f^{2n+1}(1)) \\
&= g(f^{2n+2}(0)).
\end{aligned}$$

□

Lemma 4.9. *Given the definitions of f and g above, we have*

$$g(f^n(1))g(f^n(2)) = g(f^n(5))g(f^n(6)), \tag{4.12}$$
$$g(f^n(6))g(f^n(3)) = g(f^n(9))g(f^n(a)). \tag{4.13}$$

Proof. We prove this by a simultaneous induction on n. For $n = 0$ and $n = 1$, we have the following base cases.

$$\begin{aligned} g(1)g(2) &= g(5)g(6) = 00, \\ g(f(1))g(f(2)) &= g(f(5))g(f(6)) = 010, \\ g(6)g(3) &= g(9)g(a) = 01, \\ g(f(6))g(f(3)) &= g(f(9))g(f(a)) = 000. \end{aligned}$$

Assume that the hypothesis is true for all $k \leq n$. Consider the inductive steps:

$$\begin{aligned} g(f^{n+1}(1))g(f^{n+1}(2)) &= g(f^n(234)) \\ &= g(f^n(2))g(f^{n-1}(563)) \\ &= g(f^n(2))g(f^n(7))g(f^{n-1}(63)) \\ &= g(f^n(2))g(f^n(7))g(f^{n-1}(9a)) \qquad \text{using induction hypothesis} \\ &= g(f^n(2))g(f^n(7))g(f^n(8)) \\ &= g(f^n(27))g(f^n(8)) \\ &= g(f^{n+1}(5))g(f^{n+1}(6)). \end{aligned}$$

$$\begin{aligned} g(f^{n+1}(6))g(f^{n+1}(3)) &= g(f^n(856)) \\ &= g(f^n(8))g(f^n(5))g(f^n(6)) \\ &= g(f^n(8))g(f^n(1))g(f^n(2)) \qquad \text{using induction hypothesis} \\ &= g(f^{n+1}(9))g(f^n(2)) \\ &= g(f^{n+1}(9))g(f^{n+1}(a)). \end{aligned}$$

□

Theorem 4.10. *The word* $\mathbf{w} = \tau(\varphi^\omega(0)) = g(f^\omega(0))$.

Proof. For $n > 0$, we claim that

$$\begin{aligned} \tau(\varphi^n(0)) &= g(f^{2n+1}(0))g(f^{2n+1}(5)), \\ \tau(\varphi^n(1)) &= g(f^{2n-2}(0))g(f^{2n+1}(5)). \end{aligned} \tag{4.14}$$

The proof is again by a simultaneous induction on n. The base case for $n = 1$ can be checked by hand. Assume that the hypothesis is true for all $k \leq n$. For the inductive step, we have the first statement of the claim,

$$\begin{aligned}\tau(\varphi^{n+1}(0)) &= \tau(\varphi^{n}(001))\\ &= g(f^{2n+1}(05))^2 g(f^{2n-2}(0))g(f^{2n+1}(5))\\ &= g(f^{2n}(0127))^2 g(f^{2n-2}(0))g(f^{2n}(27))\\ &= g(f^{2n-1}(012345))^2 g(f^{2n-2}(0))g(f^{2n-1}(345))\\ &= g(f^{2n-2}(0123456327))g(f^{2n-2}(0123456327))g(f^{2n-2}(0))g(f^{2n-2}(56327))\\ &= g(f^{2n-2}(0123456327))g(f^{2n-2}(8563459a27))g(f^{2n-2}(8))g(f^{2n-2}(56327)).\end{aligned}$$

Using Lemmas 4.8 and 4.9, we get

$$\begin{aligned}\tau(\varphi^{n+1}(0)) &= g(f^{2n-2}(0123456327856))g(f^{2n-2}(3459a27856327))\\ &= g(f^{2n-1}(01234563))g(f^{2n-1}(27856345))\\ &= g(f^{2n}(01234))g(f^{2n}(56327))\\ &= g(f^{2n+1}(012))g(f^{2n+1}(345))\\ &= g(f^{2n+2}(01))g(f^{2n+2}(27))\\ &= g(f^{2n+3}(0))g(f^{2n+3}(5)).\end{aligned}$$

For the inductive step on the second statement of the claim, we have

$$\begin{aligned}\tau(\varphi^{n+1}(1)) &= \tau(\varphi^{n}(01))\\ &= g(f^{2n+1}(0))g(f^{2n+1}(5))g(f^{2n-2}(0))g(f^{2n+1}(5))\\ &= g(f^{2n}(0))g(f^{2n}(1))g(f^{2n+1}(5))g(f^{2n-2}(0))g(f^{2n+1}(5))\\ &= g(f^{2n}(0))g(f^{2n-2}(34))g(f^{2n-2}(56327))g(f^{2n-2}(0))g(f^{2n-2}(56327))\\ &= g(f^{2n}(0))g(f^{2n-2}(3456327056327)).\end{aligned}$$

Using Lemmas 4.8 and 4.9, we get

$$\begin{aligned}\tau(\varphi^{n+1}(1)) &= g(f^{2n}(0))g(f^{2n-2}(3459a27856327))\\ &= g(f^{2n}(0))g(f^{2n-1}(27856345))\\ &= g(f^{2n}(0))g(f^{2n}(56327))\\ &= g(f^{2n}(0))g(f^{2n+1}(345))\\ &= g(f^{2n}(0))g(f^{2n+2}(27))\\ &= g(f^{2n}(0))g(f^{2n+3}(5)).\end{aligned}$$

Now we let $n \to \infty$ in Equation (4.14) to get $\tau(\varphi^{\omega}(0)) = g(f^{\omega}(0))$. □

4.3.2 Absence of Antisquares and the Critical Exponent

We now prove that the word **w** does not contain antisquares other than 01 and 10, and has the critical exponent $2 + \phi$. In the commands below, `FASQ` denotes the automaton in Figure 4.4.

```
eval AntisqLengths "?msd_fib Ei
    (p >= 1) &
    (Aj (j < p) => ~(FASQ[i + j] = FASQ[i + j + p]))";
```

The predicate `AntisqLengths` produces an automaton accepting only 1 as the input. This shows that only antisquares of length 2 are present in the word, and they are 01 and 10.

We now compute the periods that are associated with factors that have exponent >= 3.

```
eval HighPowPeriods "?msd_fib Ei
    (p >= 1) &
    (Aj (j <= 2*p) => FASQ[i + j] = FASQ[i + j + p])";
```

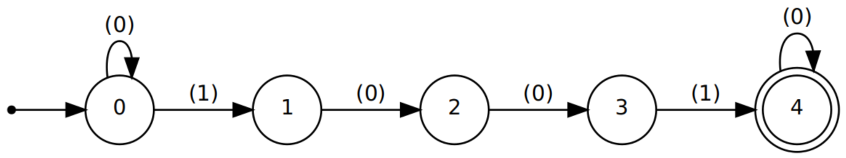

Figure 4.5: Periods associated with cubes and higher powers in **w**.

The predicate above produces the automaton in Figure 4.5, which shows that these periods are of the form `0*10010*` in Fibonacci representation. Next, we compute the pairs (n, p) such that **w** contains a factor of length $n + p$ with period p of the form `0*10010*` and $n + p$ is the longest length of any factor with this period.

```
reg Pows msd_fib "0*10010*";
def MaximalReps "?msd_fib Ei
    (Aj (j < n) => FASQ[i + j] = FASQ[i + j + p]) &
    (FASQ[i + n] != FASQ[i + n + p])";
eval HighestPowersW "?msd_fib (p >= 1) & $Pows(p) &
    $MaximalReps(n, p) & (Am $MaximalReps(m, p) => m <= n)";
```

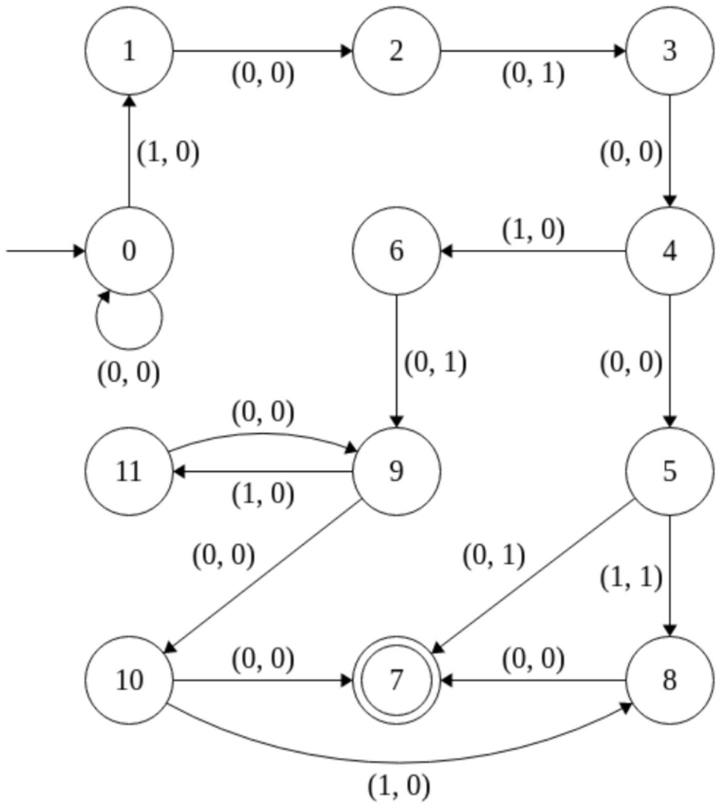

Figure 4.6: Pairs (n, p) accepted by the predicate `HighestPowersW`.

The automaton produced by the predicate `HighestPowersW` is given in Figure 4.6.

The larger pairs (n, p) accepted by this automaton have the form

$$\binom{0}{0}^*\binom{1}{0}\binom{0}{0}\binom{0}{1}\binom{0}{0}\binom{1}{0}\binom{0}{1}\left\{\binom{1}{0}\binom{0}{0}\right\}^*\binom{0}{0}\left\{\epsilon, \binom{1}{0}\right\}\binom{0}{0}.$$

From this format, we can deduce the following values for n and p. Consider the Fibonacci numbers denoted by F_k, where $F_0 = 0, F_1 = 1$, and $F_k = F_{k-1} + F_{k-2}$ for $k \geq 2$.

We have two cases for n,

$$n = F_{k+2} + F_{k-2} + F_{k-4} + \ldots + F_7 + F_5, \text{ or}$$
$$n = F_{k+2} + F_{k-2} + F_{k-4} + \ldots + F_8 + F_6 + F_3.$$

Both these expressions sum up to $2F_{k+1} - 3$. We have $p = 2F_{k-1}$. Thus, the exponent is

$$\begin{aligned} \frac{n+p}{p} &= \frac{2F_{k+1} + 2F_{k-1} - 3}{2F_{k-1}} && (4.15) \\ &= \frac{2F_k + 4F_{k-1} - 3}{2F_{k-1}} && (4.16) \\ &= 2 + \frac{2F_k - 3}{2F_{k-1}}. && (4.17) \end{aligned}$$

Dividing the numerator and denominator in Equation (4.17) by $2F_{k-1}$ and taking the limit $k \to \infty$, we see that the critical exponent is $2 + \phi$, since the ratio of consecutive Fibonacci numbers converges to ϕ.

4.4 Properties of Lucas Words

In [3], the authors introduced what they called periodic words (or LLP-words) connected with the Lucas numbers and investigated their properties. In this section, we show how our decision procedure lets us study the same properties purely by machine computation.

Analogous to the definition of Fibonacci numbers, the Lucas numbers are given by $L_0 = 2$, $L_1 = 1$, and $L_n = L_{n-1} + L_{n-2}$ for $n \geq 2$. Similarly, analogous to the infinite Fibonacci word, the infinite Lucas word $\mathbf{u}$ is given by the limit of u_n at $n \to \infty$, where $u_0 = 10$, $u_1 = 1$, and $u_n = u_{n-1}u_{n-2}$ for $n \geq 2$. The combinatorial properties of the Fibonacci and Lucas infinite words are of great interest in mathematics and physics, including topics like number theory, fractal geometry, cryptography, and quasicrystals. For details, we refer to Lothaire and Pirillo [34, 43].

Theorem 4.11. *([3, Theorem 3]) The infinite Lucas word* $\mathbf{u}$ *and the finite Lucas words* u_n *satisfy the following properties.*

1. *The words* 1111 *and* 00 *are not subwords of* $\mathbf{u}$.
2. *For all* $n > 1$, *if* ab *are the last two symbols of* u_n, *then* $ab = 10$ *if* n *is even, and* $ab = 01$ *if* n *is odd.*
3. *For all* n, $|u_n| = L_n$.

The authors also state the hypothesis that the infinite Lucas word is aperiodic. We prove statements 1 and 2 in Theorem 4.11 along with this hypothesis purely mechanically using our decision procedure. To enable working in `Walnut`, we begin by constructing an automaton that produces the Lucas word. Since the recurrence relation associated with Lucas numbers is the same as Fibonacci numbers, the infinite Lucas word is Fibonacci-automatic, that is, the corresponding DFAO reads in as input an integer N in the Fibonacci numeration system and outputs the symbol $\mathbf{u}[N]$. Figure 4.7 shows the DFAO producing $\mathbf{u} = h(g^{\omega}(0))$, where morphisms g and h are defined as follows:

$$\begin{array}{llll} g\colon & 0 \mapsto 01 & h\colon & 0 \mapsto 1 \\ & 1 \mapsto 2 & & 1 \mapsto 1 \\ & 2 \mapsto 43 & & 2 \mapsto 0 \\ & 3 \mapsto 0 & & 3 \mapsto 1 \\ & 4 \mapsto 65 & & 4 \mapsto 1 \\ & 5 \mapsto 4 & & 5 \mapsto 0 \\ & 6 \mapsto 87 & & 6 \mapsto 1 \\ & 7 \mapsto 8 & & 7 \mapsto 0 \\ & 8 \mapsto 29 & & 8 \mapsto 1 \\ & 9 \mapsto 6, & & 9 \mapsto 1. \end{array}$$

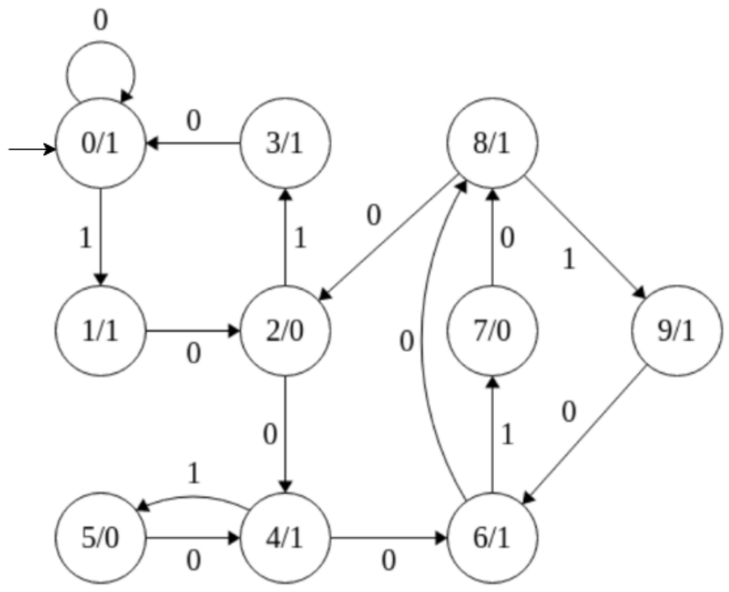

Figure 4.7: DFAO computing the infinite Lucas word.

In the following `Walnut` commands, `LUCAS` denotes the automaton in Figure 4.7. The hypothesis about the infinite Lucas word being aperiodic can be proved with a simple predicate that negates the existence of any pair of integers $n \geq 0, p \geq 1$ such that $\mathbf{u}[i] = \mathbf{u}[i+p]$ for all $i \geq n$. In other words, the predicate asserts that $\mathbf{u}$ is not ultimately periodic, and hence, aperiodic. It produces the `true` automaton, showing that $\mathbf{u}$ is indeed aperiodic.

```
eval LucasAperiodic "?msd_fib
    ~En,p (p>=1) & (Ai (i>=n) => (LUCAS[i] = LUCAS[i+p]))";
```

Next, we show that 1111 and 00 are not subwords of $\mathbf{u}$. Both the following commands produce a `true` automaton, proving the statement.

```
eval AbsetZero2 "?msd_fib ~Ei LUCAS[i]=@0 & LUCAS[i+1]=@0";
eval AbsentOne4 "?msd_fib ~Ei
    LUCAS[i]=@1 & LUCAS[i+1]=@1 &
    LUCAS[i+2]=@1 & LUCAS[i+3]=@1";
```

To prove statement 2 of Theorem 4.11, we use the fact that $|u_n| = L_n$. Notice that Lucas numbers L_n are related to Fibonacci numbers F_n, and the following relation between them is known.

$$L_n = F_{n-1} + F_{n+1}$$

The Fibonacci numeration system uses F_2 as the least significant value, so we prove the statement for L_2 separately, and for $L_n, n \geq 3$ together. For even $n \geq 3$, L_n is of the form $1010(00)^*$ in the Fibonacci numeration system, while for odd $n \geq 3$, it is of the form $101(00)^*$. First, we define these two regular expressions to denote L_n for even and odd n.

```
reg LnEven msd_fib "0*1010(00)*";
reg LnOdd msd_fib "0*101(00)*";
```

Next, we use the command below to assert the required statement, which produces the `true` automaton, and hence, proves the result.

```
eval Alternate01 "?msd_fib An
    ($LnEven(n) => (LUCAS[n-2]=@1 & LUCAS[n-1]=@0)) &
    ($LnOdd(n) => (LUCAS[n-2]=@0 & LUCAS[n-1]=@1))";
```

Chapter 5

Open Problems

5.1 Stronger Decidability Results

We can decide any first-order predicate concerning addition, comparison, and indexing into an Ostrowski-automatic word using our procedure. However, these decision procedures work for a predicate only if all sequences in the predicate are automatic in the same numeration system. We also propose that stronger decidability results are obtainable. In [51], the authors showed that for standard base-k automatic sequences, the critical exponent is always either a rational number or infinite, and its value is computable. A similar result for Ostrowski-automatic sequences should be obtainable, as suggested by several examples in Chapter 4.

5.1.1 The Language of Quotients

Consider an Ostrowski numeration system based on the irrational number α. Given a finite word $w \in (\Sigma_\alpha^2)^*$, the *projections* $\pi_i(w)$ for $i = 1, 2$ are defined onto the i^{th} coordinate. For example, consider the pair of integers $(6, 3)$, which has the following 2-dimensional representation in the Fibonacci base:

$$x = [\mathtt{1}, \mathtt{0}][\mathtt{0}, \mathtt{1}][\mathtt{0}, \mathtt{0}][\mathtt{1}, \mathtt{0}].$$

Here, $\pi_1(x) = \mathtt{1001}$ and $\pi_2(x) = \mathtt{0101}$. Note that $[\pi_1(x)]_\alpha = 6$ and $[\pi_2(x)]_\alpha = 3$. For a finite word w with $[\pi_2(w)]_\alpha \neq 0$, we define the *quotient* as

$$\text{quo}_\alpha(w) = \frac{[\pi_1(w)]_\alpha}{[\pi_2(w)]_\alpha}.$$

In [51, Section 4], the authors showed that for a standard base-k number system, there exists an algorithm that, given a DFA accepting $L \subseteq (\Sigma_k^2)^*$, will compute $\gamma = \sup \mathrm{quo}_k(L)$. They also introduce the idea of a *special point.* We present an equivalent definition with respect to the Fibonacci number system.

Let $L \subseteq (\{0,1\}^2)^*$ be a language. We define the set

$$S = \{(x, y) : x = [\pi_1(w)]_F, y = [\pi_2(w)]_F \text{ for all } w \in L\}.$$

Then the language of quotients over L is defined as follows:

$$\mathrm{quo}_F(L) = \left\{\frac{x}{y} : (x, y) \in S\right\}.$$

5.1.2 Computing the Largest Special Point

First, we define a special point in the Fibonacci number system for a language, analogous to the definition given by Schaeffer and Shallit in [51] for base-k number systems.

Definition 5.1. Let $L \subseteq (\Sigma_2^2)^*$ be a language. We say a real number β is a special point of $\mathrm{quo}_F(L)$ if there exists an infinite sequence $(x_j)_{j\geq 1}$ of distinct words of L such that $\lim_{j\to\infty} \mathrm{quo}_F(x_j) = \beta$.

Note that every infinite language L has a special point, and indeed, the largest special point. Schaeffer and Shallit showed that when the integer values are expressed in a standard base-k number system, then the largest special point is computable. In fact, they proved something more general, that the largest special point is computable if the integer values are expressed as a sum of the powers of a constant. We extend this idea to the Fibonacci number system. The irrational number associated with this system is $\phi - 1$, and the corresponding alphabet is $\{0, 1\}$. Consider a finite word $z \in \{0,1\}^*$ in the Fibonacci representation. We can express $[z]_F$ as the sum of a finite set of Fibonacci numbers. Let $[z]_F = \sum_{2\leq i\leq k} a_i F_i$, where $a_i \in \{0,1\}$ and the F_i are the Fibonacci numbers. We define a function $f : \mathbb{N} \to \mathbb{R}$ such that

$$f([z]_F) = \frac{1}{\sqrt{5}} \sum_{2\leq i\leq k} a_i \phi^i.$$

Note that $f([z]_F)$ is a real-valued approximation for the integer $[z]_F$ since we have that

$$[z]_F = \frac{1}{\sqrt{5}} \sum_{2\leq i\leq k} a_i(\phi^i - (1-\phi)^i).$$

We now define another set and the corresponding set of quotients as follows:

$$S' = \{(u, v) : u = f(x), v = f(y) \text{ for all } (x, y) \in S\}, \tag{5.1}$$

$$\text{quo}_\phi(L) = \left\{\frac{x}{y} : (x, y) \in S'\right\}. \tag{5.2}$$

Lemma 5.1. *Let $w \in \{0,1\}^*$ be a word. Then*

$$|[w]_F - f([w]_F)| \leq \frac{1}{\sqrt{5}}.$$

Proof. Let $[w]_F = \sum_{2\leq i\leq k} a_i F_i$, where $a_i \in \{0,1\}$. We have that

$$\begin{aligned}
[w]_F &= \frac{1}{\sqrt{5}} \sum_{2\leq i\leq k} a_i(\phi^i - (1-\phi)^i) \\
&= \frac{1}{\sqrt{5}} \sum_{2\leq i\leq k} a_i\phi^i - \frac{1}{\sqrt{5}} \sum_{2\leq i\leq k} a_i(1-\phi)^i \\
&= f([w]_F) - \frac{1}{\sqrt{5}} \sum_{2\leq i\leq k} a_i(1-\phi)^i.
\end{aligned}$$

Therefore, we have that

$$\begin{aligned}
|[w]_F - f([w]_F)| &= \left|\frac{1}{\sqrt{5}} \sum_{2\leq i\leq k} a_i(1-\phi)^i\right| \\
&\leq \frac{1}{\sqrt{5}} \sum_{2\leq i\leq k} a_i|1-\phi|^i \\
&\leq \frac{1}{\sqrt{5}} \sum_{2\leq i\leq \infty} |1-\phi|^i \\
&\leq \frac{1}{\sqrt{5}} \left(\frac{(1-\phi)^2}{1-|1-\phi|}\right) \\
&\leq \frac{1}{\sqrt{5}}.
\end{aligned}$$

□

For any $w \in \{0,1\}^*$, we observe that the ratio of $f([w]_F)$ and $[w]_F$ is bounded as follows:

$$\frac{f([w]_F)}{[w]_F} = \frac{[w]_F \pm |[w]_F - f([w]_F)|}{[w]_F} = 1 \pm \frac{|[w]_F - f([w]_F)|}{[w]_F}. \tag{5.3}$$

Using Lemma 5.1 on Equation (5.3), we obtain

$$\lim_{[w]_F \to \infty} \frac{f([w]_F)}{[w]_F} = 1. \tag{5.4}$$

As a result, we have the following theorem.

Theorem 5.2. *Let $L \subseteq (\Sigma_2^2)^*$ be a language. A real number β is a special point of* $\text{quo}_F(L)$ *if and only if it is also a special point of* $\text{quo}_\phi(L)$.

Proof. Consider the special point β of $\text{quo}_F(L)$. By definition, there exists an infinite sequence of distinct pairs of words $(x_j, y_j)_{j\geq 1}$ in L, such that $\lim_{j\to\infty}([x_j]_F/[y_j]_F) = \beta$. We now consider the corresponding quotient of the approximated real numbers for $[x]_F$ and $[y]_F$. We have, using Equation (5.4):

$$\lim_{j\to\infty} \frac{f([x_j]_F)}{f([y_j]_F)} = \lim_{j\to\infty} \frac{[x_j]_F}{[y_j]_F} = \beta.$$

A similar argument in the other direction shows that the special points of both the languages $\text{quo}_F(L)$ and $\text{quo}_\phi(L)$ coincide. $\square$

Thus, we have shown that the special points of the language $\text{quo}_F(L)$ remain the same if the language is replaced by its real approximation $\text{quo}_\phi(L)$. The methods of Schaeffer and Shallit in [51] can now be used to compute the largest special point if the language L is Fibonacci-automatic, as the values in $\text{quo}_\phi(L)$ are expressed as a sum of powers of the constant ϕ. We leave it as an open problem to determine if the critical exponent is also computable in the Fibonacci number system and other Ostrowski systems in general.

5.2 General Implementation for an Irrational Number

Our implementation in Chapter 3 can only generate automata recognizing the addition relation for Ostrowski numeration systems based on quadratic irrational numbers. Recall that this implementation generates an automaton that takes three inputs x, y, z in the corresponding Ostrowski numeration system, and accepts if and only if $x + y = z$. The crucial problem with a more general implementation is that the continued fraction expansion of an arbitrary irrational number α may have unbounded partial quotients, and hence, may not be ultimately periodic.

To work around this problem, a 4-input implementation is required such that all the four inputs: the partial quotients of the continued fraction, x, y, and z are encoded in a common numeration system that the underlying framework understands. For example, if all the four inputs are encoded in binary and are read in parallel, then we can simulate the adder given in Figure 2.3 in its entirety without the restriction of α being a quadratic irrational.

Note that the adder automaton in Figure 2.3 decides a transition only based on how the difference $z_{i-1} - (x_{i-1} + y_{i-1})$ compares to d_i, the corresponding term in the continued fraction. To compare the two quantities, the only requirement is that both are expressed in the same numeration system, hence, in principle, we can encode the four inputs in binary and replace the transitions in the automaton in Figure 2.3 with automata that recognize the corresponding set of four inputs in binary.

5.3 Higher-Order Numeration Systems

Ostrowski numeration systems are characterized by an irrational number α (see Section 2.1). We can introduce more irrational numbers to extend this idea to a broader class of numeration systems. Consider an order-m numeration system characterized by a sequence of m irrational numbers, $\Gamma = (\alpha_1, \alpha_2, \ldots, \alpha_m)$. For $1 \leq k \leq m$, let $[0; d_{k,1}, d_{k,2}, \ldots]$ be the continued fraction of α_k. We define a sequence $(q_i)_{i \geq 0}$ as follows:

$$q_i = \begin{cases} 0, & \text{if } i < 0; \\ 1, & \text{if } i = 0; \\ \sum_{1 \leq k \leq m} d_{k,i} q_{i-k}, & \text{otherwise.} \end{cases}$$

We can now represent an arbitrary integer N in this order-m numeration system defined by the sequence Γ as follows:

$$N = [a_{n-1} a_{n-2} \cdots a_0]_\Gamma = \sum_{0 \leq i < n} a_i q_i.$$

However, to have any practical utility, we also require the following rules for enforcing a canonical representation:

1. $a_0 < d_{1,1}$;
2. $0 \leq a_i \leq d_{1,i+1}$, for $i \geq 1$; and
3. for all $i \geq 1$, if $a_i = d_{1,i+1}$ then there exists a $k \leq m$ such that $a_{i-k} < d_{k,i+1}$.

Clearly, the Ostrowski numeration system is an example of an order-2 numeration system with $\Gamma = (\alpha, \phi - 1)$, where $\phi = (1 + \sqrt{5})/2$ is the golden ratio. The second irrational number is $\phi - 1$ because the continued fraction for $\phi - 1 = [0; \overline{1}]$; therefore, if $\alpha = [d_0; d_1, \ldots]$, then we have that $q_i = d_i q_{i-1} + q_{i-2}$. We propose the open problem to construct an adder for a generalized order-m numeration system. A possible direction to explore is to design an automaton with m-dimensional states and produce a result analogous to Theorem 2.2.

5.4 Repetition Threshold for Infinite Words

Another open problem is to determine the repetition threshold for several classes of infinite words. In Section 4.1, we resolved a previous conjecture by Rampersad et al. for infinite balanced words for alphabets of size ≤ 8, but the problem remains unsolved in general. For infinite rich words, we resolved the problem for the binary case, but it remains unsolved for larger alphabets.

Before Rampersad et al. [45] proved our conjecture about infinite rich words, we relied on a backtracking search to obtain the lower bound and construct the candidate infinite word. Our backtracking algorithm utilizes the `EERTREE` data structure given by Rubinchik and Shur [49]. Using this data structure, if we are given the number of distinct palindromes for a word w over an alphabet Σ, we can find the number of distinct palindromes in the word wa for all $a \in \Sigma$ in constant amortized time. Our computation suggests that the repetition threshold for rich words, $RRT(k)$, for larger alphabets might have the following lower bounds:

$$\begin{aligned} RRT(3) &\geq \frac{9}{4}, \\ RRT(4) &\geq \frac{11}{5}, \text{ and} \\ RRT(5) &\geq \frac{13}{6}. \end{aligned}$$

In general, for an alphabet of size $k \geq 3$, it seems that $RRT(k) \geq (2k+3)/(k+1)$, which agrees with Theorem 4.3. We leave it as an open problem to prove this and determine the exact value.

In [15], Chen et al. gave a survey of fast space-efficient algorithms for computing all maximal runs in a string. They also proposed some new and faster algorithms for this task. A possible direction for future work is to implement these algorithms in the backtracking search. Faster computation of maximal runs will help us to efficiently reject those paths in the backtracking search that violate the critical exponent threshold. Thus, we may be able to compute tighter lower bounds on the repetition threshold for larger alphabets.

Based on a suggestion by Edita Pelantová, we note another interesting direction to explore in this regard. The word we constructed in Figure 4.1 is a complementary symmetric Rote word [48]. We say that an infinite word $u = u_0u_1u_2\cdots$ over the alphabet $\{0, 1\}$ is a complementary symmetric Rote word if and only if there exists a Sturmian word $v = v_0v_1v_2\cdots$, such that $v_i = u_i + u_{i+1} \bmod 2$ for all i. The Sturmian word associated with our word $\mathbf{r}$ is the one fixed by the Sturmian substitution:

$$\begin{aligned}\xi\colon\quad 0 &\mapsto 011\\ 1 &\mapsto 01.\end{aligned}$$

It is known that complementary symmetric Rote words are rich in palindromes [8]. Therefore, a possible direction to explore the construction of infinite rich words using Sturmian words over larger alphabets. These words could serve as good candidates for achieving the repetition threshold. There has been a very recent development in this regard by Dvořáková et al. [25], where the authors determine the critical exponent and the recurrence function of complementary symmetric Rote sequences based on the study of return words to *bispecial* factors of Sturmian sequences. A factor w of a word $\mathbf{u}$ is called right-special if there exist two distinct characters a, b such that wa, wb are factors of $\mathbf{u}$. Left-special is defined symmetrically. Bispecial words are those that are both left and right special.

Another natural extension to the problem is to ask the same question on several other classes of infinite words; e.g., words avoiding certain patterns. We explore such a class in Section 4.3 where we study repetitions in infinite binary words avoiding large antisquares.

References

[1] J. P. Allouche and J. Shallit. *Automatic Sequences—Theory, Applications, Generalizations.* Cambridge University Press, 2003.

[2] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

[3] G. Barabash, Y. Kholyavka, and I. Tytar. Periodic words connected with the Lucas numbers. *Вісник Львівського університету. Серія механіко-математична*, 84:62–66, 2017.

[4] A. R. Baranwal and J. Shallit. Critical exponent of infinite balanced words via the Pell number system. In Robert Mercaş and Daniel Reidenbach, editors, *Combinatorics on Words*, pages 80–92, Cham, 2019. Springer International Publishing.

[5] A. R. Baranwal and J. Shallit. Repetitions in infinite palindrome-rich words. In Robert Mercaş and Daniel Reidenbach, editors, *Combinatorics on Words*, pages 93–105, Cham, 2019. Springer International Publishing.

[6] J. P. Bell, T. F. Lidbetter, and J. Shallit. Additive number theory via approximation by regular languages. In *International Conference on Developments in Language Theory*, pages 121–132. Springer, 2018.

[7] J. Berstel and P. Séébold. Sturmian words. In M. Lothaire, editor, *Algebraic Combinatorics on Words*, volume 90 of *Encyclopedia of Mathematics and Its Applications*, pages 45–110. Cambridge University Press, 2002.

[8] A. Blondin Massé, S. Brlek, S. Labbé, and L. Vuillon. Palindromic complexity of codings of rotations. *Theoret. Comput. Sci.*, 412:6455–6463, 2011.

[9] P. Bonardo, A. E. Frid, and J. Shallit. The number of valid factorizations of Fibonacci prefixes. *Theoret. Comput. Sci.*, 775:68–75, 2019.

[10] S. Brlek, S. Hamel, M. Nivat, and C. Reutenauer. On the palindromic complexity of infinite words. *Internat. J. Found. Comp. Sci.*, 15:293–306, 2004.

[11] V. Bruyère and G. Hansel. Recognizable sets of numbers in nonstandard bases. In R. Baeza-Yates, E. Goles, and P. V. Poblete, editors, *LATIN '95: Theoretical Informatics*, volume 911 of *Lecture Notes in Computer Science*, pages 167–179. Springer-Verlag, 1995.

[12] V. Bruyère, G. Hansel, C. Michaux, and R. Villemaire. Logic and p-recognizable sets of integers. *Bull. Belg. Math. Soc.*, 1:191–238, 1994. Corrigendum, *Bull. Belg. Math. Soc.* **1** (1994), 577.

[13] M. Bucci, A. De Luca, A. Glen, and L. Q. Zamboni. A new characteristic property of rich words. *Theoret. Comput. Sci.*, 410:2860–2863, 2009.

[14] J. R. Büchi. Weak secord-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960. Reprinted in S. Mac Lane and D. Siefkes, eds., *The Collected Works of J. Richard Büchi*, Springer-Verlag, 1990, pp. 398–424.

[15] G. Chen, S. J. Puglisi, and W. F. Smyth. Fast & practical algorithms for computing all the runs in a string. In B. Ma and K. Zhang, editors, *CPM 07*, volume 4580 of *Lecture Notes in Computer Science*, pages 307–315. Springer-Verlag, 2007.

[16] G. Christol. Ensembles presque périodiques k-reconnaissables. *Theoret. Comput. Sci.*, 9:141–145, 1979.

[17] A. Cobham. On the base-dependence of sets of numbers recognizable by finite automata. *Math. Systems Theory*, 3:186–192, 1969.

[18] A. Cobham. Uniform tag sequences. *Math. Systems Theory*, 6:164–192, 1972.

[19] J. Currie, T. Harju, P. Ochem, and N. Rampersad. Some further results on squarefree arithmetic progressions in infinite words. *Theoret. Comput. Sci.*, 799:140–148, 2019.

[20] J. Currie and N. Rampersad. A proof of Dejean's conjecture. *Math. Comp.*, 80(274):1063–1070, 2011.

[21] A. de Luca, A. Glen, and L. Q. Zamboni. Rich, Sturmian, and trapezoidal words. *Theoret. Comput. Sci.*, 407:569–573, 2008.

[22] F. Dejean. Sur un théorème de Thue. *J. Combin. Theory. Ser. A*, 13(1):90–99, 1972.

[23] C. F. Du, H. Mousavi, E. Rowland, L. Schaeffer, and J. Shallit. Decision algorithms for Fibonacci-automatic words, II: related sequences and avoidability. *Theoret. Comput. Sci.*, 657:146–162, 2017.

[24] C. F. Du, H. Mousavi, L. Schaeffer, and J. Shallit. Decision algorithms for Fibonacci automatic words, III: enumeration and abelian properties. *Internat. J. Found. Comp. Sci.*, 27(8):943–963, 2016.

[25] L. Dvořáková, K. Medková, and E. Pelantová. Complementary symmetric Rote sequences: the critical exponent and the recurrence function. *arXiv preprint arXiv:2003.06916*, 2020.

[26] S. Eilenberg. *Automata, Languages, and Machines*, volume A. Academic Press, 1974.

[27] J. E. Foster. A number system without a zero-symbol. *Math. Mag.*, 21(1):39–41, 1947.

[28] C. Frougny and B. Solomyak. On representation of integers in linear numeration systems. In M. Pollicott and K. Schmidt, editors, *Ergodic Theory of $\mathbb{Z}^d$ Actions (Warwick, 1993–1994)*, volume 228 of *London Mathematical Society Lecture Note Series*, pages 345–368. Cambridge University Press, 1996.

[29] A. Glen, J. Justin, S. Widmer, and L. Q. Zamboni. Palindromic richness. *European J. Combinatorics*, 30:510–531, 2009.

[30] C. Guo, J. Shallit, and A. M. Shur. Palindromic rich words and run-length encodings. *Inform. Process. Lett.*, 116:735–738, 2016.

[31] P. Hieronymi and A. Terry Jr. Ostrowski numeration systems, addition, and finite automata. *Notre Dame J. Formal Logic*, 59(2):215–232, 2018.

[32] J. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In *Theory of Machines and Computations*, pages 189–196. Elsevier, 1971.

[33] P. Hubert. Suites équilibrées. *Theoret. Comput. Sci.*, 242:91–108, 2000.

[34] M. Lothaire. *Algebraic Combinatorics on Words*, volume 90 of *Encyclopedia of Mathematics and Its Applications.* Cambridge University Press, 2002.

[35] R. McNaughton. Reviewed works: Weak second-order arithmetic and finite automata by J. Richard Büchi; on a decision method in restricted second order arithmetic by J. Richard Büchi. *J. Symbolic Logic*, 28(1):100–102, 1963.

[36] H. Mousavi. Automatic theorem proving in Walnut. *arXiv preprint arXiv:1603.06017*, 2016.

[37] H. Mousavi, L. Schaeffer, and J. Shallit. Decision algorithms for fibonacci-automatic words, I: basic results. *RAIRO Inform. Théor. App.*, 50(1):39–66, 2016.

[38] H. Mousavi and J. Shallit. Mechanical proofs of properties of the Tribonacci word. In *International Conference on Combinatorics on Words*, pages 170–190. Springer, 2015.

[39] T. Ng, P. Ochem, N. Rampersad, and J. Shallit. New results on pseudosquare avoidance. In *International Conference on Combinatorics on Words*, pages 264–274. Springer, 2019.

[40] A. Ostrowski. Bemerkungen zur Theorie der Diophantischen Approximationen. *Abh. Math. Sem. Hamburg*, 1:77–98, 250–251, 1922. Reprinted in *Collected Mathematical Papers*, Vol. 3, pp. 57–80.

[41] E. Pelantová and S. Starosta. Languages invariant under more symmetries: Overlapping factors versus palindromic richness. *Discrete Math.*, 313:2432–2445, 2013.

[42] E. Pelantová and Š. Starosta. Constructions of words rich in palindromes and pseudopalindromes. *Discrete Math. & Theoret. Comput. Sci.*, 18:Paper #16, 2016. Available at https://dmtcs.episciences.org/2202.

[43] G. Pirillo. Fibonacci numbers and words. In *Séminaire Lotharingien de Combinatoire (Gerolfingen, 1993)*, number 34 in Prépubl. Inst. Rech. Math. Av., Univ. Louis Pasteur, Strasbourg, 1993, pages 77–85, 1993.

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

[45] N. Rampersad, L. Mol, and J. D. Currie. The repetition threshold for binary rich words. *Discrete Mathematics & Theoretical Computer Science*, 22(1), 2020.

[46] N. Rampersad, J. Shallit, and É. Vandomme. Critical exponents of infinite balanced words. *Theoret. Comput. Sci.*, 777:454–463, 2019.

[47] M. Rao. Last cases of Dejean's conjecture. *Theoret. Comput. Sci.*, 412(27):3010–3018, 2011.

[48] G. Rote. Sequences with subword complexity $2n$. *J. Number Theory*, 46:196–213, 1994.

[49] M. Rubinchik and A. M. Shur. EERTREE: an efficient data structure for processing palindromes in strings. In *International Workshop on Combinatorial Algorithms*, pages 321–333. Springer, 2015.

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

[51] L. Schaeffer and J. Shallit. The critical exponent is computable for automatic sequences. *Internat. J. Found. Comp. Sci.*, 23(08):1611–1626, 2012.

[52] L. Schaeffer and J. Shallit. Closed, palindromic, rich, privileged, trapezoidal, and balanced words in automatic sequences. *Electronic J. Combinatorics*, 23, 2016.

[53] J. Shallit. Decidability and enumeration for automatic sequences: A survey. In Andrei A. Bulatov and Arseny M. Shur, editors, *Computer Science – Theory and Applications*, pages 49–63, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[54] J. Shallit and R. Zarifi. Circular critical exponents for thue-morse factors. *RAIRO Inform. Théor. App.*, 53(1-2):37–49, 2019.

[55] A. Thue. Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen. *Norske vid. Selsk. Skr. Mat. Nat. Kl.*, 1:1–67, 1912. Reprinted in *Selected Mathematical Papers of Axel Thue*, T. Nagell, editor, Universitetsforlaget, Oslo, 1977, pp. 413–478.

[56] J. Vesti. Extensions of rich words. *Theoret. Comput. Sci.*, 548:14–24, 2014.

[57] J. Vesti. Rich square-free words. *Theoret. Comput. Sci.*, 687:48–61, 2017.

[58] L. Vuillon. Balanced words. *Bull. Belg. Math. Soc.*, 10(5):787–805, 12 2003.

[59] E. Zeckendorf. Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas. *Bull. Soc. Roy. Liège*, 41:179–182, 1972.